Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Label-Dependent Splitting for Multi-Label Data

by
Annegret Muller

Dissertation presented for the degree of Doctor of Philosophy in Mathematical Statistics in the Faculty of Economic and Management Sciences at Stellenbosch University

Supervisor: Prof S.J. Steel
Co-supervisor: Dr T. Sandrock

December 2023

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Annegret Muller

Date: December 2023

Signature: (Declaration with signature in possession of candidate and supervisors.)

Abstract

Multi-label classification problems arise in scenarios where every data case can be associated with multiple labels simultaneously. Compared to single-label data, multi-label data possess unique characteristics which result in additional challenges when analysing the data. The aim of this dissertation is to address two of these challenging aspects of multi-label data. The first is the exploitation of label correlations to achieve accurate classification of unseen data cases. Secondly, strategies for input variable ranking within multi-label data are considered to allow for more interpretable results.

Effective exploitation of correlation amongst labels can be a vital attribute of an accurate multi-label classification method. However, label correlations are not necessarily shared globally by all data cases. Despite this, existing methods mostly focus on global exploitation of label correlations. Therefore, a new tree-based ensemble method for multi-label classification is proposed in this dissertation, Label-Dependent splitting (LDsplit). LDsplit aims to implicitly exploit *local* higher-order label correlations within multi-label data by dividing the data into subgroups. The algorithm fits an ensemble of trees based on differently ordered label subsets. For each tree, different labels are used at different levels of the tree, as determined by the label order applicable to that tree. The tree-levels are made up of nodes that are split using any binary classifier. Since a tree-level depends on its label as well as previous splits made when parent nodes were formed using other labels, higher-order label correlations are implicitly incorporated into the model in a simple manner. Depending on whether random or predetermined label orders are used to fit the ensemble, either Random LDsplit or Conditional LDsplit is fit. An extensive empirical study is performed on a range of multi-label benchmark datasets. The empirical evidence shows that despite the simple framework, both Random LDsplit and Conditional LDsplit offer very competitive classification performance in comparison with existing multi-label classification methods.

For multi-label data, an input variable is globally important if it is deemed important for several or all labels. However, an input variable can also be deemed locally important for a specific label. Few proposals for input variable ranking within multi-label data consider both global and local importance of variables. Moreover, existing methods mostly neglect to exploit label dependencies within the data. Therefore, different ways are outlined how an LDsplit ensemble can produce global and local input variable rankings and effectively allow for better interpretation of the data. Results obtained from synthetically generated multi-label datasets demonstrate that both the novel global and local importance measures give favourable performance.

Keywords: multi-label classification, ensemble method, tree, label order, label correlation, local label correlation, input variable ranking, global importance, local importance.

Opsomming

Multi-etiket klassifikasie probleme ontstaan in scenario's waar elke datageval gelyktydig met verskeie etikette geassosieer kan word. In vergelyking met enkel-etiket data, beskik multi-etiket data unieke eienskappe wat tot addisionele uitdagings lei wanneer die data ontleed word. Die doelwit van hierdie skripsie is om twee van hierdie uitdagende aspekte van multi-etiket data aan te spreek. Die eerste is die benutting van etiket-korrelasies, sodat akkurate klassifikasie van ongesiene datagevalle bereik kan word. Tweedens word strategieë vir inset-veranderlike rangskikking binne multi-etiket data beskou vir meer interpreteerbare resultate.

Effektiewe benutting van korrelasie tussen etikette kan 'n noodsaaklike eienskap van 'n akkurate multi-etiket klassifikasie-metode wees. Tog word etiket-korrelasies nie noodwendig globaal deur alle datagevalle gedeel nie. Ten spyte hiervan fokus bestaande metodes meestal op globale benutting van etiket-korrelasies. Daarom word 'n nuwe boom-gebaseerde ensemble metode vir multi-etiket klassifikasie in hierdie skripsie voorgestel, naamlik Etiket-Afhanklike splitting (LDsplit). LDsplit beoog om implisiet *plaaslike* hoër-orde etiket-korrelasies binne multi-etiket data te benut deur die data in subgroepe te verdeel. Die algoritme pas 'n ensemble van bome gebaseer op verskillende gerangskikte etiket-subversamelings. Vir elke boom word verskillende etikette by verskillende vlakke van die boom gebruik, soos gebaseer op die toepaslike etiket-rangskikking van die boom. Die boom-vlakke bestaan uit nodusse wat verdeel word deur gebruik te maak van enige binêre-klassifiseerder. Aangesien 'n boom-vlak afhanklik is van sy etiket, sowel as vorige verdelings wat gemaak was toe voorouer-nodusse gevorm het deur van ander etikette gebruik te maak, word hoër-orde etiket-korrelasies implisiet op 'n eenvoudige manier in die model geïnkorporeer. Afhangende of ewekansige of voorafbepaalde etiket-rangskikkings gebruik word om die ensemble te pas, word óf Ewekansige LDsplit óf Voorwaardelike LDsplit gepas. 'n Omvangryke empiriese studie word uitgevoer op 'n reeks standaard multi-etiket datastelle. Die empiriese resultate dui daarop dat, ten spyte van die eenvoudige raamwerk, beide Ewekansige LDsplit en Voorwaardelike LDsplit

baie kompeterende klassifikasie-prestasie bied in vergelyking met bestaande multi-etiket klassifikasie-metodes.

Vir multi-etiket data is ‘n inset-veranderlike globaal belangrik as dit belangrik geag word vir verskeie of alle etikette. Dog kan ‘n inset-veranderlike ook lokaal belangrik geag word vir ‘n spesifieke etiket. Min strategieë vir inset-veranderlike rangskikking binne multi-etiket data beskou beide globale en lokale belangrikheid van inset-veranderlikes. Bowendien, vir die bestaande metodes word die benutting van etiket-afhanklikheid binne die data meestal uitgelaat. Derhalwe word verskillende maniere uiteengesit hoe ‘n LDsplit ensemble globale en lokale inset-veranderlike rangskikkings kan genereer en sodoende beter interpretasie van die data toelaat. Resultate verkry op grond van sintetiese gegenereerde multi-etiket datastelle wys dat beide die nuwe globale en lokale belangrikheids-maatstawwe goed presteer.

Sleutelwoorde: multi-etiket klassifikasie, ensemble metode, boom, etiket-orde, etiket-korrelasie, plaaslike etiket-korrelasie, inset-veranderlike rangskikking, globale belangrikheid, lokale belangrikheid.

Acknowledgements

Firstly, I wish to thank the National Research Foundation (NRF) for their financial assistance towards my PhD. I am also thankful for the Department of Statistics and Actuarial Science of Stellenbosch University for providing me with the necessary support to complete my research.

A heartfelt thank you to my supervisors, Prof S.J. Steel and Dr T. Sandrock. I would not have been able to complete this dissertation without their guidance, assistance, feedback, insight, and time. I would like to express a special thank you to Prof Steel for his support and encouragement throughout my Honours, Masters, and now PhD. It has been an immense honour and inspiration to work with someone with his kindness and vast knowledge.

I am extremely grateful to have finished my dissertation on our beautiful farm. Thank you to my loving parents who crafted it to be my favourite place on earth. Delicious dinners and insightful conversations around the kitchen table with my father, mother, brother and sister, kept me going day after day.

Finally, I would like to thank Ruan for his endless support throughout my research, for always believing in me, and for being the best sounding board and pep talker one can ask for. He also earns a bonus “thank you” for promising to read my entire dissertation from start to finish.

“Something different bloomed – writing in my room” – T.A. Swift

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BCC	Bayesian Chain Classifiers
BN	Bayesian Network
BPM	Beats Per Minute
BP-MLL	BackPropagation for Multi-Label Learning
BR	Binary Relevance
BR_RF	Binary Relevance with a random forest as base classifier
CART	Classification And Regression Tree
CBMLC	Clustering-Based Multi-Label Classification
CC	Classifier Chains
CC-DP	Dynamic Programming based Classifier Chain
C-LDsplit	Conditional Label-Dependent splitting (as defined in Section 3.3.3)
C-LDsplit_SVM	Conditional Label-Dependent splitting with a support vector machine as base classifier
C-LDsplit_Tree	Conditional Label-Dependent splitting with decision tree base classifier
CLR	Calibrated Label Ranking
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
EBR	Ensemble of Binary Relevance
ECC	Ensemble of Classifier Chains

ECC_SVM	Ensemble of Classifier Chains with support vector machine as base classifier
ECC_Tree	Ensemble of Classifier Chains with a decision tree as base classifier
FN	False Negative
FP	False Positive
FPR	False Positive Rate
HDDT	Hellinger Distance Decision Tree
HOMER	Hierarchy Of Multi-label learnERs
IB	In-Bag
IG	Information Gain
IR	Imbalance Ratio
kNN	$k-$Nearest Neighbour
LaCova	Tree-based multi-label classifier using label covariance as splitting criterion
LDA	Linear Discriminant Analysis
LDsplit	Label-Dependent splitting (the novel approach presented in Chapter 3)
LEAD	Learning by Exploiting lAbel Dependency
LOC	Local Correlation
LP	Label Powerset
MDA	Mean Decrease Accuracy
MDI	Mean Decrease Impurity
MFCC	Mel Frequency Cepstral Coefficient

ML-C4.5	Multi-Label C4.5
ML-Forest	Multi-Label Forest
ML-kNN	$k-$Nearest Neighbour algorithm for multi-label learning
MLPP	Multi-Label Pairwise Perceptron
ML-SVMDT	Hybrid multi-label decision tree utilising local support vector machines
MODT	Multi-Objective Decision Tree
NLP	Natural Language Processing
OOB	Out-Of-Bag
OOCC	One-to-One Classifier Chains
PCT	Predictive Clustering Tree
PPT	Pruned Problem Transformation
QWeighted	Quick Weighted Voting algorithm
QWML	Quick Weighted Voting algorithm for multi-label learning
RAkEL	Random $k-$labelsets
RAkEL$_d$	Disjoint Random $k-$labelsets
RAkEL$_o$	Overlapping Random $k-$labelsets
RFML-C4.5	Random Forests of Multi-Label C4.5
RF-PCT	Random Forests of Predictive Clustering Trees
R-LDsplit	Random Label-Dependent splitting (as defined in Section 3.2)
R-LDsplit_SVM	Random Label-Dependent splitting with a support vector machine base classifier

R-LDsplit_Tree	Random Label-Dependent splitting with a decision tree base classifier
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
RPC	Ranking by Pairwise Comparison
SCUMBLE	Score of ConcUrrence among iMBalanced LabEls
SOSHF	Sparse Oblique Structured Hellinger Forest
STFT	Short-Term Fourier Transform
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TREMLC	Triple-Random Ensemble Multi-Label Classification
XMLC	Extreme Multi-Label Classification

Chapter 1: Introduction

1.1 Classification

Today the modern term “machine learning” comes up in conversations held in laboratories, boardrooms, factories, universities, classrooms, and everyday kitchens. However, looking back as recently as 15 years ago, this was not the case. A simple Google Trend search of the term “machine learning” depicts the explosive rate at which worldwide interest in machine learning has grown in recent years (Figure 1.1).

The term “machine learning” was coined in 1959 by Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence (Samuel, 1959). However, learning from data, *i.e.* statistical learning, has been the job of statisticians for decades (Hastie *et al.*, 2009:xi). In broad terms machine learning can be defined as the process of solving a practical problem by gathering a dataset and algorithmically building or training a statistical model based on that dataset (Burkov, 2019). Machine learning can be divided into four types of learning: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. This dissertation falls within the sphere of supervised learning. More specifically, the supervised learning task of multi-label classification is considered.

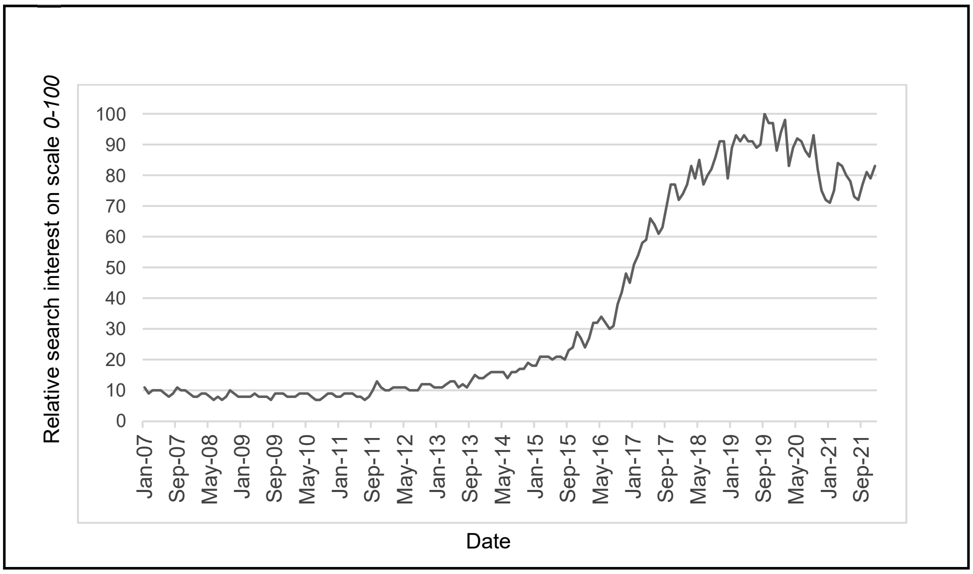


Figure 1.1 Google Trend search results for the term “machine learning”

Supervised learning is one of the most widely researched and investigated areas of machine learning (Levatić *et al.*, 2015). Suppose a dataset consists of a collection of N data observations and each data observation, $\mathbf{x}_i$, $i = 1, 2, ..., N$, is represented by a set of p input variables, $X_1, X_2, ..., X_p$. In some scenarios these input variables may influence one or more output or response variables, $Y_1, Y_2, ..., Y_K$. In that case the collection of input and output pairs, $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, ..., N\}$, may be used to solve a supervised learning problem. Supervised learning refers to the machine learning task of learning or training a model, $f(X_1, X_2, ..., X_p)$, from this data that relates the input variables to the response(s). The estimated function, $\hat{f}(X_1, X_2, ..., X_p)$, is used to predict responses for data cases which are represented by only the input variables (Hastie *et al.*, 2009:9).

Depending on how the response variables are defined, supervised learning problems can be subdivided into classification and regression problems. Regression problems have numeric response variables, *i.e.* $Y_1, Y_2, ..., Y_K$ are quantitative. However, if the response variables are specified in terms of classes, *i.e.* $Y_1, Y_2, ..., Y_K$ are qualitative, a classification task arises (Hastie *et al.*, 2009:10). For a classification task the response variables, $Y_1, Y_2, ..., Y_K$, can be referred to as labels, where K denotes the total number of labels.

Traditional single-label classification tasks, such as binary and multi-class classification, have $K = 1$. For binary classification, each data observation is associated with one of two disjoint classes. The collection of input and output pairs is denoted by $\{(\mathbf{x}_i, y_i), i = 1, 2, ..., N\}$, with $Y \in \{0, 1\}$. If $\mathbf{x}_i$ is included in the first class, $y_i = 1$, else $y_i = 0$. Many classification problems can take this form. For example, a challenging aspect in breast cancer research is to classify tumours either as malignant (cancerous) or benign (non-cancerous) (Wolberg *et al.*, 1992). Multi-class classification extends binary classification to settings where each data observation is associated with one of $G > 2$ disjoint classes. In this case the collection of input and output pairs is denoted by $\left\{\left(\mathbf{x}_i, y_i\right), i=1,2,...,N\right\}$, with $Y \in \left\{1,2,...,G\right\}$. If $\mathbf{x}_i$ is included in the g^{th} class, $y_i = g$, where $g = 1,2,...,G$. The Oxford-IIIT-Pet dataset is an example of a multi-class classification dataset (Parkhi *et al.*, 2012). The dataset consists of a collection of cat images where each image is appropriately classified according to $G = 12$ cat breeds such as American Shorthair, Maine Coon, Egyptian Mau and Ragdoll.

The wide range of research on single-label data has led to a variety of learning algorithms proposed for single-label classification. Some of the most well-known algorithms are Linear Discriminant Analysis (LDA), logistic regression, decision trees, Support Vector Machines (SVMs), $k-$Nearest Neighbour (kNN) classifiers, neural networks, and boosting methods such as AdaBoost.

In traditional single-label classification each data case is associated with one class. However, there are many scenarios where several labels may be associated simultaneously with a data case. For example, in video annotation it might be that the aim is to assign video tags to videos. A video of a lioness stalking a herd of wildebeest in the Serengeti may require multiple tags such as *Lion*, *Wildebeest*, *Africa* and *Grassland*. Multi-label classification is an extension of single-label classification to scenarios such as this.

Multi-label classification has $K > 1$ so that multiple non-disjoint labels exist, $Y_1, Y_2, ..., Y_K$. In this case the collection of input and output pairs is denoted by $\left\{\left(\mathbf{x}_i, \mathbf{y}_i\right), i=1,2,...,N\right\}$, with $Y_k \in \left\{0,1\right\}$, $k = 1,...,K$. Each $\mathbf{x}_i$, $i = 1,...,N$, is annotated with multiple or none of the K labels by setting the k^{th} entry of $\mathbf{y}_i$ either equal to 1 (if $\mathbf{x}_i$ has Y_k present) or equal to 0 (if Y_k is absent for $\mathbf{x}_i$). In this way each data observation is associated with a set of relevant and irrelevant labels. Therefore, the task of multi-label classification is to learn a function which can predict the set of labels for unseen data observations that are represented by only the input variables (Zhang and Zhou, 2013).

A related task to multi-label classification is that of label ranking. Label ranking requires an ordered set of labels ranked from most to least relevant for a given observation (Tsoumakas and Katakis, 2007). For example, in a news-filtering application users can be presented with a ranking of important to less important news articles depending on the terms they search for when using the application (Tsoumakas *et al*., 2010). Models trained on a multi-label dataset that provide a query data case with both a bipartition of the labels into relevant and irrelevant sets, as well as an ordering of the labels from most to least important, are referred to as multi-label ranking methods (Madjarov *et al*., 2012). Although the novel multi-label classification method proposed in Chapter 3 can provide a ranking of the labels of a multi-label dataset, the main focus of this dissertation is multi-label classification.

In certain classification problems the classes may have a hierarchical structure. In this case each label is composed of multiple class-levels. The top class is the most general and is subdivided into more specific classes. The hierarchical structure that formalises the relationship among classes can assume the form of a tree or of a Directed Acyclic Graph (DAG) (Ramírez-Corona *et al*., 2014). A data case that belongs to a certain class automatically belongs to all its so-called super-classes. This is referred to as the hierarchy constraint. Hierarchical multi-label classification describes the scenario where a data case can be associated with several different paths of the class hierarchy (Wehrmann *et al*., 2018). Ren *et al.* (2014) give an example of a hierarchical multi-label classification problem of social text streams. This dissertation however focuses on non-hierarchical multi-label classification (sometimes also referred to as flat multi-label classification).

1.2 Notation

In this section, multi-label classification is described in more detail by summarising the notation used throughout this dissertation.

Denote by X a $p-$dimensional input space of input variables $X_1, X_2, ..., X_p$ and by Y a $K-$dimensional output space of labels $Y_1, Y_2, ..., Y_K$. The collection of K labels is denoted by

L. In this dissertation, a multi-label training dataset containing N input and output pairs is denoted by $\{(\mathbf{x}_i, \mathbf{y}_i),\ i = 1, 2, \ldots, N\}$. Therefore the i^{th} multi-label example pair is denoted by $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i = \left[x_{i,1},\ x_{i,2},\ \ldots,\ x_{i,p}\right]$ is a row-vector of the values of the p input variables, $X_1, X_2, \ldots, X_p$, and $\mathbf{y}_i = \left[y_{i,1},\ y_{i,2},\ \ldots,\ y_{i,K}\right]$ is a row-vector of the values of the K labels, $Y_k \in \{0,1\}$, $k = 1, \ldots, K$. If $\mathbf{x}_i$ has Y_k present, $y_{i,k} = 1$, otherwise $y_{i,k} = 0$.

The multi-label data can also be summarised using matrices, $\left[\underset{N\times p}{\mathbf{X}}\ \underset{N\times K}{\mathbf{Y}}\right]$. Here $\mathbf{X}$ represents the matrix of N data cases, $\mathbf{x}_i$, $i = 1, \ldots, N$, given as rows in $\mathbf{X}$. Similarly $\mathbf{Y}$ represents the matrix of label classifications with each $\mathbf{y}_i$, $i = 1, \ldots, N$, given as a row in $\mathbf{Y}$.

In other words:

$$\left[\underset{N\times p}{\mathbf{X}}\ \underset{N\times K}{\mathbf{Y}}\right] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{y}_1 \\ \mathbf{x}_2 & \mathbf{y}_2 \\ \vdots & \vdots \\ \mathbf{x}_N & \mathbf{y}_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} & y_{1,1} & y_{1,2} & \cdots & y_{1,K} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} & y_{2,1} & y_{2,2} & \cdots & y_{2,K} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,p} & y_{N,1} & y_{N,2} & \cdots & y_{N,K} \end{bmatrix}.$$

Furthermore, in this dissertation, each unique set of labels that exists within the multi-label dataset is referred to as a labelset. As an example, consider the simple multi-label dataset (consisting of $N = 5$ data observations and $K = 4$ labels) summarised in the left panel of Figure 1.2. The right panel of Figure 1.2 gives all the unique labelsets present in this dataset.

Data cases	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}_1$	0	1	1	0
$\mathbf{x}_2$	1	0	0	0
$\mathbf{x}_3$	0	1	1	1
$\mathbf{x}_4$	1	0	1	0
$\mathbf{x}_5$	1	0	0	0

Unique labelsets	
0 1 1 0	*i.e.* Y_2 and Y_3 present
1 0 0 0	*i.e.* Y_1 present
0 1 1 1	*i.e.* Y_2, Y_3 and Y_4 present
1 0 1 0	*i.e.* Y_1 and Y_3 present

Figure 1.2 Unique labelsets present in simple multi-label dataset

The goal of multi-label classification is to construct a predictive model $f: \mathrm{X} \rightarrow \mathrm{Y}$ that provides an unseen data case, $\mathbf{x}_{new} \in \mathrm{X}$, with a vector of label classifications, $\mathbf{y}_{new} \in \mathrm{Y} = \{0,1\}^K$. In some cases the estimated function, $\hat{f}(.)$, provides $\mathbf{x}_{new}$ with a degree of confidence for each of the labels, $Y_1, Y_2, ..., Y_K$. In this case $\mathbf{y}_{new}$ represents a K–dimensional vector of confidence values in the range $[0,1]$. A threshold (either a global threshold or a threshold value per label) can be used to convert $\mathbf{y}_{new}$ to a K–dimensional $0/1$ vector. For example, if t_{Y_k} denotes the threshold applied for Y_k, y_k is transformed to 1 if $y_k \geq t_{Y_k}$, otherwise y_k is transformed to 0 if $y_k < t_{Y_k}$.

1.3 The upsurge in multi-label data

Early research in multi-label classification mainly considered multi-label problems within the text domain. Examples include McCallum (1999), Schapire and Singer (2000) and Ueda and Saito (2002). In the multi-label text domain, each data case represents a text document and each document is assigned multiple labels. For example, a news-filtering problem may assign multiple topics such as *Finance*, *Sport* and *Entertainment* to news articles. An article about a famous sportsperson starring in an upcoming film may be tagged with both the labels *Sport* and *Entertainment*. Multi-label classification within the text domain is therefore a specific Natural Language Processing (NLP) problem.

The upsurge in online content has resulted in an upsurge in multi-label text datasets in recent years. As a simple example, current online archives of scientific papers contain thousands of papers each covering a range of topics. A multi-label text dataset resembling this form is available on the online data-sharing platform Kaggle[1]. The dataset consists of a collection of abstracts and paper titles that are each labelled with six possible topics namely *Computer Science*, *Physics*, *Mathematics*, *Statistics*, *Quantitative Biology* and *Quantitative Finance* (Kaggle: NLP on research articles, 2020). More recent examples of multi-label problems within

[1] https://www.kaggle.com/ (accessed 10 October 2022)

the text domain appear in Katakis *et al*. (2008), Loza Mencía and Fürnkranz (2008), Sajnani *et al*. (2013) and Rivolli *et al*. (2017).

The range of multi-label classification domains has expanded in the last two decades due to the dramatic increase in the number of available multi-label datasets and the diversity of the corresponding multi-label problems. Apart from the text domain, multi-label datasets are found in a variety of domains such as image annotation, bioacoustics, biology, music research and medical diagnostics.

In image annotation each data case represents an image that is tagged with multiple labels. Well-known research in multi-label image annotation include that of Duygulu *et al*. (2002), Boutell *et al*. (2004) and Qi *et al.* (2008). A typical multi-label image annotation example considers photographs of worldly scenes that are to be labelled with possible tags such as *City*, *Sea*, *Grass*, *Bridge*, *Person*, *Dog*, and *Athlete*. However, image annotation is not limited to this framework. An interesting multi-label image classification dataset is given by Chu and Guo (2017). The dataset consists of a collection of film posters where each poster is labelled with possibly more than one genre such as *Action*, *Adventure*, *Animation* and *Comedy*.

Research in multi-label bioacoustics is presented by Briggs *et al*. (2013). Their work considers an audio dataset where the data observations represent audio recordings of bird species. In each recording, multiple bird species may appear. Recently, a competition on the data-sharing platform Kaggle was presented by Rainforest Connection[2]. Rainforest Connection is an acoustic monitoring platform built to protect and study remote ecosystems. In the multi-label competition dataset, observations represent audio files that include sounds from numerous animal species. The challenge is to accurately tag real-time audio clips with the appropriate species present in the clips. In this case, multi-label learning models with high predictive performance could enable earlier detection of human environmental impacts and help make

[2] https://rfcx.org/ (accessed 10 October 2022)

environmental conservation swifter and more effective (Kaggle: Rainforest Connection Species Audio Detection, 2020).

Well-known work in the multi-label biology domain is that of Elisseeff and Weston (2001). In their work a multi-label dataset is considered where each observation is a gene associated with possibly 14 biological functional groups. Other examples of multi-label research in the biology domain are Lin and Xu (2016) and Zhou *et al*. (2017).

Examples of multi-label music research include instrument recognition (Spyromitros-Xioufis *et al*., 2011), tagging music pieces with appropriate music genres (Sanden and Zhang, 2011), and classification of music pieces into the emotions they provoke (Trohidis *et al*. 2008).

In medical diagnostics patients may be suffering from multiple diseases or disorders at once so that multi-label datasets are formed (Shao *et al.* 2013). Pestian *et al*. (2007) give an example of a medical dataset where each observation is a description of a patient's symptoms history, and each patient is tagged with their diagnosed diseases from a collection of 45 possible diseases. Ratnarajah and Qiu (2014) give another example of multi-label classification research in the medical domain. In their work multiple anatomical labels of brain white matter bundles can be assigned to a voxel. Here accurate classification can result in better understanding of brain development and can help detect white matter abnormalities.

The above paragraphs give a few examples of the vast number of multi-label datasets available today. Note that popular multi-label benchmark datasets and multi-label benchmark data repositories are discussed in detail in Section 2.3.

With the upsurge in multi-label applications, multi-label research has become increasingly important. Consequently, the increase in the number of available multi-label datasets has coincided with an increase in multi-label research. Figure 1.3 illustrates this by summarising the number of papers with the keyword "multi-label classification" over the past 15 years available in the SCOPUS database[3]. Despite the almost exponential curve of growth depicted in Figure 1.3, there still remain some shortcomings in the multi-label literature. This is a consequence of the challenges faced when analysing multi-label data, as discussed in the next section.

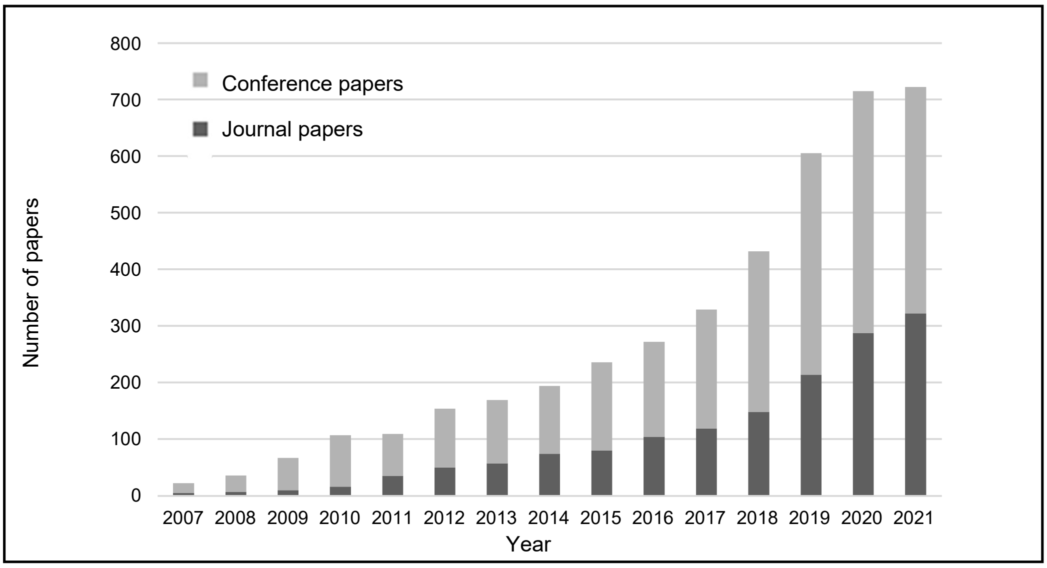


Figure 1.3 Increase in multi-label classification research on the SCOPUS database

1.4 Challenging aspects of multi-label data

Compared to traditional single-label data, multi-label data possess unique characteristics. These aspects result in additional challenges when analysing multi-label data. This dissertation considers several of these challenging aspects; however, two areas receive significant attention. The first is the consideration of label correlation to aid in achieving accurate multi-label classification of new unseen data cases. The second is the aspect of variable importance for inference and variable selection in multi-label data. The following sections give brief descriptions of both challenges as well as others faced when analysing multi-label data.

[3] http://www.scopus.com/ (accessed 10 October 2022)

1.4.1 Label correlations

It is reasonable to assume that most multi-label datasets have dependencies amongst the labels. This implies that information from one label may assist in uncovering information from another related label. For example, in video annotation, if a video has the labels *Wildebeest* and *Grassland* present, the probability of also having the label *Africa* present may be higher than that of another label such as *Urban*. Therefore, research widely considers the incorporation of label correlation into the multi-label learning method as a fundamental element, since the possible extra information that emerges could be very beneficial for classification performance (Huang and Zhou, 2012).

Although many papers have recently emerged where authors claim that their proposed multi-label learning methods exploit label correlations (see Section 2.7.), it remains a very difficult task to do so effectively. One challenge is for example that the label correlations present in the data (if any) are unknown in the case of non-hierarchical labels. Therefore, in most cases the label correlations are learned from the data instead of relying on external knowledge sources to specify the label correlations. Furthermore, it is difficult to know if an average improvement achieved by a multi-label classifier which exploits label correlation is because of the incorporation of label correlations or because of other factors instead. Another challenge is that the unknown label correlation structure may be complicated. For example, the labels *Africa* and *Urban* may be negatively correlated in videos of animals in the Serengeti, but a video showcasing tourist activities in The City of Cape Town may have both labels *Africa* and *Urban* present. This pertains to the belief that label correlations are not necessarily shared globally by all data cases (Huang and Zhou, 2012).

Figure 1.4 gives a simple image annotation example. As shown in the left panel of Figure 1.4, labels such as *Grass* and *Tree* may often occur together for many images. It could therefore be beneficial to exploit this co-occurrence relationship between *Grass* and *Tree*. For example, in images where trees are perhaps more prominent than grass, such as images (b) and (c), models that incorporate label relationships may still detect the *Grass* label. However, this co-occurrence between *Grass* and *Tree* is not shared globally by all observations, as illustrated in the right panel of Figure 1.4. As a result, if a model assumes that a global positive correlation exists between *Grass* and Tree, performance can weaken. The fact that the *Tree* label is present for the images in the right panel of Figure 1.4 could be misleading when deciding whether or not *Grass* is present in these images. Different label correlation structures exist within subgroups of the observations.

In existing work, however, multi-label learning methods mostly focus on global label correlation exploitation (Zhu *et al.*, 2017). This presents a possible limitation of existing work.

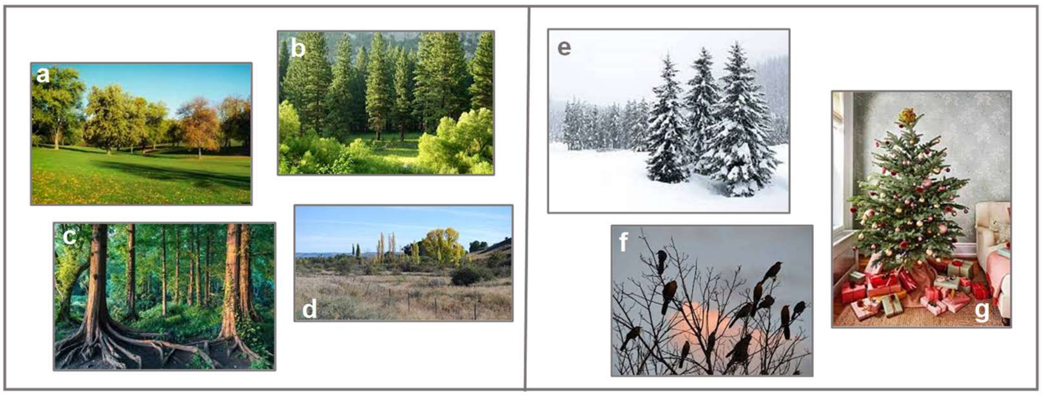


Figure 1.4 Simple image annotation example to illustrate non-global label correlations

1.4.2 Variable importance and variable selection

Accurate prediction of unseen data cases is not the only desirable property of a machine learning model. Many data mining applications require interpretable results instead of only predicting the response by means of some "black-box" model. This is true for single-label classification as well as multi-label classification. One way in which a model can be more interpretable is if the relative importance of input variables for classification can be determined. In this way the most important input variables for classification can be identified, which can lead to valuable insight concerning the classification problem at hand. The medical domain is one example where such insight could be especially important. Better understanding of input variables can lead to successful early interventions that may prevent or delay diseases such as Alzheimer's disease (Cheng *et al.*, 2019).

As it has become easier to collect and store data, many modern datasets suffer from “the curse of dimensionality”. This describes the phenomenon that an increase in the number of input variables can add more valuable information; however, the large set of input variables may also include more redundant and irrelevant variables as well. In some scenarios so-called wide datasets form where the number of input variables exceeds the number of observations. However, if the number of observations does not increase with the dimensionality of the dataset, predictive performance of models can become unreliable. Hastie *et al.* (2009:57) state that usually only a fraction of the input variables is important for classification. Other variables may be irrelevant and such variables do not provide useful information for the classifier. Some input variables may even hinder the learning process. Therefore, not only can a classifier be learned faster, but predictive performance can also sometimes improve when a classifier is learned on only a subset of the available input variables (Hastie *et al.*, 2009:57). The aim of variable selection is to effectively reduce the dimensionality of the input space by removing irrelevant and redundant input variables. Variable selection may for example be useful in the text domain where data are represented in the bag-of-words framework. In this case the set of unique words found throughout the dataset forms the “bag-of-words” of the dataset. For each text observation it is noted how many times each of these words occur. Since the size of the bag-of-words is usually large, variable selection could be useful in such scenarios to eliminate uninformative words (Sun *et al.*, 2013).

Note that the two above-mentioned concepts of variable importance and variable selection are related in supervised learning. Variable importance measures can provide information regarding the relationship between the input variables and the response of a dataset. However, since it is desirable for a variable selection strategy to keep important variables and eliminate irrelevant variables, variable importance measures can also be used to perform variable selection. In this case only the most important variables (according to an importance measure) are selected.

For single-label classification, an important input variable can effectively discriminate between the disjoint classes. However, additional complexity arises for variable importance within multi-label data. For example, a variable may be good at discriminating between values of one label, but at the same time may provide little or no insight in discriminating between the values of other labels. Due to this, one can identify two forms of variable importance within multi-label data. Firstly, a variable can be declared locally important for a label if it is deemed important for the specific label, irrespective of its importance for any of the other labels. Secondly a variable can be declared globally important if it is deemed important for several or all of the labels in the dataset.

Multi-label variable selection is more difficult than single-label variable selection since the interactions of the data are not limited to the set of input variables and a single label. Because the variables can be globally and locally important, the question arises if a variable should only be selected if it is globally important for a large subset of labels, or should other locally important variables be included as well?

Many proposals exist to measure variable importance and perform variable selection for single-label data. In comparison, the body of work on variable importance and variable selection for multi-label data is smaller (Spolaôr *et al.*, 2013). As a result of the increase in the popularity of multi-label classification in recent years, more research in these areas have started to emerge (Pereira *et al.*, 2018). Limited proposals are however found that consider both global and local importance of input variables. Furthermore, many of the available methods transform the multi-label data in such a way that single-label variable importance and selection methods can be applied (Kocev *et al.*, 2013). This strategy does not fully exploit the possible label dependencies in the multi-label data.

1.4.3 Other challenging aspects of multi-label data

- **Label imbalance**

Many real-word multi-label scenarios give rise to imbalanced datasets. The number of instances associated with each label is often notably dissimilar and the label space is generally characterised as sparse. This imbalance can influence the classification performance of a classifier. The class-imbalance problem has been more extensively studied within single-label classification scenarios. Imbalance within multi-label datasets has only recently received more attention.

Due to the more complex structure of multi-label data (compared to single-label data), it is important to distinguish between different types of imbalance that may exist in a multi-label dataset. Daniels and Metaxas (2017) distinguish between imbalance *between* labels and imbalance *within* labels. Note that between-label imbalance considers the imbalance that exists when different labels are compared (e.g. *Label 1* appears seven times more than *Label 2*), whereas within-label imbalance considers how imbalanced individual labels are (e.g. *Label 1* is marked absent for far more of the observations than for which it is marked present).

Charte *et al.* (2015) refer to three main approaches used in the literature to address the imbalance problem for multi-label data: data resampling, algorithmic adaptations and cost sensitive classification. Data resampling aims to rebalance the class distribution and may therefore involve addition of observations belonging to the infrequent class (oversampling) or exclusion of observations belonging to the frequent class (undersampling). Since this is a pre-processing approach, these methods may be used along with any classification algorithm. In contrast, algorithmic adaptations are classifier dependent since these approaches aim to adapt the classification algorithm to handle the class imbalance. Finally, cost sensitive classification is a combination of the two previous approaches and incorporates misclassification costs for labels.

- **Synthetic data generation and benchmark data**

Properties of single-label methods are often systematically investigated on synthetic single-label data. However, very little has been published regarding the generation of synthetic multi-label data in comparison. This might be because multi-label data generation is more challenging. Ideally a multivariate distribution from which to generate the input variables should be specified, and similarly, a multivariate distribution with a correlation structure should be specified from which to generate the label variables. However, one should also be able to specify a dependence structure between the input variable space and label space. This may be less straightforward if users desire control over the specification of global and local variable importance. Some approaches for synthetic multi-label data generation are presented in Read *et al.* (2012), Luaces *et al.* (2012) and Tomás *et al.* (2014). Although these approaches offer users control over certain properties of the generated synthetic data, each approach possesses some limitations. For example, *Mldatagen* (Tomás *et al.*, 2014) does not allow for data cases that have all labels absent, offers no option for specifying a multivariate distribution (for example multivariate normal) for the input variables, allows no direct control over the label densities, and no way of controlling correlations amongst the label variables. Furthermore, no distinction is made between globally and locally important input variables.

Since it is difficult to generate synthetic multi-label data, it has become common practice in multi-label literature to make use of standard publicly available multi-label benchmark datasets to investigate or compare properties of multi-label methods. These benchmark datasets cover a range of domains and vary in size as well as other measurable properties as described in Section 2.2. Many online repositories exist to provide easy access to these benchmark datasets for researchers. Some examples of common benchmark datasets and repositories are given in Section 2.3.

- **Number of labels**

The output space of multi-label datasets sometimes consists of many labels. Examples of multi-label benchmark datasets with many labels are *Corel5k* (Duygulu *et al*., 2002), *Delicious* (Tsoumakas *et al*., 2008), and the three-part collection of text documents about European Union law, *EUR-Lex*, consisting of *EUR-Lex-dc*, *EUR-Lex-ev* and *EUR-Lex-sm* (Loza Mencía and Fürnkranz, 2008). These datasets have 374, 983, 412, 3993 and 201 labels respectively.

Large values of K increase the computational complexity of the multi-label problem. In most cases the time and computer memory needed to train a classifier, and to classify an unseen data case, increases with an increase in K. This could be a serious concern if applications require fast/efficient computation. Furthermore, the multi-label problem also becomes more imbalanced since normally only a few observations are annotated with each of the many labels. Consequently, the predictive performance of models may suffer when K is large (Bogatinovski *et al*., 2022). It is therefore an advantage if a multi-label learning method can scale to settings with large K.

Many modern multi-label image and text datasets are characterised by extremely large label collections. For example, Wikipedia has over a million curator-generated category labels of which only a handful are usually applicable per article (Liu *et al*., 2017). This development has resulted in a new subfield of multi-label classification, namely extreme multi-label classification (XMLC). XMLC refers to the problem of assigning relevant labels to observations from a label collection consisting of hundreds of thousands or millions of labels (Liu *et al*., 2021). XMLC falls beyond the scope of this dissertation. When K is referred to as "large" in this dissertation, this implies that the number of labels is similar to that of the *Corel5k*, *Delicious* and *EUR-Lex* datasets.

- **Missing labels**

Certain multi-label applications may require human annotators to annotate appropriate labels to each of the data observations of a multi-label dataset. Since annotating labels is time consuming, expensive and requires expertise, in some real-world applications labels are only partially observed (Wang *et al.*, 2014). Multi-label learning with limited supervision has attracted more attention recently. Different assumptions concerning the structure of the multi-label data exist in this setting. For example, the label collection could be considered fixed, so that the classifier only learns to complete the missing entries of certain labels. In other scenarios one might assume that there exists an entire collection of missing labels so that the classifier is trained for unseen data. Multi-label learning with missing labels is however not studied in this dissertation. Liu *et al.* (2021) provide a detailed discussion of this topic.

1.5 Overview of dissertation

In this introductory chapter the goal of multi-label classification is defined. Emphasis is also placed on the importance of multi-label research due to the increase in multi-label datasets and the diversity of modern multi-label applications. Compared to single-label data, multi-label data possesses unique characteristics. Despite the increase in multi-label research, there are still some shortcomings within the literature, some of which this dissertation aims to address. The first is that few models aim to exploit local label correlation to aid in achieving accurate multi-label classification. The second is the lack of global and local importance measures for input variables to allow for better understanding of the multi-label data. The few available measures of variable importance mostly neglect to exploit label dependencies.

In the next chapter an in-depth overview of multi-label learning is given. This includes definitions of various multi-label descriptive properties and evaluation measures, summaries of popular multi-label benchmark datasets, references to popular software for multi-label learning, a comprehensive discussion of label correlation, and descriptions of multi-label learning methods related to the novel multi-label learning method proposed in this dissertation.

In Chapter 3 a new tree-based ensemble method for multi-label classification is proposed, namely Label-Dependent splitting (LDsplit). With each tree-structure in the LDsplit ensemble, the aim is to split the data in a label-dependent way so that local label correlations are implicitly exploited. For each tree, different labels are used at different levels of the tree, as determined by the label order applicable to that tree. A detailed description of the fitting and classification procedures of two label ordering strategies are given, namely Random LDsplit and Conditional LDsplit. The chapter concludes with a discussion of the distinctive and favourable properties of LDsplit compared to related methods.

Chapter 4 presents an extensive empirical study performed on standard publicly available multi-label benchmark datasets. The study investigates the properties of the newly proposed Random and Conditional LDsplit models and compares the predictive performance of the models to that of other well-known and related multi-label learning methods. The study provides proof that LDsplit is competitive with state-of-the-art multi-label learning methods in terms of predictive performance.

In Chapter 5 the discussion of LDsplit is extended beyond its satisfying predictive performance by considering aspects of variable importance and variable selection. First an overview of traditional variable importance measures for single-label trees is presented as well as discussions of approaches for variable importance and variable selection in multi-label data. Hereafter variable importance measures of single-label trees are extended to multi-label data by using the LDsplit framework. Novel LDsplit measures for both global and local importance of variables are proposed. Since the proposed variable importance measures are influenced by label correlations, the shortcoming identified in the multi-label literature in this regard is addressed. The chapter concludes with a discussion of the advantages of the proposed variable importance measures, which includes the fact that the input variable rankings can be used to perform variable selection.

Chapter 6 presents empirical properties and applications of the LDsplit variable importance measures proposed in Chapter 5. Firstly, multi-label synthetic data are generated by using the algorithm proposed by Sandrock and Steel (2017). This algorithm is particularly useful since it has the option of specifying locally and globally relevant input variables. The proposed LDsplit global and local variable importance measures are therefore effectively assessed in this chapter. To investigate the performance of the measures in different settings, several synthetic multi-label datasets are generated considering a range of configurations determined by label densities, label correlation, input variable correlation and the strength of the signal of the data. Satisfactory results confirm the ability of the LDsplit model to produce interpretable results regarding global and local importance of input variables. Chapter 6 concludes with a short benchmark dataset application of the proposed LDsplit variable importance measures.

Finally, concluding remarks and opportunities for future research are presented in Chapter 7.

Chapter 2: Multi-label learning

2.1 Introduction

The aim of this chapter is to present an overview of the literature of multi-label learning and data, and to provide a summary of the current resources available. Note that a reader versed in the multi-label literature can bypass this chapter.

The chapter includes popular measures of descriptive properties of multi-label data, a summary of well-known publicly available multi-label benchmark datasets, definitions of multi-label evaluation measures and a summary of currently used multi-label software.

Hereafter attention is directed to the important issue of label correlation. A detailed discussion is given on label correlation and how this concept is currently viewed in the literature and incorporated within multi-label learning methods. This is followed by descriptions of fitting and classification procedures of different learning methods related to the novel approach proposed in Chapter 3, namely LDsplit.

2.2 Describing multi-label datasets

A multi-label dataset can be characterised by various properties including the number of observations, input variables, labels and unique labelsets, as well as the average number of relevant labels per observation and the label imbalance in the dataset. Therefore, when describing a multi-label dataset, it is important to refer to such descriptive properties of the dataset. Standard definitions exist in the multi-label literature to measure descriptive properties of multi-label data. This section provides a summary of the most well-known measures, also referred to as meta-features (Kostovska *et al.*, 2022).

Kostovska *et al.* (2022) separate the measurable properties of a multi-label dataset into three major groups: dataset-specific properties, attribute-specific properties, and label-specific properties.

Dataset-specific properties describe the multi-label dataset using general statistics. This includes the number of observations, N, the number of labels, K, the number of labelsets, the number of input variables, p, as well as the different types of input variables (qualitative or numeric) and various ratios between these quantities (Kostovska *et al.*, 2022). Attribute-specific properties provide a detailed insight into the properties of the input variables. This for example includes the average absolute correlation between numeric variables or the mean of kurtosis between numeric variables. Finally, label-specific properties aim to describe the label space of a multi-label dataset. These label-specific properties were specifically developed for multi-label data, since it is exactly the label space structure of multi-label data which distinguishes multi-label data from single-label data.

One such label-specific property is the bound which represents the maximum number of labelsets that may exist in the dataset. This bound is 2^K (if it is assumed that some data observations may have all labels marked absent). Diversity on the other hand represents the percentage of labelsets present in the dataset considering the total possible number of labelsets. Another label-specific property is referred to as label cardinality, which gives the average number of relevant labels per observation (Tsoumakas *et al.*, 2010). Label cardinality is therefore computed as $\frac{1}{N}\sum_{i=1}^{N}|\mathbf{y}_i|$, where $|\mathbf{y}_i|$ denotes the number of entries in $\mathbf{y}_i$ that are equal to 1, *i.e.* the number of relevant labels of $\mathbf{x}_i$. For some datasets the label cardinality may be small compared to K, whereas for other datasets it can be large. Label density of a multi-label dataset is defined as the average number of relevant labels per observation divided by K. By considering the size of the label space, label density gives the frequency of the labels among all the instances. Label cardinality and label density are often used to describe the label spaces of multi-label datasets in the literature. However, to quantify the imbalance of a multi-label dataset, specific multi-label imbalance measures have also been proposed, as outlined next.

For single-label classification an Imbalance Ratio (IR) can simply be defined as the ratio between the number of observations belonging to the majority class and the number of observations belonging to the minority class. Thus, the higher the IR, the larger the level of imbalance in a single-label classification dataset (Charte *et al*., 2013). A similar procedure can be followed per label for a multi-label dataset. This would evaluate the within-label imbalance of the dataset. However, to consider between-label imbalance, Charte *et al.* (2013) define another imbalance ratio per label, $IRperLabel\left(k\right)$, given as:

$$IRperLabel\left(k\right)=\frac{\max\limits_{\{k'\}_1^K}\left(\sum_{i=1}^{N}y_{i,k'}\right)}{\sum_{i=1}^{N}y_{i,k}}.$$

For the most frequent label, $IRperLabel\left(k\right)=1$ and all other labels will have a value greater than 1. Higher values indicate more imbalance. The highest imbalance ratio of a dataset is denoted by $MaxIR$.

Furthermore, Charte *et al.* (2013) define $MeanIR$ (the average value of $IRperLabel\left(k\right)$) and $CVIR$ (the coefficient of variation of $IRperLabel\left(k\right)$). These measures are defined as follows:

$$MeanIR=\tfrac{1}{K}\sum_{k=1}^{K}\left(IRperLabel\left(k\right)\right)$$

and

$$CVIR=\frac{SDIR}{MeanIR}\quad \text{where}\quad SDIR=\sqrt{\frac{\sum_{k=1}^{K}\left(IRperLabel\left(k\right)-MeanIR\right)^2}{K-1}}.$$

Another aspect to consider when evaluating the imbalance of a multi-label dataset is the concurrence of the labels. In this case it is investigated whether there are some labels that often jointly occur with other imbalanced labels. Charte *et al*. (2014) designed the $SCUMBLE$ (Score of ConcUrrence among iMBalanced LabEls) measure. The measure aims to quantify the imbalance variation among the labels present in each observation. It provides a score in the range $[0,1]$ where higher scores imply more concurrence among imbalanced labels. The measure is based on $IRperLabel(k)$ as well as the Atkinson index (Atkinson, 1970), which is an econometric measure used to assess social inequalities among individuals in a population.

$SCUMBLE$ is defined as follows:

$$SCUMBLE = \tfrac{1}{N}\sum_{i=1}^{N} SCUMBLE_{ins(i)}$$

where $$SCUMBLE_{ins(i)} = 1 - \left(\overline{IRperLabel(k)}_i\right)^{-1}\left(\prod_{k=1}^{K}\left(IRperLabel(k)\right)^{y_{i,k}}\right)^{\frac{1}{\sum_{k=1}^{K} y_{i,k}}}$$

and $$\overline{IRperLabel(k)}_i = \frac{\sum_{k=1}^{K} y_{i,k}\left(IRperLabel(k)\right)}{\sum_{k=1}^{K} y_{i,k}} .$$

For a measure of relative spread of concurrence over the instances, the coefficient of variation, $SCUMBLE.CV$, is calculated as:

$$SCUMBLE.CV = \tfrac{SCUMBLE\sigma}{SCUMBLE} \quad \text{where} \quad SCUMBLE\sigma = \sqrt{\sum_{i=1}^{N}\frac{\left(SCUMBLE_{ins(i)} - SCUMBLE\right)^2}{N-1}} .$$

Table 2.1 provides a summary of the above-mentioned label-specific properties of multi-label data.

Table 2.1
Summary of label-specific properties of multi-label data

Measure	Description	Definition
$Bound$	The maximum number of labelsets that may exist in the dataset	$Bound = 2^K$
$Diversity$	The percentage of distinct labelsets present in the dataset considering the total possible number of labelsets	$Diversity = \frac{number\ \ of\ distinct\ labelsets}{Bound}$
$Label\ \ cardinality$	The average number of relevant labels per observation	$Label\ \ cardinality = \frac{1}{N}\sum_{i=1}^{N}\lvert\mathbf{y}_i\rvert$, where $\lvert\mathbf{y}_i\rvert$ denotes the number of entries in $\mathbf{y}_i$ that are equal to 1
$Label\ \ density$	The average number of relevant labels per observation divided by the total number of labels	$Label\ \ density = \frac{1}{K}\left(\frac{1}{N}\sum_{i=1}^{N}\lvert\mathbf{y}_i\rvert\right)$, where $\lvert\mathbf{y}_i\rvert$ denotes the number of entries in $\mathbf{y}_i$ that are equal to 1
$IRperLabel(k)$	Imbalance ratio for label Y_k	$IRperLabel(k) = \frac{\max_{\{k'\}_1^K}\left(\sum_{i=1}^{N} y_{i,k'}\right)}{\sum_{i=1}^{N} y_{i,k}}$
$MaxIR$	The highest imbalance ratio of a dataset	$\max_{\{k\}_1^K}\left(IRperLabel(k)\right)$
$MeanIR$	The average imbalance ratio of the dataset	$MeanIR = \frac{1}{K}\sum_{k=1}^{K}\left(IRperLabel(k)\right)$
$CVIR$	The coefficient of variation of the imbalance ratios of the dataset	$CVIR = \frac{SDIR}{MeanIR}$, where $SDIR = \sqrt{\frac{\sum_{k=1}^{K}\left(IRperLabel(k) - MeanIR\right)^2}{K-1}}$
$SCUMBLE$	The score of concurrence among imbalanced labels	$SCUMBLE = \frac{1}{N}\sum_{i=1}^{N} SCUMBLE_{ins(i)}$, where $SCUMBLE_{ins(i)}$ is defined as $1-\left(\overline{IRperLabel(k)}_i\right)^{-1}\left(\prod_{k=1}^{K}\left(IRperLabel(k)\right)^{y_{i,k}}\right)^{\frac{1}{\sum_{k=1}^{K} y_{i,k}}}$ and $\overline{IRperLabel(k)}_i = \frac{\sum_{k=1}^{K} y_{i,k}\left(IRperLabel(k)\right)}{\sum_{k=1}^{K} y_{i,k}}$

2.3 Multi-label benchmark datasets

As referred to in Section 1.4.3, standard publicly available multi-label benchmark datasets are used for experimentation and comparison of multi-label learning methods. When selecting datasets for research, it is important to select a diverse set of benchmark datasets. Fortunately, as discussed in Section 1.3, these benchmark datasets cover a range of domains including but not limited to image, text, video, music, biology and audio. The datasets also differ with respect to size and other measurable descriptive properties as defined in the previous section. Furthermore, these datasets come pre-divided into standard training and testing parts. This allows for easy comparison between different research papers.

Online multi-label dataset repositories give researchers easy access to multi-label benchmark datasets. In this dissertation the Mulan[4] repository is used, since it is also used by Madjarov *et al.* (2012), and the empirical study outlined in Chapter 4 considers their work. In addition, Mulan is a very popular and frequently cited repository for multi-label data. Another popular online multi-label dataset repository is Cometa[5] (Charte *et al.*, 2018). Charte *et al.* (2018) also developed the R-package *mldr.datasets*. This package allows users to import multi-label datasets from web-based repositories and view properties of the datasets such as the list of labels and labelsets, as well as a list of descriptive measures such as those outlined in Section 2.2. Other multi-label dataset repositories include KEEL[6] (Alcalá-Fdez *et al*., 2011) and MEKA[7]. Recently, Kostovska *et al.* (2022) also introduced an online catalogue[8] of 89 benchmark datasets. Their web-based system allows users to interactively inspect the available datasets. Kostovska *et al.* (2022) claim that their catalogue links to the largest number of publicly available multi-label classification datasets.

---

[4] http://www.uco.es/kdis/mllresources/ (accessed 14 October 2022)
[5] https://cometa.ujaen.es/datasets/ (accessed 14 October 2022)
[6] https://sci2s.ugr.es/keel/multilabel.php (accessed 14 October 2022)
[7] https://sourceforge.net/projects/meka/files/Datasets/ (accessed 14 October 2022)
[8] http://semantichub.ijs.si/MLCdatasets/home (accessed 14 October 2022)

Table 2.2 summarises some of the most well-known and widely used benchmark datasets in the multi-label literature. A short description of each of these datasets is provided below Table 2.2.

Table 2.2
Summary of well-known publicly available multi-label benchmark datasets

Dataset	Domain	Training cases, N	Testing cases	p	K	Label cardinality	Label density	$MaxIR$	$MeanIR$	$CVIR$
Emotions	music	391	202	72	6	1.868	0.311	1.784	1.478	0.180
Scene	image	1 211	1 196	294	6	1.074	0.179	1.464	1.254	0.122
Yeast	biology	1 500	917	103	14	4.237	0.303	53.412	7.197	1.884
Medical	text	645	333	1449	45	1.245	0.028	266.000	89.501	1.148
Enron	text	1 123	579	1001	53	3.378	0.064	913.000	73.953	1.960
Corel5k	image	4 500	500	499	374	3.522	0.009	1120.000	189.568	1.527
Tmc2007	text	21 519	7 077	49060	22	2.220	0.101	41.980	17.134	0.814
Mediamill	video	30 993	12 914	120	101	4.376	0.043	1092.550	256.400	1.175
Birds	audio	322	323	260	19	1.014	0.053	17.167	5.407	0.817
Delicious	text	12 920	3 185	500	983	18.936	0.019	923.286	72.563	0.777

- **Emotions**

For the *Emotions* data (Trohidis *et al*., 2008) each observation is a piece of music. Each piece of music is labeled according to the emotions it provokes when it is listened to. Six possible emotions exist, namely: *Sad-Lonely*, *Angry-Aggressive*, *Amazed-Surprised*, *Relaxing-Calm*, *Quiet-Still*, and *Happy-Pleased*.

- **Scene**

The *Scene* dataset (Boutell *et al*., 2004) is a scene classification dataset. Each observation is a scene that can be annotated in the following six contexts: *Beach*, *Sunset*, *Field*, *Fall-foliage*, *Mountain* and *Urban*.

- **Yeast**

Yeast (Elisseeff and Weston, 2001) is a widely used dataset where each observation is a gene and each gene can be associated with the following 14 biological functional groups: *Metabolism*, *Energy*, *Transcription*, *Protein synthesis*, *Protein destination*, *Cell growth*, *Transport facilitation*, *Cell transport*, *Cellular biogenesis*, *Ionic homeostasis*, *Cellular organization*, *Transportable elements*, *Cell communication* and *Cell death & ageing*.

- **Medical**

Medical (Pestian *et al*., 2007) is a text dataset where each observation is a document that gives a summary of a patient's symptoms history. These documents are represented using the bag-of-words strategy. The labels of the dataset represent 45 possible diseases. The goal is therefore to build a classifier that can annotate a document containing a patient's symptoms with the probable diseases the patient could have.

- **Enron**

Enron (Klimt and Yang, 2004) is a dataset containing e-mails where each e-mail is represented using the bag-of-words strategy. Each e-mail is annotated with a selection of 53 possible topics. Some of these topics for example include: *Company strategy*, *Legal advice* and *Humour*.

- **Corel5k**

The *Corel5k* dataset (Duygulu *et al*., 2002) consists of a total of 5000 images obtained through the software company Corel. Each image is segmented using the normalised-cuts method. The segmented regions are then clustered into 499 bins which are further used to describe the images. The images are assigned to a wide range of 374 possible labels. Some examples of these labels include: *City*, *Sea*, *Grass*, *Bridge*, *People*, *Dog*, *Tiger*, *Athlete*, *Spider*, *Tent*, *Clouds*, *Plane* and *Runway*.

- **Tmc2007**

Tmc2007 (Srivastava and Zane-Ulman, 2005) is a text dataset where each observation is an aviation safety report obtained by crew members about various events during a flight. These documents are represented using the bag-of-words method. The 22 labels of this dataset represent the problems being described by these reports.

- **Mediamill**

The *Mediamill* dataset (Snoek *et al.*, 2006) is a video indexing dataset. The training dataset contains 85 hours of international broadcast news data. Each video instance is represented as a 120-dimensional input variable vector of numeric variables. There is a total of 101 possible labels for each video. Examples of these labels include: *Aircraft*, *Building*, *Court*, *Anchor*, *Flag*, *Weather, Truck* and *Explosion*.

- **Birds**

Birds (Briggs *et al*., 2013) is an audio dataset consisting of recordings of bird species. In each recording, multiple bird species may appear. The labels represent 19 different bird species.

- **Delicious**

Delicious (Tsoumakas *et al*., 2008) is a text dataset obtained from the social bookmarking web service “del.ico.us”. In a nutshell, social bookmarking is used to organise discussion topics into specific rooms or threads so that users can follow content that interest them. The dataset consists of web pages represented in the bag-of-words format and the labels of each observation correspond to the different tags that appear on the bookmarking website.

2.4 Multi-label classification evaluation measures

In supervised learning it is common practice to train a classifier on the so-called training dataset and thereafter evaluate the performance of the model using the separate testing dataset. Conventional metrics for evaluating the performance of a single-label classifier include misclassification error, accuracy, F-score, and the area under the Receiver Operating Characteristic (ROC) curve. However, due to the complexity of the label space of multi-label data, it is more difficult to evaluate the performance of a multi-label classifier. Consequently, a range of multi-label evaluation measures have been proposed that assess different aspects of the performance of a multi-label classifier.

In general, multi-label classification evaluation measures can be grouped into two main categories: example-based measures and label-based measures (Tsoumakas *et al*., 2010).

Example-based measures consider each data observation individually and computes the metric for each. The final performance value is given by the average performance across the individual observations. On the other hand, label-based metrics consider each label individually and computes the metric for each label.

Consider a multi-label test dataset $\{(\mathbf{x}_i, \mathbf{y}_i),\ i = 1, 2, ..., N_{new}\}$, and suppose the multi-label classifier gives classifications $f(\mathbf{x}_i) = \mathbf{z}_i$, $i = 1, 2, ..., N_{new}$ where $\mathbf{z}_i \in \{0,1\}^K$. Furthermore, denote by TP_k, TN_k, FP_k, and FN_k the number of true positive, true negative, false positive and false negative classifications after binary evaluation of Y_k, $k = 1, 2, ..., K$. The confusion matrix (Figure 2.1) summarises the classification types.

		True	
		1	**0**
Predicted	**1**	TP	FP
	0	FN	TN

Figure 2.1 Confusion matrix

Multi-label example-based measures and label-based measures are defined and discussed in Section 2.4.1 and Section 2.4.2 respectively.

2.4.1 Example-based evaluation measures

Example-based measures used in this work are Hamming loss, accuracy, precision, recall, F-score and subset accuracy.

Hamming loss is defined by:

$$Hamming\ \ loss = \frac{1}{N_{new}} \sum_{i=1}^{N_{new}} \left(\frac{1}{K} \sum_{k=1}^{K} \left| y_{i,k} - z_{i,k} \right| \right).$$

Therefore, for each observation, the true set of classifications is compared to the predicted set of classifications to calculate the proportion of incorrectly predicted labels. Hamming loss is therefore defined as the average proportion of incorrectly predicted labels across the test data observations. Consequently, smaller values for Hamming loss indicate better performance and a value of 0 shows perfect classification. For all other evaluation measures used in this work, higher values indicate better performance.

Accuracy is defined by:

$$accuracy = \tfrac{1}{N_{new}} \sum_{i=1}^{N_{new}} \frac{\left|\mathbf{y}_i \cap \mathbf{z}_i\right|}{\left|\mathbf{y}_i \cup \mathbf{z}_i\right|}.$$

In other words, for each test observation the average Jaccard similarity coefficient between the predicted set of labels and the true set of labels is calculated. Accuracy is defined by the average of these values across the test data observations.

Note that accuracy is undefined once both $\mathbf{y}_i$ and $\mathbf{z}_i$ consist of only 0 entries.

Precision is defined by:

$$precision = \tfrac{1}{N_{new}} \sum_{i=1}^{N_{new}} \frac{\sum_{k=1}^{K} y_{i,k}\, z_{i,k}}{\sum_{k=1}^{K} z_{i,k}}.$$

Precision gives the average proportion of labels that are predicted to be present which are in truth present. Models that give high precision are characterised as more conservative since they give a small number of false positive classifications.

Note that in the above expression precision is undefined if $\sum_{k=1}^{K} z_{i,k} = 0$. In this case $f\left(\mathbf{x}_i\right)$ gives all K labels as absent for $\mathbf{x}_i$. Therefore, denote by $N_{z-present}$ the collection of observations for which $\mathbf{z}_i$ has at least one non-zero entry. Precision is then defined by:

$$precision = \frac{1}{|N_{z-present}|} \sum_{i \in N_{z-present}} \frac{\sum_{k=1}^{K} y_{i,k} \, z_{i,k}}{\sum_{k=1}^{K} z_{i,k}} ,$$

where $|N_{z-present}|$ is the number of observations in $N_{z-present}$.

Recall is defined by:

$$recall = \frac{1}{N_{new}} \sum_{i=1}^{N_{new}} \frac{\sum_{k=1}^{K} y_{i,k} \, z_{i,k}}{\sum_{k=1}^{K} y_{i,k}} .$$

Recall gives the average proportion of truly relevant labels that are predicted to be present. Models with high recall therefore give a small number of false negative classifications.

Recall is undefined if $\sum_{k=1}^{K} y_{i,k} = 0$. In this case $\mathbf{x}_i$ has all K labels absent in truth. Therefore, denote by $N_{y-present}$ the collection of observations for which $\mathbf{y}_i$ has at least one non-zero entry. Recall is then defined by:

$$recall = \frac{1}{|N_{y-present}|} \sum_{i \in N_{y-present}} \frac{\sum_{k=1}^{K} y_{i,k} \, z_{i,k}}{\sum_{k=1}^{K} y_{i,k}} ,$$

where $|N_{y-present}|$ is the number of observations in $N_{y-present}$.

A trade-off exists between precision and recall. For example, although many labels that are classified as present by models with high precision are truly relevant, the model may simultaneously leave out relevant labels by wrongly classifying labels as absent. This gives high precision and low recall. On the other hand, many truly relevant labels are classified as present by models with high recall. However, besides the labels that are truly relevant, the model may also classify non-relevant labels as present. This gives high recall and low precision.

The **F-score** is a harmonic mean that captures the trade-off between precision and recall and it is calculated as:

$$F - score = \frac{2 \times precision \times recall}{precision + recall}.$$

Finally, **subset accuracy** is defined by:

$$subset\ accuracy = \frac{1}{N_{new}} \sum_{i=1}^{N_{new}} I(\mathbf{z}_i = \mathbf{y}_i).$$

In the above expression $I(true) = 1$, whereas $I(false) = 0$. Therefore, subset accuracy is a very strict evaluation measure as it requires the predicted set of labels to be an exact match to the true set of labels.

2.4.2 Label-based evaluation measures

The label-based measures used in this work are macro-precision, macro-recall and macro-F1, as well as micro-precision, micro-recall and micro-F1.

Note that if a binary evaluation measure, calculated from the number of TP, TN, FP and FN classifications, is denoted by $B(TP, TN, FP, FN)$, the so-called macro-averaged and micro-averaged versions of B are given by:

$$B_{macro} = \frac{1}{K} \sum_{k=1}^{K} B(TP_k, TN_k, FP_k, FN_k) \text{ and}$$

$$B_{micro} = B\left(\sum_{k=1}^{K} TP_k, \sum_{k=1}^{K} TN_k, \sum_{k=1}^{K} FP_k, \sum_{k=1}^{K} FN_k \right).$$

In other words, with macro-averaging the measure is calculated per label and thereafter averaged across the labels. However, with micro-averaging all the TP, TN, FP and FN are added across the labels and used to calculate the measure.

Consequently, **macro-precision**, **macro-recall** and **macro-F1** are computed as:

$$macro-precision = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_k}{TP_k+FP_k},$$

$$macro-recall = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_k}{TP_k+FN_k} \quad \text{and}$$

$$macro-F1 = \frac{1}{K}\sum_{k=1}^{K}\frac{2\times P_k\times R_k}{P_k+R_k},$$

where P_k and R_k denote the precision and recall for Y_k.

Note that macro-precision is undefined if one or more of the labels are never classified as present by the multi-label classifier. Therefore, macro-precision only takes an average over those labels classified as having at least one non-zero entry. Similarly, macro-recall is undefined if one or more of the labels are never present. Therefore, when calculating macro-recall, the average is taken over those labels that have at least one non-zero entry.

Finally, **micro-precision**, **micro-recall** and **micro-F1** are computed as:

$$micro-precision = \frac{\sum_{k=1}^{K}TP_k}{\sum_{k=1}^{K}TP_k+\sum_{k=1}^{K}FP_k},$$

$$micro-recall = \frac{\sum_{k=1}^{K}TP_k}{\sum_{k=1}^{K}TP_k+\sum_{k=1}^{K}FN_k} \quad \text{and}$$

$$micro-F1 = \frac{2\times(micro-precision)\times(micro-recall)}{(micro-precision)+(micro-recall)}.$$

Note that the area under the ROC curve is not often considered in the multi-label literature. For traditional single-label classification, the ROC curve is constructed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) of a classifier when various classification thresholds are applied. However, it is not a straightforward task to construct a similar curve for a multi-label classifier, due to the complexity of the label space. For example, the TPR and FPR can be calculated in either an example-based manner or a label-based manner when a multi-label classifier is considered. Furthermore, either a global threshold can be applied for all labels or different thresholds can be considered per label.

2.5 Multi-label software

A range of multi-label software exists for different programming languages. The different software often include functions for fitting well-known multi-label classifiers and functions for calculating evaluation measures such as those outlined in the previous section. Some software also provide easy access to a range of multi-label benchmark datasets and give specialised plots for closer investigation of the label space. Researchers can therefore conveniently fit and compare the performance of different multi-label learning methods by making use of the available software.

In this dissertation the programming language R is preferred. Therefore, all empirical work, such as that of Chapter 4 and Chapter 6, are done in R. A useful R-package designed to handle multi-label datasets is the package *mldr* (Charte and Charte, 2019). The package allows users to load multi-label datasets, obtain characteristics from the data such as those defined in Section 2.2, and contains built-in functions to produce specialised plots from the data. Charte and Charte (2015) provide a useful guideline for working with *mldr*. Furthermore, a useful R-package for multi-label classification is the *utiml* package (Rivolli, 2021). The package contains built-in functions to fit popular multi-label learning models and to calculate standard multi-label evaluation measures. Rivolli and de Carvalho (2018) give a detailed description of the *utiml* package.

A Python library for performing multi-label classification is *scikit-multilearn* (Szymanski and Kajdanowicz, 2018). The library conveniently provides popular multi-label classification algorithms, fast calculation of multi-label evaluation measures, as well as other operations to inspect the label space of a multi-label dataset. Szymanski and Kajdanowicz (2019) give a detailed discussion of *scikit-multilearn*.

Another well-known software package for multi-label learning is the Java library Mulan (Tsoumakas *et al.*, 2011b). Similar to the above-mentioned software, Mulan provides a range of multi-label classification algorithms and evaluation measures.

Furthermore, MEKA (Read *et al.*, 2016) is based on the WEKA machine learning toolkit (a collection of machine learning algorithms from the University of Waikato) and provides open-source implementation of multi-label learning methods such as Classifier Chains (CC) and Ensemble of Classifiers Chains (ECC) which are discussed in Section 2.7.2 and Section 2.7.5 respectively.

Finally, the CLUS system (Blockeel and Struyf, 2002) implements the predictive clustering framework discussed in Section 2.7.3.

Note that to conduct the extensive experimental comparison of different multi-label learning methods in Madjarov *et al*. (2012), the authors make use of the Mulan library, as well as the MEKA extension for the WEKA framework, and the CLUS system. The empirical study outlined in Chapter 4 considers their work.

2.6 Categorisation of multi-label learning methods based on label correlation

This section provides more detail with regards to label dependency and label correlation in multi-label data.

It should be noted that the term “label correlation” is often used in an intuitive manner in multi-label literature to indicate a kind of non-independence between labels (Dembczynski *et al*., 2010). Few authors consider a precise formal definition. Consequently, a wide range of classification proposals are found in the literature that explicitly or implicitly explore label correlations (Wang *et al.*, 2014).

Methods that explicitly exploit label correlations include those that for example model the dependencies among labels using a Bayesian Network (BN) (Wang *et al.*, 2014), group dependent labels based on χ^2 – scores such as the ChiDep method of Tenenboim-Chekina *et al.* (2010), or estimate label correlations by the co-occurrence of labels in training data

(Petterson and Caetano, 2011). On the other hand, methods that implicitly exploit label correlations include those that for example extend the input variable space with the label indicator variables (Read *et al.*, 2011) or with label outputs of a first level classifier (also referred to as multi-label stacking – see Tsoumakas *et al*. (2009)).

Apart from this distinction between explicit and implicit utilisation of label correlation, a distinction is also drawn between global and local label correlations in the multi-label literature. As explained in Section 1.4.1, when label correlations are viewed globally, it is assumed that the label correlations are shared globally by all instances. However, when label correlations are viewed locally, label correlations are assumed to be shared only by subsets of instances. Since existing multi-label learning methods mostly focus on global label correlation exploitation, a method for local correlation exploitation is proposed in this dissertation, namely LDsplit.

One previous proposal for the exploitation of local correlations between labels is given by Nasierding *et al.* (2009). The Clustering-Based Multi-Label Classification (CBMLC) framework proposed by the authors consists of two steps. In the first step the training instances are clustered into k clusters by using a clustering algorithm such as $k-$means clustering. The label space is not considered in this step. Each cluster is hereafter annotated with only those labels that are present for at least one of the training observations in the cluster. Nasierding *et al.* (2009) argue that similar observations should be associated with similar labels, so that this step would reduce the label space associated with each cluster. In the second step a multi-label classifier is fit to each cluster considering its associated label space. A possible disadvantage of this approach is that the label space is excluded in the clustering step.

Another previous proposal for the exploitation of local correlations between labels is given by Huang and Zhou (2012), who argue that it can be beneficial to exploit label correlations locally by dividing the data into subgroups. In their work the training data are divided into subgroups by applying clustering on the label space. From these clusters a LOC (Local

Correlation) code is developed for each data case and these LOC codes are then used as an additional input variable for the multi-label classification task. However, since these LOC codes are based on co-occurrence in the label space, the original input variables are excluded when determining the local correlation structure. Furthermore, in situations where the dimensionality of the input space is large, such a code may be less discriminative and could end up being dominated by the original input variables (Zhu *et al.*, 2017).

The novel multi-label classification method proposed in this dissertation, LDsplit, also aims to exploit label correlations locally by dividing the data into subgroups. However, LDsplit uses a tree-based structure to do so. In general, decision trees divide data into subgroups by taking local decisions about how best to model label dependency. This property therefore provided a natural starting point for developing a multi-label classification method that aims to implicitly exploit local label correlations by dividing the data into subgroups. As will become clear in Chapter 3, LDsplit improves upon the methods of Nasierding *et al.* (2009) and Huang and Zhou (2012), since interactions of original input variables and labels are considered when determining the data subgroups. Furthermore, LDsplit requires no input variable augmentation.

A further means of categorisation of learning methods with regard to label correlation is given by Zhang and Zhang (2010). These authors outline three categories that are based on the order of label correlations considered by the learning method: first-order, second-order and higher-order approaches. First-order approaches do not exploit label correlations in the fitting process. These methods decompose the multi-label classification problem into multiple independent binary classification problems. Second-order approaches consider pairwise relations between labels, whereas higher-order approaches consider an even higher order of correlation between labels than the first- or second-order approaches. Higher-order approaches include methods with a random style of forming an ensemble of classifiers that addresses correlations among random subsets of labels. Unfortunately, inconsistencies were found in the literature regarding the categorisation of learning methods (especially tree-based methods) into these three categories. This could be a consequence of the absence of a standard definition of label correlation in the multi-label literature. Consequently, when these categories are referred to in this dissertation, brief reasonings will be included to motivate the specific categorisation of the learning method. Interestingly, by referencing the work of Tsoumakas *et al*. (2010), Zhang and Zhang (2010) also refer to the term “label correlation” in an intuitive manner to indicate a kind of non-independence between labels. However, in their proposed model, Zhang and Zhang (2010) exploit label correlations explicitly by making use of a BN to model the dependencies among labels.

The categorisation of learning methods outlined by Madjarov *et al*. (2012) are in general preferred in this dissertation. These categories of Madjarov *et al*. (2012) are used as a framework in the next section to discuss various multi-label classifications methods related to LDsplit.

In conclusion, Figure 2.2 provides a summary of the different viewpoints in the multi-label literature for the categorisation of learning methods based on label correlation.

It is motivated in Chapter 3 that LDsplit aims to implicitly exploit local higher-order label correlations.

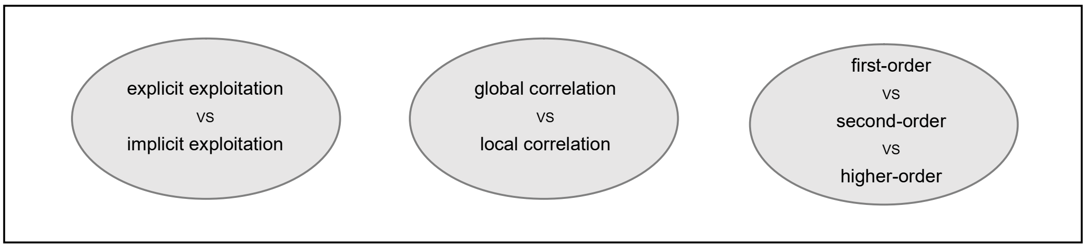


Figure 2.2 Different viewpoints for the categorisation of learning methods based on label correlation

2.7 Multi-label learning methods

Due to the increase of multi-label datasets in recent years, more research proposing new multi-label classification methods have emerged. This section refers to some of these methods that are related to the work presented in other chapters of this dissertation.

2.7.1 Categories

In general, Madjarov *et al*. (2012) group multi-label learning methods into three categories: problem transformation methods, algorithm adaptation methods and ensemble methods.

Problem transformation methods are algorithm independent. The multi-label learning task is transformed into one or more single-label classification tasks so that single-label classification methods can be applied. Some well-known problem transformation methods are described in Section 2.7.2, including Binary Relevance (BR) (Tsoumakas and Katakis, 2007), Classifier Chains (CC) (Read *et al.*, 2011), Label Powerset (LP) (Tsoumakas and Katakis, 2007), and pairwise methods such as Calibrated Label Ranking (CLR) (Fürnkranz *et al.*, 2008) and the adaptation of the Quick Weighted Voting algorithm for multi-label learning (QWML) (Mencía *et al.*, 2010).

Algorithm adaptation methods adapt a single-label learning algorithm to handle multi-label data directly. Examples include an adaptation of the popular $k-$Nearest Neighbours algorithm to multi-label data (ML-kNN) (Zhang and Zhou, 2007), adaptations of the Support Vector Machine (SVM) namely Rank-SVM (Elisseeff and Weston, 2001), extensions of AdaBoost for multi-label data namely AdaBoost.MH and AdaBoost.MR (Schapire and Singer, 2000) and neural networks for multi-label data such as BackPropagation for Multi-Label Learning (BP-MLL) (Zhang and Zhou, 2006) and an adaptation of deep Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in CNN-RNN (Wang *et al.*, 2016). Decision trees have also been adapted for multi-label learning. Some examples of multi-label tree-based methods found in the literature are Multi-Label C4.5 (ML-C4.5) (Clare and King, 2001), Multi-Objective Decision Trees (MODTs) (Kocev *et al.*, 2007), which are an instantiation of Predictive Clustering Trees (PCT) (Blockeel *et al*., 1998), LaCova (Al-Otaibi *et al*., 2014b), a hybrid decision tree utilising local SVMs (ML-SVMDT) (Gjorgjevikj *et al*., 2013) and Hierarchy Of Multi-label learnERs (HOMER), initially presented in Tsoumakas *et al.* (2008). In Chapter 3, a new tree-based multi-label classification method is proposed, namely LDsplit. Therefore, tree-based methods are of particular interest in this work and are discussed in more detail in Section 2.7.3.

Finally, multi-label ensemble schemes use several multi-label classifiers and combine them in a specified way to build one powerful classifier. By doing this the generalisation ability is improved and the risk of overfitting reduced (Moyano *et al*., 2018). A necessary condition for an ensemble to be more accurate than any of its individual members, is that the classifiers are accurate and diverse (Kocev *et al*., 2007). Here a classifier is considered “accurate” if it does better than random guessing on new observations. Furthermore, two classifiers are “diverse” if these classifiers make different errors on new observations.

Such statistical ensemble methods can be compared to the concept of “the wisdom of crowds”. Suppose for example information on a certain subject is needed. There likely exists an individual with vast knowledge on the subject compared to most individuals with limited knowledge. However, the highly informed individual may be difficult to track down for questioning. In that case, instead of asking a single less informed individual for information, a better strategy may be to ask a crowd of little-informed individuals and combine all the information received from the crowd. This relates to the three reasons given by Dietterich (2000) why statistical ensemble methods can be better than a single classifier namely: (i) a bad classifier can be the one selected if only one classifier is used, (ii) since learning algorithms often use local search, running the algorithm many times and combining the output may result in a better approximation of the unknown optimal classifier, and (iii) for many problems the optimal function cannot be found, but a combination of several classifiers may give a close approximation.

LDsplit fits an ensemble of trees. Therefore, multi-label tree ensemble methods are of particular interest in this work, such as Random Forests of ML-C4.5 (RFML-C4.5), Random

Forests of Predictive Clustering Trees (RF-PCT) (Kocev *et al.*, 2007), Sparse Oblique Structured Hellinger Forests (SOSHF) (Daniels and Metaxas, 2017) and Multi-Label Forests (ML-Forests) (Wu *et al.*, 2016). These methods are discussed in Section 2.7.4. Other well-known multi-label ensemble methods related to LDsplit are discussed in Section 2.7.5, namely Ensemble of Binary Relevance (EBR) (Read *et al.*, 2011), Ensemble of Classifier Chains (ECC) (Read *et al.*, 2011), and Random $k-$labelsets (RAkEL) (Tsoumakas *et al.*, 2011a).

Figure 2.3 summarises the three categories of multi-label learning as given by Madjarov *et al.* (2012), including examples of learning methods found in each category. As outlined in Figure 2.3, methods related to LDsplit are discussed in the next four sections.

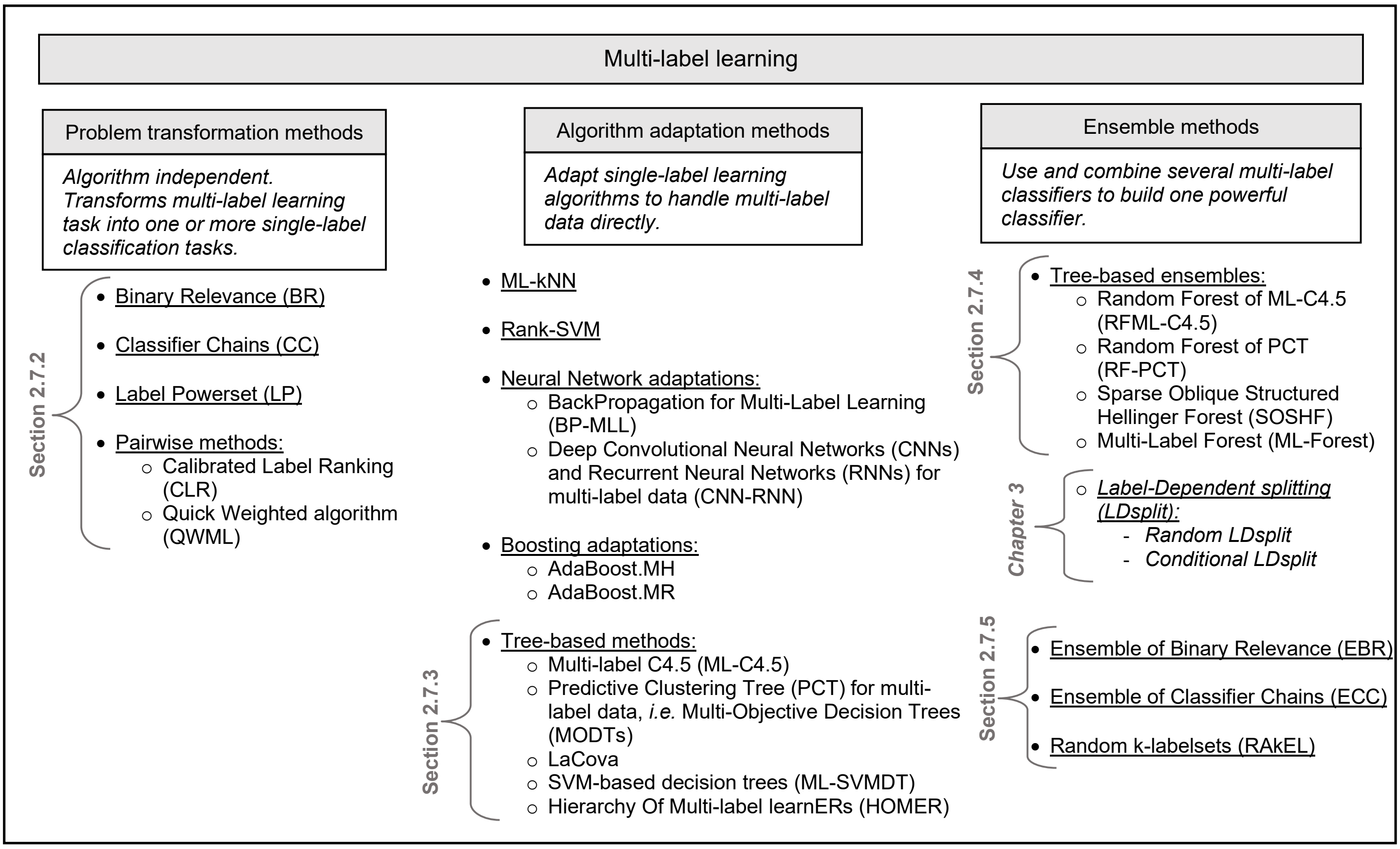


Figure 2.3 Categories of multi-label learning

2.7.2 Problem transformation methods

Tsoumakas and Katakis (2007) and Tsoumakas *et al.* (2010) give detailed discussions of problem transformation methods. However, in this section some of the most popular transformations are briefly described. Table 2.3 gives a simple multi-label dataset with $N=5$ data cases and $K=4$ labels which will be used to illustrate the transformations throughout this section.

Table 2.3
Summary of a simple multi-label dataset with $N=5$ observations and $K=4$ labels

Data cases	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}_1$	0	1	1	0
$\mathbf{x}_2$	1	0	0	0
$\mathbf{x}_3$	0	1	1	1
$\mathbf{x}_4$	1	0	1	0
$\mathbf{x}_5$	1	0	0	0

- **Binary Relevance (BR)**

One of the simplest problem transformation methods may be the one-against-all strategy, BR. BR fits one binary classifier for each of the K labels separately by decomposing the multi-label dataset into K independent binary classification datasets (Tsoumakas and Katakis, 2007). This gives K binary classifiers $f_k : \mathbf{x} \rightarrow Y_k\left(\mathbf{x}\right) \in \{0,1\}$, $k = 1,...,K$. An illustration of the decomposition of the multi-label dataset of Table 2.3 for BR is given by Figure 2.4. To find a multi-label classification for a data case, each of the K binary classifiers classifies the data case independently and the K classifications (one for each label) are combined. In other words, the multi-label classification of a data case, $\mathbf{x}_{new}$, is given by $\left[f_1\left(\mathbf{x}_{new}\right)\ f_2\left(\mathbf{x}_{new}\right)\ ...\ f_K\left(\mathbf{x}_{new}\right)\right]$.

Data cases	Y_1
$\mathbf{x}_1$	0
$\mathbf{x}_2$	1
$\mathbf{x}_3$	0
$\mathbf{x}_4$	1
$\mathbf{x}_5$	1

Binary classifier

f_1

Data cases	Y_2
$\mathbf{x}_1$	1
$\mathbf{x}_2$	0
$\mathbf{x}_3$	1
$\mathbf{x}_4$	0
$\mathbf{x}_5$	0

Binary classifier

f_2

Data cases	Y_3
$\mathbf{x}_1$	1
$\mathbf{x}_2$	0
$\mathbf{x}_3$	1
$\mathbf{x}_4$	1
$\mathbf{x}_5$	0

Binary classifier

f_3

Data cases	Y_4
$\mathbf{x}_1$	0
$\mathbf{x}_2$	0
$\mathbf{x}_3$	1
$\mathbf{x}_4$	0
$\mathbf{x}_5$	0

Binary classifier

f_4

Figure 2.4 Data from Table 2.3 transformed by using BR

It is unanimously agreed in the multi-label literature that since BR fits a binary model for each label independently, label correlations are not explicitly exploited when fitting the model. Therefore, considering the order of label correlations, BR is a first-order approach. BR does however remain an important baseline method for comparison of other multi-label learning methods. In fact, a recent study by Bogatinovski *et al.* (2022), conducted on 42 multi-label benchmark datasets comparing 26 multi-label learning methods, identified BR with random forest decision trees as one of the top five methods considering the average predictive performance across the different problems.

- **Classifier Chains (CC)**

CC is a problem transformation method introduced with the aim to improve on BR (Read *et al.*, 2011). CC fits one binary model for each label as BR does; however, label correlations are incorporated into the model by allowing a random label order to define a chain of K binary classifiers. The input space of each binary classifier is extended with the $0/1$ label relevancies of all previous classifiers in the chain. Since each label is in turn used to extend the input space, CC is regarded as a higher-order label correlation method that exploits label correlations implicitly. Furthermore, for CC it is assumed that label correlations are shared by all the data cases at once, so that CC exploits global label correlations.

Figure 2.5 illustrates the decomposition of the multi-label dataset of Table 2.3 using CC with label order $\left[Y_2\ Y_3\ Y_1\ Y_4\right]$.

In general, a fitted CC model can be used to classify an unseen data case, $\mathbf{x}_{new}$, as follows. Start with the first classifier, f_1, and obtain the $0/1$ classification, $\hat{f}_1\left(\mathbf{x}_{new}\right)$. Now f_2 produces the relevance of the second label in the chain for the data case $\mathbf{x}_{new}$, given the input space augmented by $\hat{f}_1\left(\mathbf{x}_{new}\right)$. Continuing in this way, f_K produces the relevance of the last label in the chain given the input space augmented by all previous classifiers in the chain.

Read *et al.* (2021) give a general definition of CC by discarding the property that a chain should be fully connected. Instead they define CC under two properties. First, one classifier is fit per label which is considered as a node in a chain. Secondly, the chain is any directed acyclic structure in which the output of one classifier becomes input to the subsequent classifier to which it is connected in that structure. With this definition, many extensions and variations of CC exist as outlined in Read *et al.* (2021).

Input	Y_2
$\mathbf{x}_1$	1
$\mathbf{x}_2$	0
$\mathbf{x}_3$	1
$\mathbf{x}_4$	0
$\mathbf{x}_5$	0

Binary classifier f_1

Input		Y_3
	Y_2	
$\mathbf{x}_1$	1	1
$\mathbf{x}_2$	0	0
$\mathbf{x}_3$	1	1
$\mathbf{x}_4$	0	1
$\mathbf{x}_5$	0	0

Binary classifier f_2

Input			Y_1
	Y_2	Y_3	
$\mathbf{x}_1$	1	1	0
$\mathbf{x}_2$	0	0	1
$\mathbf{x}_3$	1	1	0
$\mathbf{x}_4$	0	1	1
$\mathbf{x}_5$	0	0	1

Binary classifier f_3

Input				Y_4
	Y_2	Y_3	Y_1	
$\mathbf{x}_1$	1	1	0	0
$\mathbf{x}_2$	0	0	1	0
$\mathbf{x}_3$	1	1	0	1
$\mathbf{x}_4$	0	1	1	0
$\mathbf{x}_5$	0	0	1	0

Binary classifier f_4

Figure 2.5 Data from Table 2.3 transformed by using CC

- **Label Powerset (LP)**

The LP approach is a problem transformation method that incorporates label correlation into the model by recognising each unique set of labels as a class. The multi-label data are thus transformed to a multi-class structure allowing any multi-class classifier to be used for classification. Figure 2.6 illustrates how the multi-label data of Table 2.3 is transformed to a multi-class dataset with four classes considering the four unique labelsets of Table 2.3. Unfortunately in practice, the transformation of the multi-label data to a multi-class structure may lead to a dataset with many classes and few observations per class. The total number of possible classes for the multi-class version of the data has an upper bound at $\min\left(N, 2^K\right)$. Furthermore, LP can only classify observations to a labelset present in the training data.

Unique labelsets	
0 1 1 0	*i.e.* Y_2 and Y_3 present
1 0 0 0	*i.e.* Y_1 present
0 1 1 1	*i.e.* Y_2, Y_3 and Y_4 present
1 0 1 0	*i.e.* Y_1 and Y_3 present

<u>Transformation to multi-class data</u>

Data cases	Y_2 **and** Y_3 **present**	Y_1 **present**	Y_2, Y_3 **and** Y_4 **present**	Y_1 **and** Y_3 **present**
$\mathbf{x}_1$	1	0	0	0
$\mathbf{x}_2$	0	1	0	0
$\mathbf{x}_3$	0	0	1	0
$\mathbf{x}_4$	0	0	0	1
$\mathbf{x}_5$	0	1	0	0

Figure 2.6 Data from Table 2.3 transformed to multi-class data using LP

- **Calibrated Label Ranking (CLR)**

Fürnkranz *et al.* (2008) introduce CLR as an extension of Ranking by Pairwise Comparison (RPC) (Hüllermeier *et al.*, 2008). RPC applies a pairwise approach (as outlined in Figure 2.7) to extend binary classification algorithms to give a finite ranking of the labels per observation. A total of $\frac{K(K-1)}{2}$ binary datasets are formed, one for each pair of labels (Y_a, Y_b), $1 \leq a < b \leq K$. A dataset constructed for a pair of labels contains all those observations that have one of the two corresponding labels present. A binary classifier fit to such a dataset consequently gives a preference between the two corresponding labels for a new observation. After finding these preferences for a new observation using each of the $\frac{K(K-1)}{2}$ datasets, a label ranking is produced by counting the votes received for each label. Ties can be broken randomly. CLR extends such a ranking to the multi-label scenario by introducing an artificial calibration label to separate the relevant and irrelevant labels within the ranking.

CLR fits K additional binary classifiers that discriminate between each of the original labels and the artificial label. Note that these K additional binary classifiers correspond to the BR model given by Figure 2.4. This is because each observation that has a particular original label present is annotated with "1" for this label and "0" for the artificial label, while each observation that has a particular original label absent is annotated with "0" for this label and "1" for the artificial label. The position of the artificial label in the label ranking of a new observation provides a separation point between relevant and irrelevant labels for this observation.

CLR produces both a ranking and a bipartition of the labels. However, the main disadvantage of CLR is that the method is not suitable for datasets with many labels due to the large exploration space and time complexity (Bogatinovski *et al.*, 2022).

Since CLR considers pairwise correlations between labels, it is categorised as a second-order label correlation approach in the multi-label literature.

Data cases	Y_1	Y_2
$\mathbf{x}_1$	0	1
$\mathbf{x}_2$	1	0
$\mathbf{x}_3$	0	1
$\mathbf{x}_4$	1	0
$\mathbf{x}_5$	1	0

Binary classifier

f_1

Data cases	Y_1	Y_3
$\mathbf{x}_1$	0	1
$\mathbf{x}_2$	1	0
$\mathbf{x}_3$	0	1
$\mathbf{x}_5$	1	0

Binary classifier

f_2

Data cases	Y_1	Y_4
$\mathbf{x}_2$	1	0
$\mathbf{x}_3$	0	1
$\mathbf{x}_4$	1	0
$\mathbf{x}_5$	1	0

Binary classifier

f_3

Data cases	Y_2	Y_3
$\mathbf{x}_4$	0	1

Binary classifier

f_4

Data cases	Y_2	Y_4
$\mathbf{x}_1$	1	0

Binary classifier

f_5

Data cases	Y_3	Y_4
$\mathbf{x}_1$	1	0
$\mathbf{x}_4$	1	0

Binary classifier

f_6

Figure 2.7 Data from Table 2.3 transformed by using RPC

- **Quick Weighted Voting algorithm for multi-label learning (QWML)**

Loza Mencía and Fürnkranz (2008) introduce the Multi-Label Pairwise Perceptron (MLPP) algorithm as an instantiation of RPC with the perceptron algorithm as base classifier. This is done in an attempt to speed up the classification process. Mencía *et al.* (2010) extend this MLPP algorithm using an adaptation of the Quick Weighted Voting (QWeighted) algorithm proposed by Park and Fürnkranz (2007).

The QWeighted algorithm was initially developed for multi-class data. However, instead of stopping once the single top class has been determined, Mencía *et al.* (2010) extend the QWeighted algorithm to multi-label data by repeatedly fitting the algorithm after removing the top class found in each iteration. Similar to CLR, Mencía *et al.* (2010) include an artificial label so that the iterations of the QWeighted algorithm stop once the artificial label is returned as the top class. All labels obtained prior to the artificial label are relevant for the observation, whereas the remaining labels are all irrelevant.

2.7.3 Tree-based methods

In traditional single-label classification, tree-based methods apply recursive binary splitting to the data cases by partitioning the input space. For each split the input variable and split-point are identified that achieves the largest information gain (*i.e.* decrease in node impurity) after the split. Information gain can for example be measured as the difference between the Shannon entropy of the set of training observations being considered for splitting and the weighted sum of the Shannon entropy of the subsets resulting from splitting the observations. The learning method is appropriately named since a tree-like structure is formed. The root node contains the full set of data cases. By moving down either the left or right branches of nodes (as determined by the splitting rules of the nodes) data cases are recursively split into sub-nodes. Tree-growing can be stopped once a minimum node-size is reached. Some methods also make use of post- or pre-pruning. With post-pruning a large tree is grown but pruned backwards by collapsing those nodes which cause the smallest increase in impurity. This is normally guided by a cost complexity criterion. In the case of pre-pruning, node-splitting stops if a given split does not result in a significant reduction in variance. This is for example determined by conducting an F-test.

The terminal nodes of a fitted tree determine the classification of new data cases. The majority class can for example be assigned to each terminal node by considering the training observations found in the node. New data cases that filter down the tree by following the splitting rules are classified to the majority class of the terminal node they end up in.

Popular traditional single-label tree-based methods are CART and C4.5 (Hastie *et al.*, 2009:305). Figure 2.8 gives an example of a binary classification tree fit to a binary classification dataset where each instance is a recorded voice sample of either a male or female voice. Figure 2.8 shows that the first splitting variable is “meanfun” with a splitting-value of 0.14. The terminal nodes indicate the gender assigned to the voice samples that end up in each node.

The conceptionally simple framework of single-label trees have led to adaptations for multi-label data. This section summarises a selection of previously proposed multi-label tree-based classifiers. Note that this section specifically focuses on algorithm adaptations of trees for multi-label data. It does not include problem transformation methods, such as those mentioned in the previous section, with a single-label tree as base classifier.

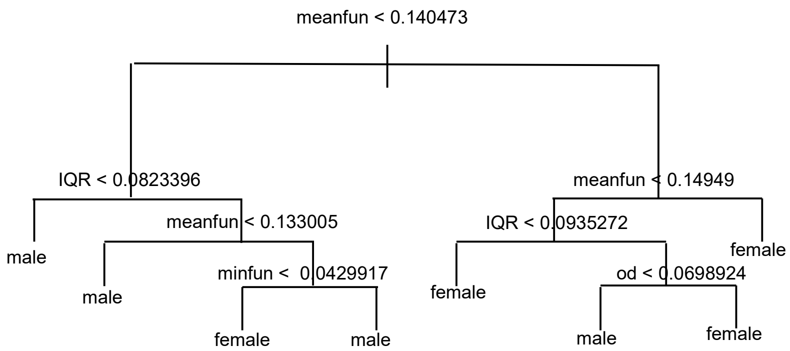


Figure 2.8 Example of fitted binary classification tree

- **Multi-Label C4.5 (ML-C4.5)**

Clare and King (2001) adapt the well-known C4.5 algorithm (Quinlan, 1995) for multi-label data by modifying the formula for calculating Shannon entropy (Shannon, 1948) as a sum of entropies of individual labels.

Let $p(Y_k)$ be the probability (relative frequency) of label Y_k and $q(Y_k) = 1 - p(Y_k)$ be the probability of not having Y_k present. Clare and King (2001) calculate the entropy of a subset as:

$$-\sum_{k=1}^{K} \left(p(Y_k) \log p(Y_k) + q(Y_k) \log q(Y_k) \right).$$

By using this formula for entropy, for each split of the ML-C4.5 tree, the input variable and split-point are identified that achieve the largest information gain after the split. Note that in this case when the information gain of a split is calculated, the weighted sum means that if an item appears twice in a subset because it belongs to two classes, it is counted twice.

After an ML-C4.5 tree has been fit, each terminal node can for example be assigned the majority labelset of the training observations in that node. To classify a new observation, the observation moves through the nodes of the ML-C4.5 tree by following the splitting rules and is assigned the labelset of the terminal node in which it ends.

Such an ML-C4.5 tree can identify input variables that are relevant when considering all the labels simultaneously. However, node-splitting is not explicitly guided by label correlation since the splitting criterion is the sum of individual label entropies. Due to this, the majority of multi-label work that were examined categorise ML-C4.5 as a first-order label correlation method. However, Wu *et al.* (2016) and Bogatinovski *et al.* (2022) categorise ML-C4.5 as a higher-order label correlation method. Bogatinovski *et al.* (2022) state that multi-label trees have a built-in mechanism to deal with label correlation. Although the authors do not expand on this statement, it is surmised that they reason as follows. Each node of the ML-C4.5 tree depends on the splitting rules that sent the specific observations to the node. Suppose I_j denote the indices of the data cases in the j^{th} node. Then:

$$I_1 = \{1, 2, \ldots, N\} = I_2 \cup I_3 = (I_4 \cup I_5) \cup (I_6 \cup I_7) = \ldots .$$

Assume that the first split is on variable X_1, with split-point s_1, resulting in the two subsets of data, $D_2 \triangleq \{(\mathbf{x}_i, \mathbf{y}_i) : x_{i,1} < s_1\}$ and $D_3 \triangleq \{(\mathbf{x}_i, \mathbf{y}_i) : x_{i,1} \geq s_1\}$. A change in the values of $\mathbf{Y}$ will influence D_2 and D_3, and therefore also the splits of the data occurring lower in the tree. In other words, when calculating the sum of individual label entropies for lower-level nodes, this sum is calculated conditional on the previous splits. Consequently, the previous splits, which are based on more than one label, influence the lower-level splits of the observations. Label correlations can therefore be exploited implicitly by the model.

- **Multi-Objective Decision Trees (MODTs)**

Blockeel *et al*. (1998) describe a Predictive Clustering Tree (PCT) as a hierarchy of clusters. The root node corresponds to one cluster containing all the data, which is recursively partitioned into smaller clusters while moving down the tree. A variance function, which measures cluster impurity, is defined, and used to form the hierarchy of clusters with maximum homogeneity. This is done by maximising the variance reduction achieved when partitioning the training observations into smaller clusters. With this general framework, PCTs can be used for a variety of applications, including multi-label classification. In this case the so-called prototype function would define a multi-label classification for each terminal node of a PCT. Therefore, the main difference between a PCT and a standard decision tree is that the former treats the variance function and the prototype function as parameters that can be instantiated for a given learning task (Kocev *et al.*, 2007).

Kocev *et al.* (2007) explain how MODTs are an instantiation of PCTs where the variance and prototype functions are extended to multi-label data. In this setting cluster impurity, as defined by the variance function, can for example be specified as the sum of the Gini indices of the labels, $\sum_{k=1}^{K} Gini\left(\mathbf{X}, Y_k\right)$. The input variable and split-point are therefore identified per cluster that maximises the reduction in $\sum_{k=1}^{K} Gini\left(\mathbf{X}, Y_k\right)$ achieved after the split.

Here, $\sum_{k=1}^{K} Gini\left(\mathbf{X}, Y_k\right)$ is computed as:

$$\sum_{k=1}^{K} 2p\left(Y_k\right) q\left(Y_k\right),$$

where $p\left(Y_k\right)$ is the probability (relative frequency) of label Y_k and $q\left(Y_k\right) = 1 - p\left(Y_k\right)$ is the probability of not having Y_k present.

If instead the variance function is defined as the sum of entropies per label, and the same node-splitting stopping criterion is used as that of ML-C4.5, the MODT model would fit an ML-C4.5 tree as described in the previous section.

To classify a new observation by means of the fitted MODT, the observation moves through the nodes of the MODT by following the splitting rules until it reaches a terminal node. To produce a multi-label classification, the prototype function of the MODT can for example be defined as the majority vote or posterior probability of each label within the cluster.

The order of label correlations considered by the MODT depends on the definition of the variance function. For example, if the variance function is taken as the sum of Gini indices or label entropies of the labels, the same arguments regarding label correlations hold as given for ML-C4.5 in the previous section.

- **LaCova**

Although less well-known than ML-C4.5 and MODTs, Al-Otaibi *et al.* (2014b) introduce a multi-label tree-based method, LaCova, that uses a splitting criterion based on a label covariance matrix. For a set of K labels, the $K \times K$ label covariance matrix has the respective label variances on the diagonal and the pairwise covariances on the off-diagonal entries of the matrix. At each node of a LaCova tree a label dependency test is conducted. This test uses the label covariance matrix of the node to implement a three-way splitting criterion.

Firstly, Al-Otaibi *et al.* (2014b) argue that tree-growing should stop if the sum of variances of the labels is low. According to their experiments, a variance threshold of 0 works well. If the sum of variances is not considered small, the sum of absolute covariances is computed. If this sum is lower than a threshold value, λ, BR with a decision tree base classifier is applied. This is referred to as a vertical split. The argument is that since labels show signs of independence, it is better to fit a single tree for each label. If the sum of absolute covariances is larger than λ, a single tree is grown for the labels. In other words, the input variable and splitting-value are found which best partitions the data at this stage according to all the labels. This is referred to as a horizontal split. Here, to find the “best” partitioning, the quality of a partition is measured by the minimum of two quantities: the sum of label variances resulting from the partition and the sum of absolute label covariances resulting from the partition.

Figure 2.9 gives an example of a fitted LaCova tree where the root node is split horizontally followed by a vertical and horizontal split for the respective child nodes. Tree-growing stops after the horizontal split of the one child node results in two vertical splits.

A new observation is classified by following the rules of the fitted LaCova tree. If a new observation is for example dropped into the root node of the LaCova tree of Figure 2.9, the observation moves down either the left or right branch of the tree depending on its value for the splitting variable of the root node. If the observation moves down the left branch, the BR model of this node defines the classification for the observation. On the other hand, if the observation initially moves down the right branch of the tree, the splitting rule of this node is considered to determine which branch the observation moves through next. In this case, the final classification is given by the BR model of the node in which it lands.

The decision between a vertical and horizontal split at a node of a LaCova tree is guided by the observed pairwise correlations between the labels. However, the order of label correlations considered by the fitted LaCova tree depends on the three-way splitting criteria of the model, as outlined next.

First, if the sum of absolute covariances at the root node is smaller than λ, BR with a decision tree base classifier is applied at the root node. In such a scenario, LaCova does not include label correlations when fitting the model, and therefore a first-order label correlation model is fit. However, if the sum of absolute covariances at the root node exceeds λ, the observations in the root node are split into two disjoint nodes. The splitting criterion considers pairwise interactions between labels. Therefore, if the two resulting nodes are not split again due to low label variance sums, or if the two nodes each apply a vertical split, LaCova fits a second-order label correlation model. Lastly, if at least one of the two child nodes of the root node apply a horizontal split, LaCova implicitly includes higher-order label correlations in the model. The motivation for this is the following. Since the label covariance matrix of each node is calculated using the label vectors of the observations at the node, the matrix is dependent on the splitting rule that sent those specific observations to the node. The covariance matrix of a node is therefore calculated conditional on the previous split. In this work, this is recognised as an implicit way of including higher-order label correlations in the model.

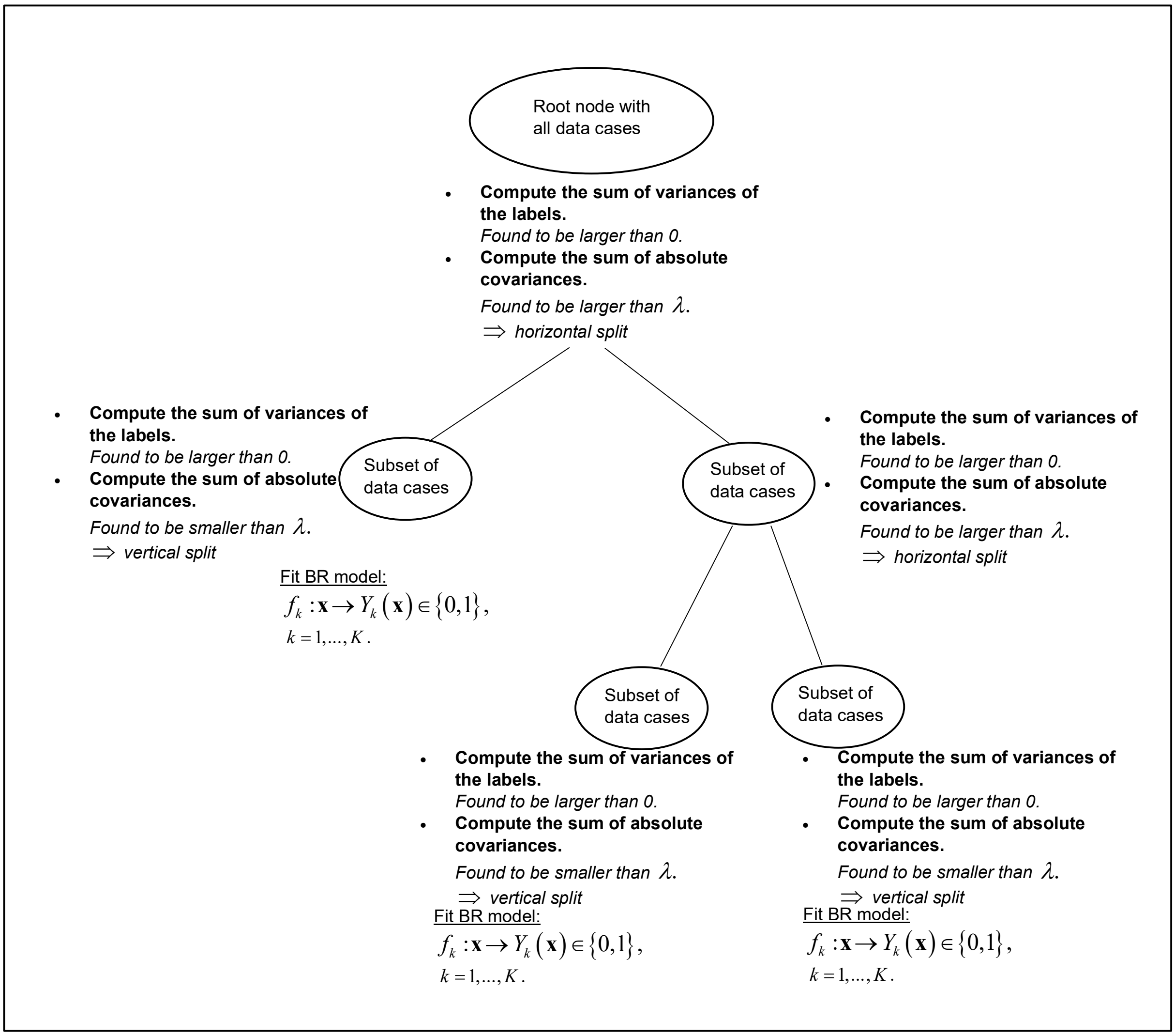


Figure 2.9 Example of a fitted LaCova tree

- **SVM-based decision trees for multi-label learning (ML-SVMDT)**

Gjorgjevikj *et al.* (2013) introduce a hybrid tree for multi-label data. The model applies ML-C4.5 to split the training observations and uses SVMs as base classifiers when solving partial binary classification problems.

Initially the training data are split into a training subset, S^{train}, and validation subset, S^{val}. The fitting procedure consists of three phases.

In the first phase an ML-C4.5 tree (as described above) is fit to S^{train}. Three stopping criteria are used by Gjorgjevikj *et al.* (2013), namely that a node becomes terminal if all observations in the node have the same labelset, no partitioning provides any information gain, or that the number of observations in any of the child nodes after the split would be below some predetermined minimum. Each terminal node of the fitted ML-C4.5 tree is assigned a labelset based on the majority vote per label for the training observations in the node.

After the ML-C4.5 tree has been fit, it is pruned back in the second phase by using S^{val}. This is done by allowing the set of observations in S^{val} to filter down the fitted ML-C4.5 tree until each observation reaches some terminal node. For each node of the ML-C4.5 tree, two quantities are estimated using the corresponding validation observations found in the node. The first of these quantities is referred to as “leaf error”, which is an estimate of classification error if the current node had been a terminal node. The second quantity is referred to as “tree error” and gives the weighted sum of the estimates of classification error of all sub-nodes of the current node. Any multi-label evaluation measure can be used to estimate the classification error of a node. If the tree error exceeds the leaf error of a node, this node becomes a terminal node.

In the third phase BR with an SVM base classifier is fit to all terminal nodes of the pruned ML-C4.5 tree. These local models are fit using the S^{train} observations found in the respective nodes. The classification error of each of these local models can be estimated by using the validation observations found in the terminal nodes. The final ML-SVMDT will not necessarily have a local model at each terminal node. Local model replacement only takes place if the local model error is smaller than the leaf error of the terminal node. In this case, the S^{train} and S^{val} observations in the node are joined first, after which a new local model is fit that is learned on the combined set of observations. This new local model replaces the previous local model of the terminal node to increase the predictive performance of the local models.

A new observation obtains a multi-label classification by moving through the nodes of the ML-SVMDT by following the splitting rules until it ends in a terminal node. If the node contains a local model, this model is used to classify the observation, else the multi-label classification assigned to the node defines the classification of the observation.

Since an ML-SVMDT is fit by initially splitting the training observations based on ML-C4.5, the same arguments with regards to label correlations hold as given for ML-C4.5.

- **Hierarchy of multi-label learners (HOMER)**

Initially presented in Tsoumakas *et al.* (2008), the well-known method HOMER aims to transform a multi-label classification task into a tree-shaped hierarchy of simpler multi-label classification tasks that deal with a small number of labels per node.

A tree-shaped label hierarchy is formed by recursively partitioning the original labels of a multi-label dataset into smaller disjoint subsets which form the nodes. The labels are partitioned by means of clustering. In Papanikolaou *et al.* (2018) balanced $k-$means is used, but the procedure allows for the implementation of other clustering methods as well. Similarity of labels is based on co-occurrence of training observations. More precisely, if the multi-label data is given by $\begin{bmatrix} \underset{N\times p}{\mathbf{X}} & \underset{N\times K}{\mathbf{Y}} \end{bmatrix}$, the hierarchy of labels is formed by successively clustering the K column-vectors of $\mathbf{Y}$ into increasingly smaller clusters. The process continues until every label ends up on its own, or until a prescribed minimum node cluster size is reached. The set of labels making up a cluster at Node n is referred to as the meta-label, M_n. The full input dataset, $\mathbf{X}$, is also partitioned into (possibly overlapping) subsets.

The subset of data cases for which at least one of the labels in M_n is present, is denoted by D_n. The recursive label clustering procedure therefore determines how training observations are partitioned into sub-nodes. Once the hierarchy has been constructed, a local multi-label classification model is fit at every node considering the meta-labels, input variables and the corresponding data subset.

As an example, consider a dataset with $K = 12$ labels at the root node of a HOMER label hierarchy and suppose the labels are split into two disjoint clusters. The resulting meta-labels are, for example, $M_1 = \{Y_1, Y_2, Y_7, Y_9, Y_{10}, Y_{12}\}$ and $M_2 = \{Y_3, Y_4, Y_5, Y_6, Y_8, Y_{11}\}$, as indicated in Figure 2.10. The dataset is also partitioned into two (possibly overlapping) subsets, D_1 and D_2, where D_1 represents the subset of data cases for which at least one of the labels in M_1 is present and a similar interpretation holds for D_2. Now the labels in M_1 are clustered once more, forming meta-labels M_3 and M_4 with corresponding data subsets of D_1 denoted by D_3 and D_4. Similarly M_2 is clustered into meta-labels M_5 and M_6, while D_2 is partitioned into data subsets D_5 and D_6. As seen in Figure 2.10, the process continues until every label ends up on its own.

Local multi-label classifiers are fit at the nodes of a HOMER-tree using the relevant meta-labels and corresponding data subsets. Using Figure 2.10 as an example, the first local multi-label dataset has M_1 and M_2 as labels and each data case in $\mathbf{X}$ is annotated with one, both or none of these labels, depending on if the observation is present in D_1, D_2, both D_1 and D_2, or do not have any labels present. Similarly M_3 and M_4 are used as labels for the second local multi-label dataset and each observation in D_1 is annotated with one or both of these labels depending on if the observation is present in D_3, D_4, or found in both D_3 and D_4.

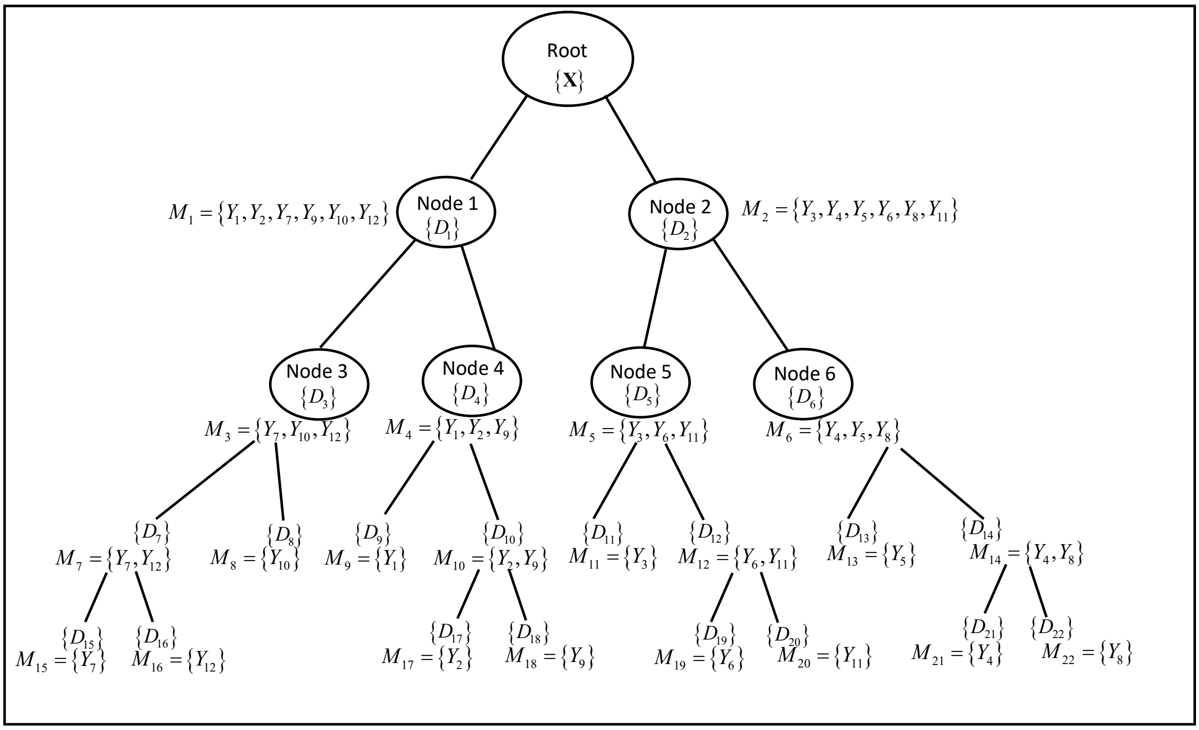


Figure 2.10 Example of HOMER tree-shaped hierarchy

After all the local multi-label classifiers have been fit, a new data case can undergo classification. This would entail the data case to be classified into none, one, or more of the meta-labels at each of the nodes using the local multi-label classifiers. Suppose for example at the root node of the HOMER tree in Figure 2.10, the multi-label classifier assigns a value of 1 to M_1 and 0 to M_2. The implication is that this observation is classified to have the labels in M_2 all absent, while the classifications for labels in M_1 are still to be determined. Further down the tree only the nodes corresponding to the labels in M_1 are considered. Suppose at Node 1 a value of 1 is assigned for both meta-labels, $M_3 = \{Y_7, Y_{10}, Y_{12}\}$ and $M_4 = \{Y_1, Y_2, Y_9\}$. This implies that the multi-label classifiers of both Node 3 and Node 4 should be considered next. If these classifiers respectively assign a 0 for M_7 and M_{10}, and a 1 for M_8 and M_9, the final multi-label classification for the new case is given by $[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]$.

As seen with the above example, the design of HOMER gives large importance to the first clustering split of the labels. The fact that the local multi-label classifier assigned a 0 to M_2 resulted in an immediate final classification of 0 for all the labels in this meta-label. If the local multi-label classifier would have also assigned a 0 to M_1, this would have immediately resulted in a final classification of 0 for all the labels of the dataset. The other local multi-label classifiers deeper down the tree would not even be considered for the observation.

Explicit higher-order label correlations are considered within HOMER since the tree-shaped hierarchy is formed in a recursive manner with clustering based on co-occurrence of labels. Furthermore, HOMER implicitly deals with the multi-label imbalanced problem since imbalance will be less severe for a meta-label than for a single label. It is also particularly efficient for datasets with many labels (Papanikolaou *et al.*, 2018). However, it has been found that HOMER does not reach its full potential when fit to datasets with a small to moderate number of labels (Bogatinovski *et al.*, 2022).

2.7.4 Tree-based ensemble methods

In this work, multi-label tree-based ensemble methods are defined as learning methods that form an ensemble of trees where each of the tree base classifiers give a multi-label classification. This section refers to some of these methods found in the literature.

- **Random Forests of ML-C4.5 (RFML-C4.5) and Random Forests of Predictive Clustering Trees (RF-PCT)**

Two well-known ensemble methods often considered in the context of single-label decision trees are bagging and random forests. For bagging, each classifier in the ensemble is trained on a different bootstrap sample taken from the training data. Similar to bagging, when fitting a random forest each classifier is also trained on a bootstrap sample, however additional diversity among classifiers is obtained by sampling candidate input variables from the input set. Specifically, if a decision tree is used as base classifier, at each node only a random subset of input variables is considered as candidates for splitting the data. Kocev *et* *al*. (2007) explain how bagging and random forests can be extended to multi-label data by using a multi-label tree as base classifier. This has led to the development of Random Forests of ML-C4.5 (RFML-C4.5) and Random Forests of MODT, more commonly referred to in multi-label literature as Random Forests of Predictive Clustering Trees (RF-PCT).

- **Sparse Oblique Structured Hellinger Forests (SOSHF)**

Daniels and Metaxas (2017) propose SOSHF by extending structured forests, a type of random forest used for structured prediction, to multi-label data. The structured random forests are based on the proposal by Dollár and Zitnick (2013) developed for structured learning. Structured learning addresses the problem of learning a mapping where the input or output space may be arbitrarily complex, for example represented by sequences or graphs. In the work by Dollár and Zitnick (2013), only the output space is structured. Daniels and Metaxas (2017) argue that the higher-order correlations between labels of multi-label data represent a special "structure", and that structured forests can learn how to identify and utilise these correlations.

In general, the structured forests of Dollár and Zitnick (2013) differ from traditional random forests only in how the splitting functions are learned. At each node during training, structured forests learn a transformation that maps the multiple structured labels to a single discrete label, so that a standard single-label-based splitting criterion can be optimised. The mapping is two-stage. The first stage entails mapping the structured labels of the label space, Y, to an intermediate space, Z, where distance is easily measured. This first mapping of $\mathrm{Y} \to Z$, is followed by a second mapping of $Z \to C$, by using $k-$means clustering for example. Here C denotes the cluster space that defines the single discrete label.

SOSHF aims to address imbalance within multi-label data. Therefore, Daniels and Metaxas (2017) modify the clustering step of the structured forests of Dollár and Zitnick (2013) by using cost-sensitive clustering to address between- and within-label class imbalance.

Instead of directly applying the clustering on the label-space, each label is assigned a corresponding cost and weighted $k-$means clustering is performed.

The weight of label Y_A is given as:

$$IDF(A) = \beta \frac{\log\left(1+\frac{N_g}{n_{g\,A}}\right)}{\max_j \log\left(1+\frac{N_g}{n_{g\,j}}\right)} + (1-\beta)\frac{\log\left(1+\frac{N_l}{n_{l\,A}}\right)}{\max_j \log\left(1+\frac{N_l}{n_{l\,j}}\right)}.$$

In the above expression N_g denotes the total number of training observations, while N_l denotes the total number of training observations at the node. The total number of all observations that have Y_A present is denoted by $n_{g\,A}$, and the number of observations that have Y_A present at the current node is denoted by $n_{l\,A}$. The parameter $\beta \in [0,1]$ is used to find a compromise between the importance of global imbalance (considering all observations) and imbalance at the current node. Daniels and Metaxas (2017) found that $\beta = 0.5$ works well. Since $IDF(A)$ increases the importance of the minority class for each label, the impact of within-label imbalance is reduced. Furthermore, since $IDF(A)$ gives more importance to labels with significant imbalance problems, it adjusts for between-label imbalance as well.

By setting $k = 2$ when performing weighted $k-$means clustering at a node, the clustering step transforms the original multi-label problem into a single-label problem. However, even when applying the above cost-sensitive clustering approach, the newly formed single-label problem is often still imbalanced. Consequently, Daniels and Metaxas (2017) propose using a splitting criterion that is well suited for imbalanced single-label data, such as Hellinger distance. It has however been empirically observed that Hellinger Distance Decision Trees (HDDT) produce deeper trees. This is because HDDT more finely partition the data. To account for this, Daniels and Metaxas (2017) propose using oblique trees.

Different from standard trees that find axis parallel hyperplanes, each node of an oblique tree contains a hyperplane that can take any orientation in input space. An exhaustive search for the best hyperplane at each node is often not feasible, therefore approximate solutions are used. SOSHF uses a first-order gradient-based method to perform the optimisation at a node. A differentiable loss function that approximately maximises the squared Hellinger distance is constructed. For a detailed description of the derivation of this loss function and the optimisation per node, refer to Daniels and Metaxas (2017).

The training observations of a terminal node are used to calculate the labelset assigned to the node. This is done using the original label indicator variables of the observations. To classify new observations per tree, the observations are assigned the labelsets of the terminal nodes in which they end.

When fitting the forest of trees, Daniels and Metaxas (2017) randomly sample 75% of the input variables, training observations and labels, without replacement per tree. Every tree predicts all the labels simultaneously, even though all labels are not used for fitting each tree. In Daniels and Metaxas (2017), if a label is used in learning a tree, that tree's prediction for that label is weighted five times higher than for trees where the label is not used in the training procedure.

- **Multi-Label Forest (ML-Forest)**

Wu *et al.* (2016) propose ML-Forest which learns an ensemble of hierarchical multi-label tree classifiers. Each tree is learnt on a bootstrap sample of the training data.

The root node of such an ML-Forest tree contains all the data. Disjoint child nodes sprout from the root node by recursively splitting the data. Wu *et al.* (2016) refer to this splitting process as SplitTest, which is implemented as follows.

Initially, BR is applied to the observations in the root node. This gives K binary classifiers (one for each label). K confidence scores are calculated for each observation in the root node by using the K fitted classifiers. Each observation is then classified to the class with maximum confidence score. If the maximum confidence score corresponds to more than one class, the observation is classified to the class with the largest prior. This produces the first level of the tree with potentially as many as K child nodes. A label purity vector, $\mathbf{p} = \left[p^1, p^2, \dots, p^K \right]$, is calculated for each of these child nodes. Here $p^k \in [0,1]$ denotes the purity of the node with respect to the k^{th} label calculated as $p^k = \frac{1}{|D|} \sum_{\mathbf{x}_i \in D} y_i^k$, where D gives the collection of observations at the node and $|D|$ the number of observations at the node. The purity vector of a node is used to calculate a relevant label vector for the node, $\mathbf{b} = \left[b^1, b^2, \dots, b^K \right]$, with

$$b^k = \begin{cases} 1 & if \;\; p^k \geq \lambda \\ 0 & otherwise \end{cases}$$

the relevant label indicator for the k^{th} label and $\lambda \in (0.5\,,1)$ a purity threshold. Wu *et al.* (2016) select λ at random in the range $(0.5,1)$ for each tree in the ensemble.

Wu *et al.* (2016) make use of a label transfer mechanism to recursively propagate the relevant label vector to lower levels of the tree. This preserves the relevant label vector of a parent node and incorporates it as an additional indicator when forming the relevant label vector of the child node. In other words, once a label purity vector, $\mathbf{p}$, of a node is transformed to a relevant label vector, $\mathbf{b}$, this relevant label vector is updated using the relevant label vector of the node's parent node by letting

$$b^k_{updated} = \begin{cases} 1 & if \;\; b^k_{parent} = 1 \;\; or \;\; b^k = 1 \\ 0 & otherwise \end{cases} .$$

Note that the root node has $\mathbf{b} = \mathbf{0}$.

Hereafter each node is potentially split into child nodes by applying BR at the node using only those labels with $b^k_{updated} = 0$. In other words, the relevant labels identified for parent nodes are used as priors to reduce the label space when splitting child nodes. Once again, each data observation moves to only one child node corresponding to the maximum confidence score achieved considering the classifiers of the BR model. This splitting process continues until the data cannot be further split by the induced classifiers, thus forming a set of terminal nodes that are each associated with an (updated) relevant label vector.

To classify an unseen data observation, $\mathbf{x}_{new}$, this observation is dropped into each tree in the ensemble and reaches some terminal node within each. Suppose the ensemble consists of M trees. Since each terminal node has a corresponding relevant label vector, $\mathbf{x}_{new}$ obtains M relevant label vectors, $\mathbf{b}_1,...,\mathbf{b}_M$. A confidence value, c^k, is calculated for each of the K labels as $c^k = \frac{1}{M}\sum_{j=1}^{M} b_j^k$. To obtain a final multi-label classification, conclude that the k^{th} label is present for $\mathbf{x}_{new}$ if c^k exceeds a predefined threshold value, t_{Y_k}.

With their approach, Wu *et al.* (2016) argue that the relevant labels at higher levels of ML-Forest trees capture more discriminable label concepts which are then transferred to child nodes that are harder to evaluate. Furthermore, since learning models at different levels of the tree work together to reveal multiple label concepts, ML-Forest implicitly exploits label correlations and is regarded as a higher-order label correlation method (Wu *et al.*, 2016).

Since nodes of an ML-Forest tree are potentially split into many child nodes, a shallow tree with very few training data cases found in some nodes may easily arise. This poses a risk of overfitting, especially in cases with large K. Wu *et al.* (2016) comment on the risk of overfitting and therefore suggest fitting a bagged ensemble of trees. It is unclear if this strategy alone sufficiently prevents overfitting. Furthermore, little information is given by Wu *et al.* (2016) regarding the stopping criteria of an ML-Forest tree. Wu *et al.* (2016) provide readers with a link which they claim lead to the code of ML-Forest; however, this link opens the homepage of one of the authors and the code is unfortunately not openly available.

2.7.5 Other ensemble methods

Multi-label ensemble methods are not limited to ensembles of tree-based classifiers. Other ensembles have also been introduced that are based on common problem transformation methods or algorithm adaptation methods. This section highlights three ensemble methods often considered in the multi-label literature. These methods are also related to LDsplit (proposed in Chapter 3).

- **Ensemble of Binary Relevance (EBR)**

Read *et al.* (2011) consider an ensemble of BR models (EBR). Each BR model in the ensemble is trained on a bootstrap sample taken from the training data.

To classify a new data case, each BR model produces a multi-label classification for the data case as outlined in Section 2.7.2. Hereafter the mean of the binary predictions over the ensemble is found per label. This gives one confidence output per label. A label is considered present for the data case if the confidence output of the label exceeds a predefined threshold value.

Even though EBR can improve on BR due to the diversity among base classifiers, label correlations remain unexploited so that EBR is a first-order label correlation method (Moyano *et al.*, 2018).

- **Ensemble of Classifier Chains (ECC)**

Read *et al.* (2011) extend CC (as described in Section 2.7.2) to an ensemble of differently ordered classifier chains referred to as Ensembles of Classifier Chains (ECC).

Instead of considering only one order of the labels to fit one CC model, ECC considers M random orders of the K labels. For each of these M random chain orders a single CC model is trained using a bootstrap sample of the training data. Read *et al.* (2009) initially suggested using a subset of the training data to fit each classifier, however Read *et al.* (2011) state that a bagging approach can achieve higher predictive performance with only a small increase in computation cost.

Some label orders may produce poorer CC models in terms of prediction accuracy than other orders. However, determining an optimal order is a difficult task. Instead of running the risk of randomly selecting a single poor CC model, Read *et al.* (2011) argue that by implementing an ensemble of CC models, the overall negative impact of poorer orders is reduced and the difficult task of determining a single optimal order is obviated.

To classify a new data case, $\mathbf{x}_{new}$, the M CC models each give a multi-label classification for $\mathbf{x}_{new}$ as outlined in Section 2.7.2. The mean of the binary predictions over the M classifications is found per label. This gives a single confidence output per label. A label is considered present for $\mathbf{x}_{new}$ if the confidence output of the label exceeds a predefined threshold value.

- **Random $k-$labelsets (RAkEL)**

Tsoumakas *et al*. (2011a) propose RAkEL, an ensemble method which attempts to improve on the limitations of LP. RAkEL divides the multi-label data into smaller multi-label datasets by considering random subsets of the labels. Two different versions of RAkEL exist, depending on if the label subsets are disjoint or overlapping: RAkEL$_d$ and RAkEL$_o$.

For RAkEL$_d$, $M = \lceil \frac{K}{k} \rceil$ disjoint subsets of size k are formed from the K labels of a multi-label dataset. Here $\lceil a \rceil$ denotes the ceiling function of a real number a. For example if $a = 3.2$, $\lceil 3.2 \rceil = 4$. Note that if $\frac{K}{k}$ is not an integer, the M^{th} label subset contains $K \bmod k$ labels. Hereafter, M multi-label datasets are formed by separately considering the input observations, $\mathbf{x}_i$, $i = 1,...,N$, with their corresponding label indicator variables of the labels that make up each label subset. Tsoumakas *et al*. (2011a) apply LP to each sub-dataset. This gives rise to M multi-label classifiers that can be used to classify a new observation. In this case, each of the M classifiers give a multi-label classification for the new observation. Since the label subsets are disjoint, the M classifications can simply be gathered to produce a final multi-label classification for all K labels.

In the case of RAkEL$_o$, the M label subsets may overlap. Therefore, M possibly overlapping subsets of size k are formed from the K labels. Hereafter a similar procedure is followed as for RAkEL$_d$. In other words, M multi-label datasets are formed on which LP models are fit, consequently giving rise to M multi-label classifiers for prediction. However, since the label subsets possibly overlap, the M multi-label classifications of a new observation cannot simply be gathered. Instead, the M multi-label classifications produced for the observation are considered and, for each label, the mean of all binary predictions produced for the label is found. If the mean exceeds a predefined threshold value, the label is deemed present for the observation.

2.8 Conclusion

This chapter provided a comprehensive overview of the literature of multi-label data and the current software available in the field.

Summaries were provided for measures of descriptive properties of multi-label data, popular multi-label benchmark datasets, as well as definitions of multi-label evaluation measures. These summaries are referred to in later chapters of this dissertation where empirical studies are conducted.

Furthermore, an outline was provided of the different viewpoints of label correlation in multi-label literature. It is generally agreed that exploitation of label correlation is a desirable property of a multi-label classification method. However, as a result of the different viewpoints of label correlation exploitation, a wide range of methods exist that exploit label correlations in some way. Nonetheless, most existing multi-label learning methods focus on global label correlation exploitation. Since decision trees divide data into subgroups by taking local decisions, an appropriate multi-label tree-based classifier may be able to effectively exploit local label correlations.

Several multi-label learning methods were discussed in this chapter. This included a detailed discussion of previously proposed multi-label tree-based methods and multi-label tree-based ensemble methods. A new tree-based ensemble method for multi-label classification is presented in the next chapter, namely LDsplit. As will become clear, LDsplit differs from the previously proposed multi-label tree-based methods discussed in this chapter in several ways.

Chapter 3: Label-Dependent splitting (LDsplit)

3.1 Introduction

A new tree-based ensemble method for multi-label classification is proposed in this chapter. This method uses a divide-and-conquer strategy by constructing an ensemble of tree-structures that are based on different label subsets. With each tree-structure, the aim is to split the data in a label-dependent way, thereby implicitly incorporating local label correlations. The multi-label classification method is therefore referred to as Label-Dependent splitting (LDsplit). Considering the three categories of multi-label classification methods based on the order of the label correlations (Zhang and Zhang, 2010), LDsplit is a higher-order method.

A detailed description of the fitting and classification procedure of LDsplit is given in the following sections. In general, an LDsplit tree-structure consists of levels, each containing a collection of nodes. A different label is used at each level of a tree-structure, and each node is split by considering a binary classification problem. However, a specific level is not only dependent on its label; it also depends on previous splits made when parent nodes were formed using other labels. The main contribution of the strategy is not only that possible shared information and higher-order label correlation are implicitly exploited locally by dividing the data into subgroups, but that it is done in a simple and clear-cut manner by using hierarchical tree-structures of labels. Any off-the-shelf binary classifier can be used to perform the splitting. Binary classifiers may even be selected per label. This contributes to the flexibility of the method.

Since LDsplit tree-structures are dependent on the label ordering used to construct the different levels of the trees, two different ordering strategies are implemented in this work: Random LDsplit and Conditional LDsplit. The simplest of these is Random LDsplit which uses a random label ordering for the construction of each tree-structure. Conditional LDsplit on the other hand includes a theoretically motivated strategy that determines a set of label orders for the tree-structures.

For the initial description of the LDsplit methodology, Random LDsplit is introduced in the next section. This includes a detailed discussion of the fitting and classification procedure of Random LDsplit. The discussion is extended to Conditional LDsplit in Section 3.3. Scaling aspects of LDsplit for large K is considered in Section 3.4. Section 3.5 gives a short note on fitting LDsplit in R. To further highlight the contribution of this work, distinctive and favourable properties of LDsplit, as opposed to some related methods, are given in Section 3.6.

3.2 Random LDsplit

Consider a multi-label dataset, $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, ..., N\}$, with N $p-$component input observations, $\mathbf{x}_i$, and their corresponding $K-$component multi-label response vectors, $\mathbf{y}_i$. Denote by m a positive integer smaller than or equal to K. Recall that the collection of all labels, $Y_1, Y_2, ..., Y_K$, is denoted by L and denote by L^m the set of all $m-$permutations of the elements in L. The size of L^m is therefore $\frac{K!}{(K-m)!}$. Specification of the tuning parameter, m, is discussed in Section 3.2.4. Before LDsplit is implemented, each label in L is also allocated a classification threshold, $t_{Y_1}, t_{Y_2}, ..., t_{Y_K}$ - more on this later.

To implement Random LDsplit, start by randomly sampling from L^m, without replacement, M label permutations, P_j, $j = 1, 2, ..., M$. M therefore denotes the size of the LDsplit ensemble. If $M = \frac{K!}{(K-m)!}$, all the permutations in L^m are selected. Specifying a value for the tuning parameter, M, is also discussed in Section 3.2.4. Each permutation, P_j, has m entries, denoted by $P_{j,s}$, $s = 1, 2, ..., m$, corresponding to m of the labels in L. For each of the M permutations a tree-structure, T_j, $j = 1, 2, ..., M$, is constructed as follows.

3.2.1 Fitting tree-structure, T_j

The tree-structure, T_j, for a given permutation, P_j, is constructed by first forming the root node of the tree. The root node consists of all N input observations, $\mathbf{x}_i$, along with the N corresponding response entries of label $P_{j,1}$, *i.e.* the first label of the permutation. This node is referred to as Node 1. Note that the data in Node 1, $\{(\mathbf{x}_i, y_i), i = 1, 2, ..., N\}$, represent a binary classification dataset, and that any binary classifier can be fit to these data. By doing so each $\mathbf{x}_i$ in Node 1 is assigned a posterior probability for label $P_{j,1}$ as given by the fitted binary classifier. These posterior probabilities can be used to split $\{\mathbf{x}_i, i = 1, 2, ..., N\}$ into two disjoint groups, called Node 2 and Node 3. Observations for which the posterior probabilities are less than $t_{P_{j,1}}$, move down the left branch of the tree and form Node 2. Similarly, observations for which the posterior probabilities of label $P_{j,1}$ exceed $t_{P_{j,1}}$, move down the right branch of the tree and form Node 3. This first splitting of the data cases forms Level 1 of the tree.

Continuing in this way, another binary classification problem is created by all the $\mathbf{x}_i$ cases found in Node 2 along with their corresponding response entries for the next label in permutation P_j, *i.e.* label $P_{j,2}$. Again, a binary classifier can be fit and used to split the data cases of Node 2 into two disjoint nodes by using $t_{P_{j,2}}$. A similar procedure is applied to the data cases in Node 3 and their response entries for label $P_{j,2}$. This second series of splitting gives the four nodes in Level 2.

As a simple example consider a multi-label dataset with $N = 10$ observations and $K = 6$ labels, $Y_1, Y_2, ..., Y_6$. The example-data are summarised in Table 3.1. By setting $m = 3$, L^3 gives the collection of all label permutations of size $m = 3$. Suppose the permutation $[Y_3 \; Y_6 \; Y_1]$

is sampled from L^3. Figure 3.1 gives an example of an LDsplit tree-structure constructed from this permutation. Since Y_3 is the first label of the permutation, in Node 1 a binary classifier is fit to all the data cases and their corresponding response entries of Y_3. The observations are then split into Node 2 and Node 3 based on the posterior probabilities given by the fitted binary classifier. Two separate binary classification problems are formed at Level 1 by considering the observations found in Node 2 and Node 3 along with their corresponding response entries for the second label in the permutation, *i.e.* Y_6. A binary classifier is fit to each node and the data are split based on the posterior probabilities so that Level 2 is formed.

Table 3.1
Summary of a simple multi-label dataset with $N = 10$ data cases and $K = 6$ labels

Data cases	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
$\mathbf{x}_1$	0	1	1	0	1	0
$\mathbf{x}_2$	1	0	0	0	1	1
$\mathbf{x}_3$	0	1	1	1	0	0
$\mathbf{x}_4$	1	0	1	0	0	0
$\mathbf{x}_5$	1	1	0	0	1	1
$\mathbf{x}_6$	0	0	1	1	0	0
$\mathbf{x}_7$	1	1	0	0	0	1
$\mathbf{x}_8$	1	1	1	0	0	0
$\mathbf{x}_9$	0	0	0	1	0	1
$\mathbf{x}_{10}$	1	0	0	1	1	0

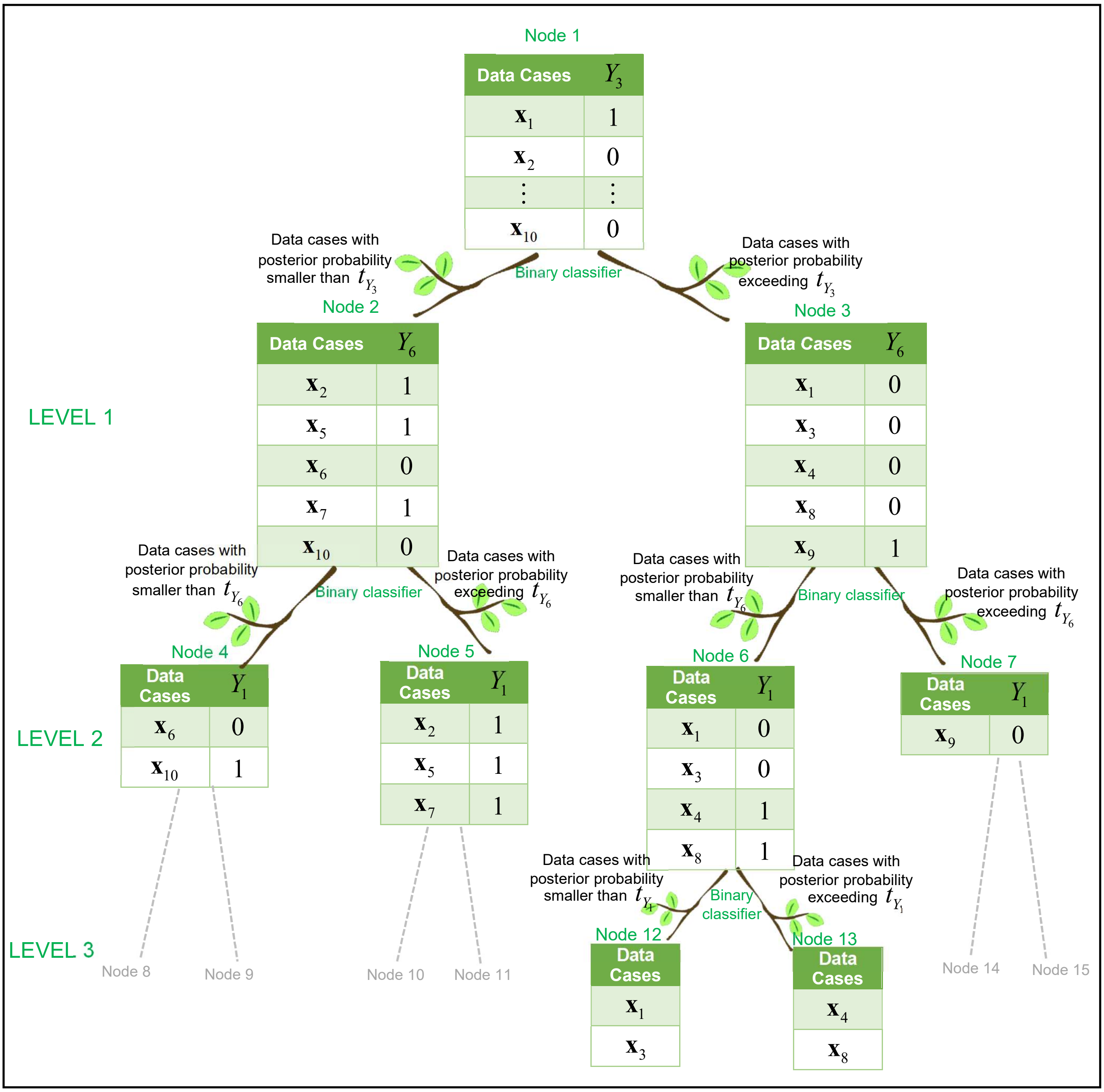


Figure 3.1 Example of an LDsplit tree-structure with three levels

In the simplest case, splitting continues until Level m is reached. In other words, splitting continues until each node of Level $m-1$ forms a binary classification problem with $P_{j,m}$ and splits into two new nodes by using $t_{P_{j,m}}$. However, two constraints must be added.

Firstly, a binary classifier is only fit to a node if the data at the node displays impurity regarding the relevant label, *i.e.* there are observations at the node that have $y_i = 1$ as well as observations that have $y_i = 0$, so that both classes are represented. If this requirement is not met, the node is pure and becomes a terminal node, with no further splitting. An example of such a terminal node in Figure 3.1 is Node 5.

Secondly, it may happen that all the observations in a node move to only one of the resulting sub-nodes, so that the other sub-node is empty. It is evident that a binary classifier cannot be fit to an empty node. In addition, it may also happen that some splits cause a very small number of observations to be present in one (or both) of the resulting sub-nodes. Even though it is possible to fit a binary classifier to nodes containing few observations, the resulting classifier may be unstable and not very informative. For these reasons, a minimum node-size, denoted by n, is chosen before the M tree-structures are fit. When constructing a tree-structure, binary classifiers are only fit to nodes of size more than n. If this threshold size is not exceeded, the node becomes terminal. For simplicity, $n = 2$ in Figure 3.1, therefore Node 4 is an example of a node that is terminal because its size did not exceed n. Note that Node 7 is also a terminal node since it is both pure and the node-size does not exceed n.

3.2.2 Algorithm

By using the procedure of Section 3.2.1, LDsplit fits M tree-structures. Each tree-structure consists of a set of numbered nodes at each of its m levels. In general, at Level s, the nodes numbered 2^s , 2^s+1 , ... , $2^{s+1}-1$ are present, resulting in 2^s nodes at this level. If splitting continues along every branch, a complete tree-structure has a total of $2^{m+1}-1$ nodes after a total of 2^m-1 splits have been made. To avoid confusion in this work, even if a node becomes terminal at one of the levels preceding Level m, the numbering of the nodes does not change. Nodes that would have sprouted from a node that became a terminal node at an earlier stage, keep their numbers and are simply regarded as being empty. The total number of splits will however be less than 2^m-1 in this case. For example, the terminal nodes of Figure 3.1 are Node 4, Node 5, Node 7, Node 12 and Node 13. Therefore Node 4, Node 5 and Node 7 are examples of terminal nodes that occur on a level preceding Level m (in this case $m=3$). Node 8, Node 9, Node 10, Node 11, Node 14 and Node 15 however keep their numbers even though they are regarded as empty, since they would have sprouted from Node 4, Node 5 and Node 7 had these nodes not been terminal.

The Random LDsplit algorithm is outlined in Algorithm 3.1.

Algorithm 3.1

Random LDsplit algorithm

The following algorithm summarises fitting M $m-$level tree-structures T_j, $j=1,2,...,M$, using Random LDsplit.

1) Specify $m \le K$.
2) Sample M times without replacement from L^m, producing P_j, $j=1,2,...,M$.
3) For $j=1,2,...,M$:
   a) Producing Level 1: Fit a binary classifier to all observations $\mathbf{x}_i$, $i=1,2,...,N$, and label $P_{j,1}$. Use this classifier to split the observations into Node 2 and Node 3 based on the threshold $t_{P_{j,1}}$.
   b) For $l=2,3,...,m$:

      Producing level l: Consider each node, η, where $\eta=$node number 2^{l-1}, ... , node number 2^l-1.

      - When node η displays impurity for label $P_{j,l}$ and the number of observations in $\eta > n$, fit a binary classifier to the observations in node η along with their corresponding response entries for label $P_{j,l}$. Use this classifier to split observations based on $t_{P_{j,l}}$.
      - When node η displays purity for label $P_{j,l}$ or the number of observations in $\eta \le n$, no binary classifier is fit to node η and node η does not split. This node then becomes a terminal node.

3.2.3 Classification of new cases

Given a set of (unseen) observations, the M fitted tree-structures, T_j, $j=1,2,...,M$, are used as follows to generate multi-label classifications. The classification procedure resembles that of a binary classification tree, but is adjusted to handle the multi-label design of the fitted tree-structures.

For each tree-structure the new observations are dropped into the root node of the tree and these filter down to the terminal nodes by means of the tree-structure rules. While a given observation moves through the nodes of T_j, it collects a posterior probability for each of the labels $P_{j,s}$, $s=1,2,...,m$, as follows.

During fitting of the model, each node at Level s, $s=1,2,...,m$, of tree-structure, T_j, is assigned a posterior probability for label $P_{j,s}$, by calculating the proportion of training observations in that node that have label $P_{j,s}$ present. When a terminal node occurs at a level preceding Level m, say at Level t, this proportion is also found for the labels $P_{j,t+1},...,P_{j,m}$.

Therefore, an (unseen) observation that ends up in a terminal node at Level r collects the assigned posterior probabilities for labels $P_{j,1},P_{j,2},...,P_{j,r}$ by moving through nodes at Level 1 to Level r. If $r<m$ the posterior probabilities of the remaining labels, $P_{j,r+1},...,P_{j,m}$, are the posterior probabilities obtained from the training observations in the terminal node. It is important to realise that these collected posterior probabilities are not related to the posterior probabilities used to determine how observations move through nodes as based on the tree-structure rules.

Figure 3.2 extends the example of Figure 3.1 by calculating the proportion of training observations in each node that have the appropriate label present. At Level 1 this label is Y_3 while Y_6 and Y_1 are considered for Level 2 and Level 3 respectively. Node 4, Node 5 and Node 7 are terminal nodes that appear before Level 3 (in this case all occurring at Level 2). Therefore, the training observations in these terminal nodes are also used to calculate the posterior probabilities of Y_1 at Level 3. As an example, a new data case that ends up in Node 13 collects the posterior probabilities $\left[0.8 \quad 0 \quad 1\right]$ for labels Y_3, Y_6 and Y_1 respectively, while a data case that ends up in Node 4 collects the posterior probabilities $\left[0.2 \quad 0 \quad 0.5\right]$.

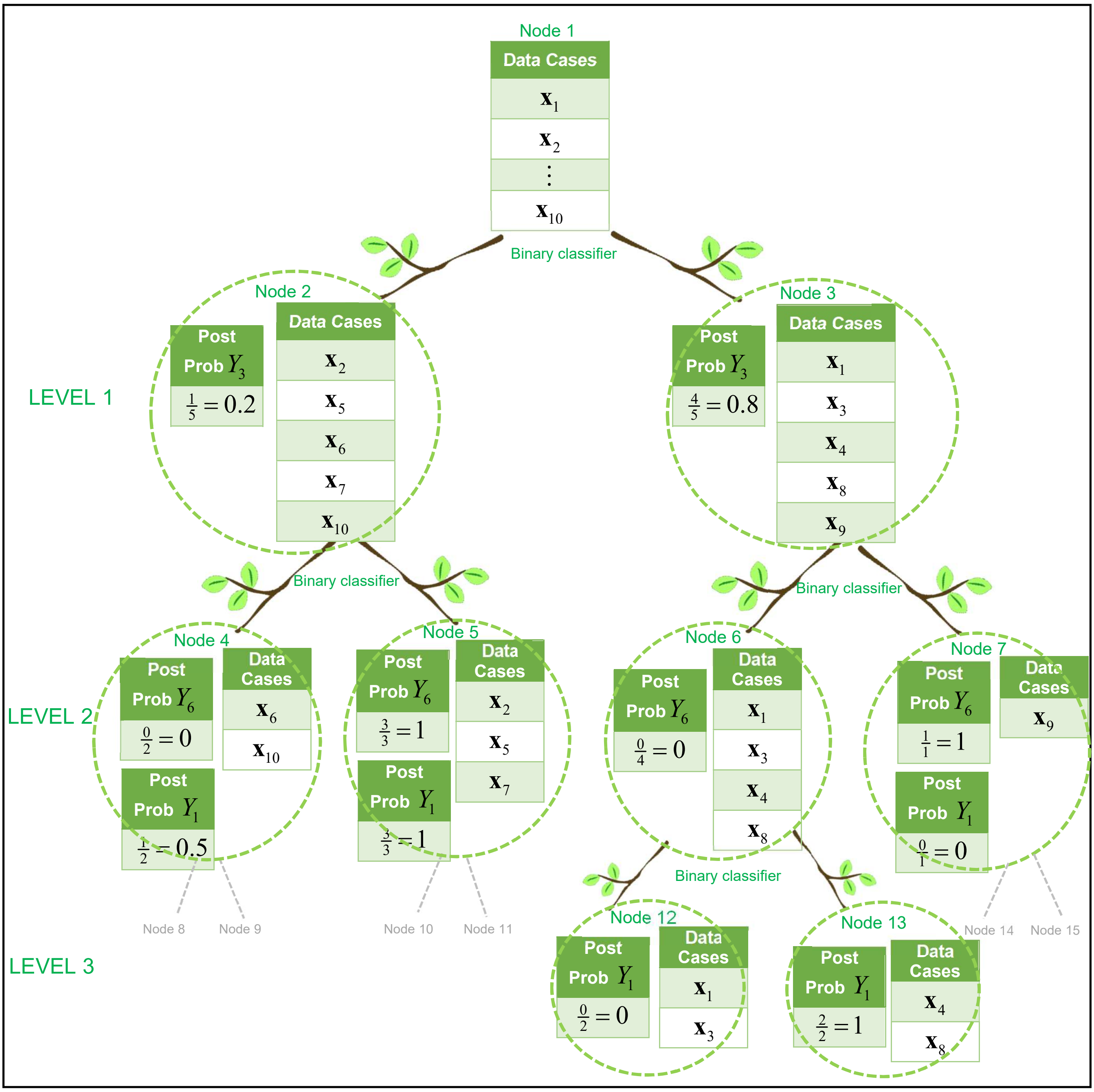


Figure 3.2 Example of an LDsplit tree-structure used for classification

Note that in some cases during training, splitting a node may cause all the observations in a node to move to only one of the two sub-nodes. Due to the LDsplit constraints, the corresponding empty node becomes a terminal node. However, when assigning posterior probabilities to all existing nodes, there are no training observations in this terminal node to base the posterior probability calculations on. Nevertheless, during classification, the binary model of such a split may send some new observations to the node that is empty in training.

Though there may be different ways of handling this, in this work the training observations in the parent node of such an empty terminal node are used to calculate the posterior probability assigned to the node. Figure 3.3 illustrates this using a three-level tree-structure constructed for the permutation $\left[Y_4 \ Y_2 \ Y_6\right]$. The simple example data summarised in Table 3.1 are once again used to fit the tree-structure. In this case $n = 3$.

The tree-structure in the left panel of Figure 3.3 shows which data cases are found in which nodes based on the fitted binary classifiers and label thresholds. The figure shows that the binary classifier fit to Node 3 finds posterior probabilities smaller than t_{Y_2} for all the data cases in the node. Therefore, even though a binary classifier is fit to Node 3, all training observations move to Node 6 making Node 7 an empty terminal node. The tree-structure in the right panel of Figure 3.3 gives the posterior probabilities assigned to each node. Since Node 7 is an empty terminal node, the training observations in Node 7's parent node (Node 3) are used to calculate the posterior probabilities assigned to Node 7. Note that this strategy causes Node 6 and Node 7 to have the same assigned posterior probability for Y_2.

After an observation has collected a set of posterior probabilities by filtering through all M tree-structures, this set is converted to a final multi-label classification. Since a tree-structure typically does not provide a classification for all K labels, denote by $R_{k,i}$ the collection of tree-structures that produce a posterior probability for label Y_k for data case $\mathbf{x}_i$, and let $\left|R_{k,i}\right|$ be the number of elements in this collection. Also, let $\hat{T}_j\left(\mathbf{x}_i, k\right)$ be the posterior probability of label Y_k for observation $\mathbf{x}_i$ given by the j^{th} tree-structure. To classify $\mathbf{x}_i$ for label Y_k, calculate $\frac{1}{\left|R_{k,i}\right|} \sum_{j \in R_{k,i}} \hat{T}_j\left(\mathbf{x}_i, k\right)$ and conclude that the label is present if this quantity exceeds t_{Y_k}.

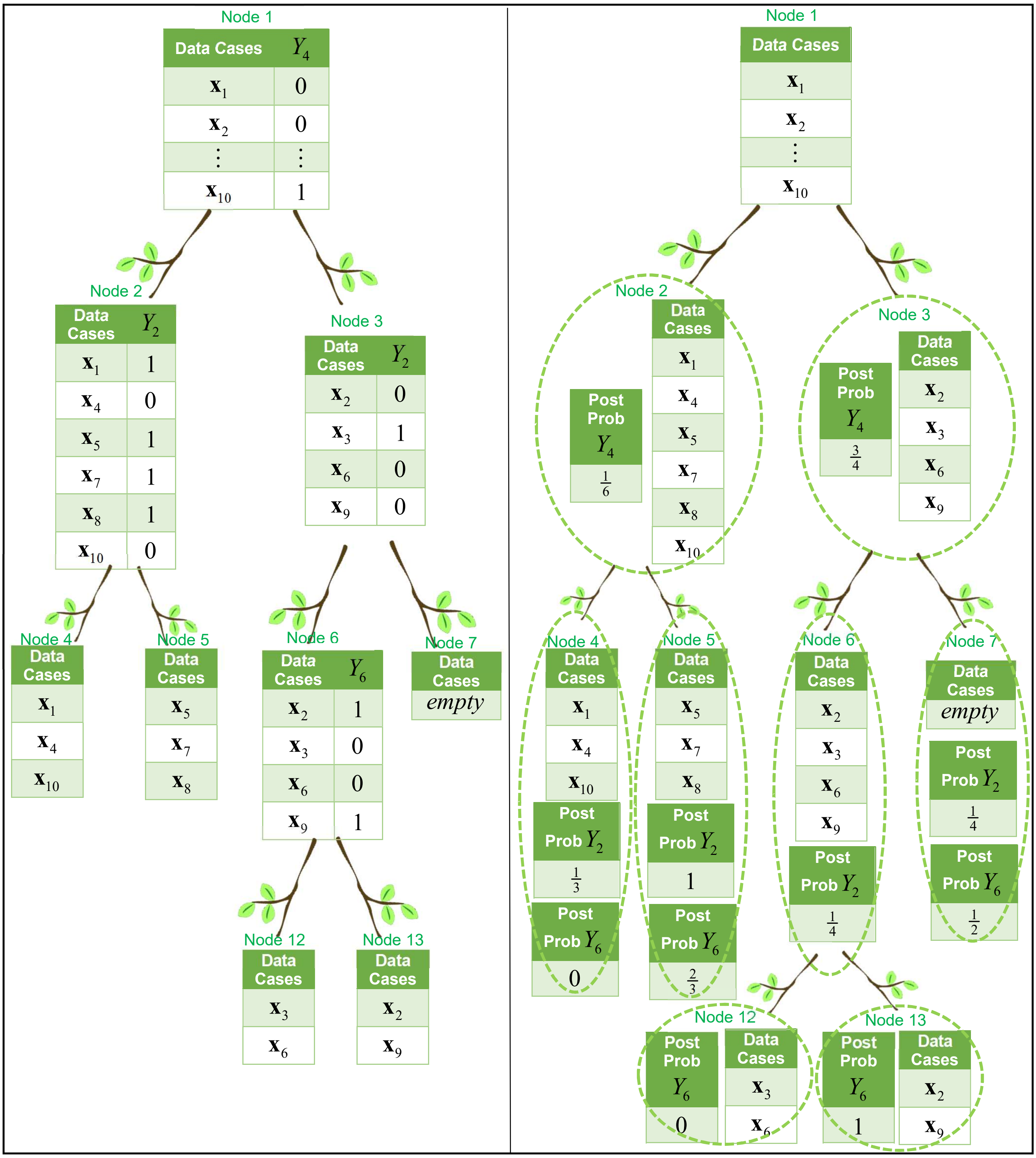


Figure 3.3 LDsplit tree-structure where a split results in an empty node

3.2.4 Tuning parameters

To fit Random LDsplit, one must specify: the number of tree-levels, m, the size of the ensemble, M, a minimum node-size, n, classification thresholds, t_{Y_k}, $k = 1,...,K$, as well as a binary classifier for node-splitting.

For large values of m, more labels are used to construct each tree-structure, fitting larger tree-structures taking more label correlations into account. The maximum number of levels per tree-structure is $m = K$. In this case, each label is represented in each tree-structure. Due to the limitation implied by n, using a large value of m may lead to several terminal nodes occurring on levels preceding Level m. For large m it may even be that the terminal nodes are all at levels preceding Level m and that many levels are empty. Simply using the training observations in the terminal nodes to make classifications for many labels that correspond to empty levels, seems inappropriate. Intuitively, a better strategy may be to fit many smaller tree-structures, setting M large while m is relatively small. The empirical study given in Chapter 4 investigates this hypothesis and provides some guidelines for setting the values of m and M.

The empirical study of Chapter 4 also investigates the classification performance of two binary classifiers, namely an SVM and a standard single-label decision tree. As will be seen in Chapter 4, the SVM classifier yields the best performance for most of the multi-label benchmark datasets considered.

Furthermore, for simplicity $n = 5$ and a classification threshold of 0.5 is used throughout this dissertation. Therefore, the influence of n and t_{Y_k}, $k = 1,...,K$, on classification performance are avenues for future research.

3.2.5 Approximation of multi-label posterior probabilities

Denote the probabilities of the label combinations given the specific input vector, $\mathbf{x}$, by $\mathrm{P}\left(Y_1 = y_1, Y_2 = y_2, \ldots, Y_K = y_K \mid \mathbf{X} = \mathbf{x}\right)$. This section discusses how the posterior probabilities are approximated using an LDsplit tree-structure.

For this discussion, suppose that each label is represented in the LDsplit tree-structure, *i.e.* $m = K$. Therefore, L^K denotes the set of all $K-$permutations of the elements in L. However, the discussion could also be extended for $m < K$. Furthermore, suppose the sampled permutation, P_j, from L^K is denoted by κ for simplicity, *i.e.* $P_j = \kappa$.

The conditional probability of the label combination given an input vector can be written as:

$$\mathrm{P}\left(Y_1 = y_1, Y_2 = y_2, \ldots, Y_K = y_K \mid \mathbf{X} = \mathbf{x}\right)$$

$$= \mathrm{P}\left(Y_{\kappa_1} = y_{\kappa_1} \mid \mathbf{X} = \mathbf{x}\right) \mathrm{P}\left(Y_{\kappa_2} = y_{\kappa_2}, \ldots, Y_{\kappa_K} = y_{\kappa_K} \mid Y_{\kappa_1} = y_{\kappa_1}, \mathbf{X} = \mathbf{x}\right)$$

$$= \mathrm{P}\left(Y_{\kappa_1} = y_{\kappa_1} \mid \mathbf{X} = \mathbf{x}\right) \mathrm{P}\left(Y_{\kappa_2} = y_{\kappa_2} \mid Y_{\kappa_1} = y_{\kappa_1}, \mathbf{X} = \mathbf{x}\right) \mathrm{P}\left(Y_{\kappa_3} = y_{\kappa_3}, \ldots, Y_{\kappa_K} = y_{\kappa_K} \mid Y_{\kappa_1} = y_{\kappa_1}, Y_{\kappa_2} = y_{\kappa_2}, \mathbf{X} = \mathbf{x}\right)$$

$$\vdots$$

$$= \mathrm{P}\left(Y_{\kappa_1} = y_{\kappa_1} \mid \mathbf{X} = \mathbf{x}\right) \prod_{k=2}^{K} \mathrm{P}\left(Y_{\kappa_k} = y_{\kappa_k} \mid Y_{\kappa_1} = y_{\kappa_1}, \ldots, Y_{\kappa_{k-1}} = y_{\kappa_{k-1}}, \mathbf{X} = \mathbf{x}\right).$$

Therefore, based on the above formulation, $\mathrm{P}\left(Y_1 = y_1, Y_2 = y_2, \ldots, Y_K = y_K \mid \mathbf{X} = \mathbf{x}\right)$ can be estimated by multiple steps, with each step for one class.

The LDsplit tree-structure approximates $\mathrm{P}\left(Y_{\kappa_k} = y_{\kappa_k} \mid Y_{\kappa_1} = y_{\kappa_1}, \ldots, Y_{\kappa_{k-1}} = y_{\kappa_{k-1}}, \mathbf{X} = \mathbf{x}\right)$ as:

$$\prod_{r=1}^{J_k} \mathrm{P}\left(Y_{\kappa_k} = y_{\kappa_k} \mid \mathbf{X} \in R_{kr}^{\kappa}\right).$$

Here $\left\{R_{kr}^{\kappa}\right\}_{r=1}^{J_k}$ are mutually exclusive and exhaustive sub-regions of the input space at the k^{th}

level of the tree-structure, and J_k denotes the total number of nodes at the k^{th} level of the tree-structure.

LDsplit imposes that all the regions, $\left\{\left\{R_{kr}^{\kappa}\right\}_{r=1}^{J_k}\right\}_{k=1}^{K}$, be represented in the form of a tree-structure with $\left\{R_{kr}^{\kappa}\right\}_{r=1}^{J_k}$ contained in the k^{th} level of the tree-structure. The regions and posterior probabilities are approximated in a top-down greedy manner. Furthermore, the approximation of the posterior probabilities is affected by the permutation, κ, with each permutation in L^K being equally valid. The final step is to calculate the posterior probability estimates as an average over the ensemble of M tree-structures.

3.3 Label-Dependent splitting with predefined label orders

The LDsplit method is highly dependent on the order of the labels used to construct the tree-structures. Random LDsplit uses an ensemble of many different permutations of labels in an attempt to stabilise performance and produce a classification for all K labels in a straightforward way. However, the question arises whether certain label orders produce better performing tree-structures than others. If so, can a collection of these orders be identified before implementing LDsplit and will an ensemble of the corresponding tree-structures lead to poorer, similar, or better predictive performance than Random LDsplit? Furthermore, if predictive performance does not suffer severely, fitting a smaller number of tree-structures based on a smaller set of label orders would be computationally more efficient. Consequently, in this section an extension of LDsplit is introduced, referred to as Conditional LDsplit. Conditional LDsplit incorporates a strategy to determine informative label orders used for fitting tree-structures.

3.3.1 Label ordering problem

As mentioned in Section 2.7.2, although different to LDsplit, CC is a multi-label classification method for which label order is important. The label order of the CC model defines the order in which the input space is sequentially extended. This is fundamentally different to LDsplit since no input space augmentation takes place when fitting LDsplit. However, both CC and

LDsplit are highly dependent on the label order used for fitting. Read *et al*. (2011) acknowledge that the order of the classifier chain will normally affect its accuracy. However, determining an optimal order is a difficult task. Read *et al*. (2011) therefore suggest employing an ensemble of differently ordered classifier chains to overcome the issue, leading to ECC being introduced (as outlined in Section 2.7.5). This strategy is comparable to the multiple random label orders used when fitting Random LDsplit.

In recent years, this label ordering problem received more attention with research focusing on extensions and variations of CC (see for example Read *et al.* (2021)). The motivation for much of this research is the issue of error propagation (Senge *et al.*, 2014). The concern is that uncertainty at some part of the chain can lead to an error which is then propagated down the chain causing further errors. An ensemble of random label orders possibly alleviates the negative effect of poor label orders. However, proposals have also been made where authors aim to determine optimal label orders.

Jun *et al*. (2019) define two main types of methods to determine label order: single order and multiple orders. Single order methods use a single order of classifiers for the chain, whereas multiple order methods use multiple classifier chains with different orders to bypass the problem of determining one optimal label order.

Jun *et al*. (2019) propose a single order method which considers pairwise conditional entropies of labels. This method is outlined in Section 3.3.2 below. Other examples of single order methods include those that aim to find an optimal label order under a particular loss function and base classifier. Examples of these methods include the following:

- Dynamic Programming based Classifier Chain (CC-DP) as given in Liu and Tsang (2015).
- Genetic algorithms for optimising the label ordering as given in Gonçalves *et al.* (2013) and Gonçalves *et al.* (2015).

- Novel Monte Carlo schemes as presented in Read *et al.* (2013) and Read *et al.* (2014).

ECC, as discussed in Section 2.7.5, is an example of a multiple order method. Other examples of multiple order methods include the following:

- One-to-One Classifier Chains (OOCC) as presented by Da Silva *et al.* (2014). This method uses a distinct label order for each new observation to be classified. By using a nearest neighbour approach, training observations are identified which resemble the new observation. The label orders that perform well for these neighbour observations in training are considered as candidates for the new observation for testing.
- Bayesian Network (BN) methods form a sub-category of multiple order methods. Such methods build a BN (Koller and Friedman, 2009) or DAG to characterise the joint probability of all labels conditioned on the input space. A binary classifier is trained for each label by considering its parent labels in the DAG as additional input variables. By considering conditional independence conditions, these methods can form multiple label orders where each order is not necessarily a fully connected chain of all K labels. Examples of these methods include the following:
  - Bayesian Chain Classifiers (BCC) in Zaragoza *et al.* (2011). This method learns a BN that represents the label dependency structure of the data. Hereafter a CC model is fit where the order of the labels is consistent with the BN. The simplest version of BCC is implemented in Zaragoza *et al.* (2011), since each label has at most one parent label. Therefore, only one label is incorporated as an additional input variable for each classifier in the chain. Enrique Sucar *et al.* (2014) present a deeper analysis of BCC to get insights into its behaviour.
  - Learning by Exploiting lAbel Dependency (LEAD) in Zhang and Zhang (2010). The authors note that all labels inherently depend on the input space, so that

$\mathbf{X}$ is a common parent of all labels in the BN. Therefore, labels become dependent even if they are conditionally independent given $\mathbf{X}$. Zhang and Zhang (2010) attempt to remove this effect of $\mathbf{X}$ on all labels as follows. First a binary classifier is fit for each label separately and the errors of the classifiers are computed. If no error is made $e=0$. However, if the observation is in truth from class 1, but is classified to class 0, $e=1$. Similarly, if the observation is in truth from class 0, but is classified to class 1, $e=-1$. A BN of the computed errors is treated as an approximation of that of the labels with $\mathbf{X}$ as a common parent. This BN is used to determine the parents of each label which are used as additional input variables when fitting the classifier for each label.

Any sensible label ordering strategy can be used for Conditional LDsplit. In this work, the conditional entropy-based method of Jun *et al*. (2019) is used to define a set of orders for fitting tree-structures. This method seemed particularly attractive as it is straightforward and easily adapts to the LDsplit framework. The conditional entropy-based method is outlined in the next section.

3.3.2 Conditional entropy-based label ordering method

Consider two labels, Y_a and Y_b. Let $p(Y_a)$ be the probability (relative frequency) of label Y_a and $q(Y_a)=1-p(Y_a)$ be the probability of not having Y_a present. Furthermore, denote by $H(Y_a)$ the entropy of label Y_a estimated on the training data. $H(Y_a)$ is therefore calculated as:

$$H(Y_a)=-p(Y_a)\log p(Y_a) - q(Y_a)\log q(Y_a).$$

Jun *et al*. (2019) argue that if $H(Y_a)\le H(Y_b)$, then order $Y_a Y_b$ is better than $Y_b Y_a$, so that a label with higher entropy (higher uncertainty) is placed behind those with lower entropy (lower uncertainty). The argument is that a label with higher entropy, would probably lead to greater difficulty in classification. Therefore, labels with lower entropy should be considered first to reduce the risk of error propagation. However, instead of only evaluating label entropies separately, Jun *et al*. (2019) go further and include label relationships by extending their argument to conditional entropies.

Let $H(Y_a \mid Y_b)$ denote the conditional entropy of Y_a conditioned on Y_b, estimated on the training data. $H(Y_a \mid Y_b)$ is therefore calculated as:

$$H(Y_a \mid Y_b) = -\sum_{y_a, y_b} p(y_a, y_b) \log p(y_a \mid y_b),$$

where $p(y_a, y_b)$ is the value of the joint probability mass function of Y_a and Y_b, and $p(y_a \mid y_b)$ is the value of the conditional probability mass function of Y_a given Y_b.

By theoretically extending their arguments of entropy to conditional entropy, Jun *et al*. (2019) show that if $H(Y_a \mid Y_b) \leq H(Y_b \mid Y_a)$, order $Y_a\, Y_b$ is better than order $Y_b\, Y_a$. With some intuitive generalisations, the argument is extended to more than two labels.

Jun *et al*. (2019) construct a $K \times K$ matrix of conditional entropies where entry (a, b) gives $H(Y_b \mid Y_a)$, after which the sum of each row is calculated. If row a has the smallest row sum among the K rows of the matrix, Y_a should be placed at the end of the label ordering. The a^{th} row and a^{th} column of the conditional entropy matrix is now deleted. Again, the row sums of the now $(K-1)\times(K-1)$ matrix is calculated to find the minimum row sum. The corresponding label takes the $(K-1)^{th}$ position in the label ordering after which the matching row and column are deleted from the matrix. This is repeated until all the labels have been ordered.

Although it is possible to use this single ordering to fit one $K-$level LDsplit tree-structure, the method of Jun *et al*. (2019) is slightly adapted in this work so that multiple $m-$level tree-structures can be fit, where $m \leq K$.

3.3.3 Conditional LDsplit

Conditional LDsplit implements a strategy that determines a set of label orders to fit LDsplit tree-structures. Denote such a set of orders by L^{order}. Since $m-$level tree-structures are constructed for LDsplit, where $m \leq K$, each element of L^{order} is an ordering of m distinct labels. The size of L^{order} is dependent on the label ordering strategy that is used.

In general, to fit M $m-$level tree-structures, randomly sample M orderings without replacement from L^{order} and use these to fit the corresponding tree-structures as before. In other words, step 3 of Algorithm 3.1 is applied to the sampled label orders.

As mentioned, different strategies may be used to determine L^{order}. In this work the label ordering strategy of Jun *et al*. (2019) outlined in Section 3.3.2 above is applied to each of the $\binom{K}{m}$ possible combinations of selecting m labels from K, thus forming a collection of $\binom{K}{m}$ orders.

Note that in this work if the minimum row sum of the conditional entropy matrix corresponds to two or more rows, the label of the first of these rows is used for the next position in the label ordering and the corresponding row and column of this label are deleted from the conditional entropy matrix.

3.4 Considerations for datasets with large K

With the upsurge in online content, modern multi-label datasets often have many labels. This challenging aspect of multi-label data was also discussed in Section 1.4.3. For many multi-label learning methods, as the number of labels increases the methods become more computationally expensive. This is also true for LDsplit. It would therefore be valuable to outline how LDsplit can be scaled for such settings so that the computation time and computer memory needed to fit LDsplit for datasets with larger K, remains reasonable. Two scaling concerns of LDsplit are highlighted in this section. The first scaling concern is that if M is not set sufficiently large for datasets with large K, some labels may be excluded from the ensemble entirely, and no classification would be produced for these labels. Secondly, in this work, the sizes of L^m and L^{order} are dependent on K. In settings with large K, considerable computation time and computer memory is needed if all label orders are to be generated first, before randomly sampling M of these label orders for fitting tree-structures. These two concerns are discussed in the following two sections. For each concern, the scaling technique applied in this work is given.

3.4.1 Setting M sufficiently large to ensure all labels are represented

Since M label orders are randomly sampled from L^m or L^{order} to fit Random LDsplit or Conditional LDsplit, choosing M too small could cause some labels to be excluded from all the selected label orders in the ensemble. Consequently, no classification would be produced for such excluded labels. Generally, M should be so that at least $mM \geq K \Rightarrow M \geq \frac{K}{m}$. However, this will not be sufficient to ensure that all labels are represented at least once in the ensemble.

For any label, the probability to be included in a label order is given by $\frac{m}{K}$. Furthermore, since each of the M label orders are sampled independently, the probability distribution of the number of times any label is represented in the ensemble is given by $binomial\left(M, \frac{m}{K}\right)$. As a result, the probability of a label being excluded from the ensemble is given by $\left(1-\frac{m}{K}\right)^M$. Consequently, by increasing M the probability of excluding a label from the ensemble decreases.

As a simple example, consider a dataset with $K = 5000$ labels and suppose the number of tree-levels is set to $m = 3$. Figure 3.4 illustrates how the probability of excluding a label decreases as the ensemble grows. To ensure the probability of excluding a label is smaller than 0.001 for example, approximately 12000 tree-structures will have to be fit.

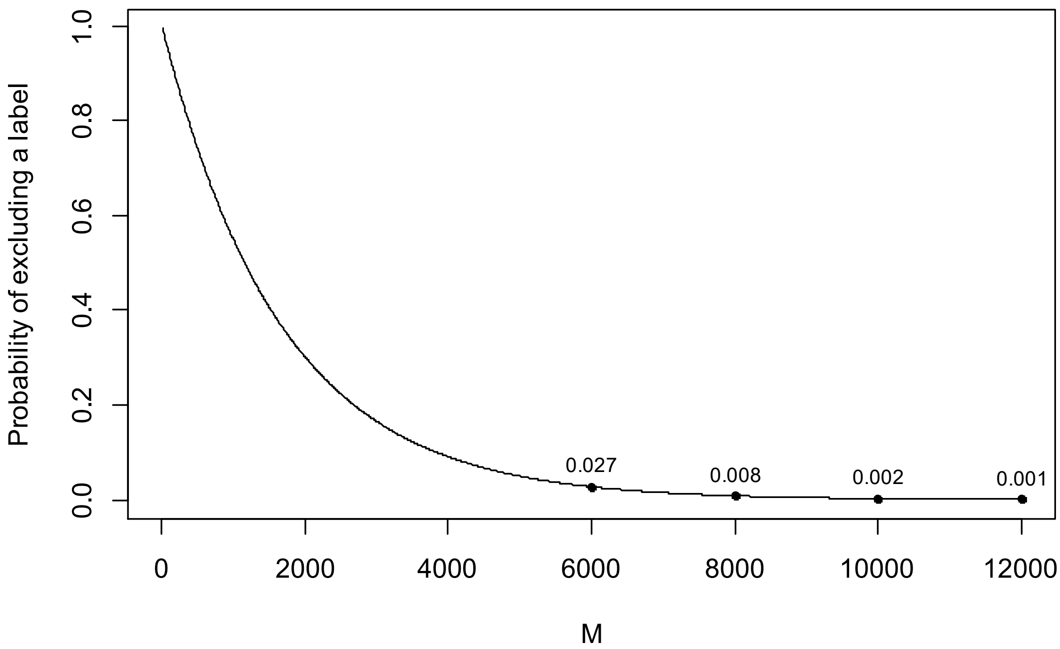


Figure 3.4 Probability of excluding a label when $K = 5000$ and $m = 3$

It does however remain difficult to know at what point M is set sufficiently large. Moreover, for datasets with large K a sufficiently large value of M will most likely require substantial computation time. Therefore, in this work the following strategy is used to ensure all K labels are represented at least once in the ensemble of tree-structures when K is large ($K > 100$).

- **Scaling technique**

Let $M_{\min}$ denote a user-specified starting value for the size of the LDsplit ensemble. The value of $M_{\min}$ will be influenced by the value of K and the computation time available. After $M_{\min}$ label orders have been established for either Random LDsplit or Conditional LDsplit, the label orders are reviewed to verify which (if any) of the K labels are excluded from all orders. The collection of excluded labels is denoted by $L_{excluded}$ and the size of $L_{excluded}$ is denoted by $|L_{excluded}|$.

For Random LDsplit, if $|L_{excluded}| \geq 1$, $L_{excluded}$ is randomly split up into $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil$ $m-$ permutations that are then used to fit additional tree-structures. If $|L_{excluded}|$ is not a multiple of $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil$, labels are randomly sampled from L to augment the $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil^{th}$ permutation.

For Conditional LDsplit, if $|L_{excluded}| \geq 1$, the collection of excluded labels is randomly split up into $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil$ disjoint subsets of size m. If $|L_{excluded}|$ is not a multiple of $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil$, labels are randomly sampled to augment the $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil^{th}$ set of labels. Each of the $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil$ sets of labels is thereafter ordered using the conditional entropy-based method of Jun *et al*. (2019). These orders are used to fit $\left\lceil \frac{|L_{excluded}|}{m} \right\rceil$ additional tree-structures.

3.4.2 Size of L^m and L^{order}

In this work, the sizes of L^m and L^{order} are $\frac{K!}{(K-m)!}$ and $\binom{K}{m}$ respectively. Consequently, the sizes of these collections increase as K increases. Generating all label orders in these collections before randomly sampling M of them without replacement, can take considerable computation time and computer memory for large K. Therefore, in this work, to reduce computation time and computer memory needed for datasets with large K ($K > 100$), the following strategy is used.

- **Scaling technique**

When fitting either Random LDsplit or Conditional LDsplit, $M_{\min}$ random samples of size m are initially taken from L without replacement.

For Random LDsplit, the order in which the labels appear for each of the $M_{\min}$ samples give the permutations P_j, $j = 1, 2, ..., M_{\min}$. Duplicates of permutations may occur, in which case the same permutation is used for more than one tree-structure.

For Conditional LDsplit, each of the $M_{\min}$ label samples is ordered using the conditional entropy-based method of Jun *et al*. (2019). Duplicates of orders may once again occur, causing the same label orders to be used for more than one tree-structure.

After this technique is applied, the scaling technique outlined in the previous section can be applied to ensure all K labels are represented in the ensemble.

Since the sizes of L^m and L^{order} are also influenced by the value of m, the scaling technique outlined here could also be applied when users want to consider large values of m.

3.5 Fitting LDsplit in R

As mentioned, in this dissertation the programming language R is preferred. Therefore, all R-functions to fit Random and Conditional LDsplit (as described in Section 3.2 and Section 3.3), as well as functions to classify observations using the fitted LDsplit ensembles, are provided in Appendix C. For each subsection of Appendix C, a diagram that contains short descriptions of the functions is provided to summarise the interaction of the respective functions.

To fit an LDsplit ensemble and obtain the appropriate output, the R-packages *rlist* (Ren, 2021) and *combinat* (Chasalow, 2012) are required. Furthermore, the written functions in Appendix C allow for two binary base classifiers within Random LDsplit and Conditional LDsplit. The first of these is an SVM, as implemented in the R-package *e1071* (Meyer *et al*., 2022). The default SVM-function arguments scale the data to zero mean and unit variance, use a soft margin and a radial basis kernel with $\gamma = \frac{1}{p}$ and $C = 1$. The second base classifier is a traditional binary classification decision tree, as implemented in the R-package *rpart* (Therneau *et al*. 2022). When a Conditional LDsplit model is fit, the conditional entropy-based ordering strategy of Jun *et al.* (2019) is implemented using the adaptation outlined in Section 3.3.3. The function *condentropy()* from the R-package *infotheo* (Meyer, 2022) is used to calculate conditional entropy.

Appendix C also contains functions to apply the two scaling techniques described in Section 3.4. To apply these techniques the R-package *FRACTION* (Ming, 2012) is required.

It is possible to implement parallel processing in R when fitting an LDsplit ensemble, since tree-structures are fit independently. Each tree-structure considers the same data but applies a different label order. Consequently, the tree-structures can be fit simultaneously on different cores so that the total running time is reduced. The adaptations required to fit an LDsplit ensemble using parallel processing is given in Appendix C. Note that the R-package *doParallel* (Daniel *et. al*., 2022) is required.

3.6 Distinctive and favourable properties of LDsplit compared to related methods

In this section distinctive properties of LDsplit, compared to related methods such as those given in Section 2.6 and Section 2.7, are discussed. While doing so the favourable properties of LDsplit are brought to the foreground. The section concludes with a summary of the major contributions of LDsplit as a multi-label tree-based ensemble classifier.

For LDsplit, one label is used per tree-level and simple binary classifiers are used to split the data. However, these resulting homogenous nodes are used as priors for estimation of the probability of another label. Therefore, observations that make up a node at a level of an LDsplit tree-structure do not only share information regarding the label used for that level, but also share information regarding the labels of parent levels. In this way, possible shared information and higher-order label correlations are included in an LDsplit tree-structure implicitly and in a simple manner. This is advantageous since it is widely acknowledged that effective exploitation of correlation among different labels is crucial for the success of a multi-label classifier (Zhang and Zhang, 2010). Simple problem transformation methods such as **BR** or **CLR** and the ensemble method **EBR**, for example, only consider first- and second-order label correlations.

The problem transformation method **CC** aims to improve upon BR by including higher-order label correlations through a chain of binary classifiers. Therefore, for both CC and LDsplit, label ordering helps to incorporate higher-order label correlation into the models. However, for

CC the label order determines the order in which the input space is sequentially extended, whereas for LDsplit the label order determines the labels used for the sequential levels of an LDsplit tree-structure. No input space augmentation takes place within LDsplit. Since LDsplit sequentially splits the data, this is also different to CC which uses all the data cases when fitting each binary classifier. For CC it is assumed that label correlations are shared by all the data cases at once, so that global label correlations are exploited. However, with LDsplit, it is assumed that label correlations are shared within subgroups of the data cases. This is a possible advantage of LDsplit over CC, since label correlations are not necessarily shared globally by all data cases.

As mentioned in Section 2.6, Huang *et al.* (2012) argue that it can be beneficial to exploit label correlations locally by dividing the data into subgroups. However, in their work this is done by performing clustering on the label space alone to develop an additional **LOC** input variable. On the other hand, for **CBMLC** the clustering step is based only on the input space. Different to the LOC and CBMLC frameworks, LDsplit considers the interactions between both input variables and labels when attempting to exploit local label correlations by dividing the data into subgroups. The intention is that observations that make up a terminal node share label information. Figure 3.5 below illustrates this by extending the simple image annotation example of Figure 1.4 in Chapter 1 to LDsplit. The LDsplit tree-structure of Figure 3.5 splits the data based on the label order [*Grass*, *Bird*, *Tree*]. Notably, images in the respective terminal nodes share label information. For example, Node 4 images contain only trees, whereas Node 6 images contain both grass and trees but no birds. Since an LDsplit tree-structure splits observations based on their posterior probabilities given by the fitted classifier, in a node some training observations may differ from the majority of observations with respect to the label of the level. This is illustrated by image $\mathbf{x}_6$ which moves to Node 7 not Node 6, even though no birds are in truth present in the image.

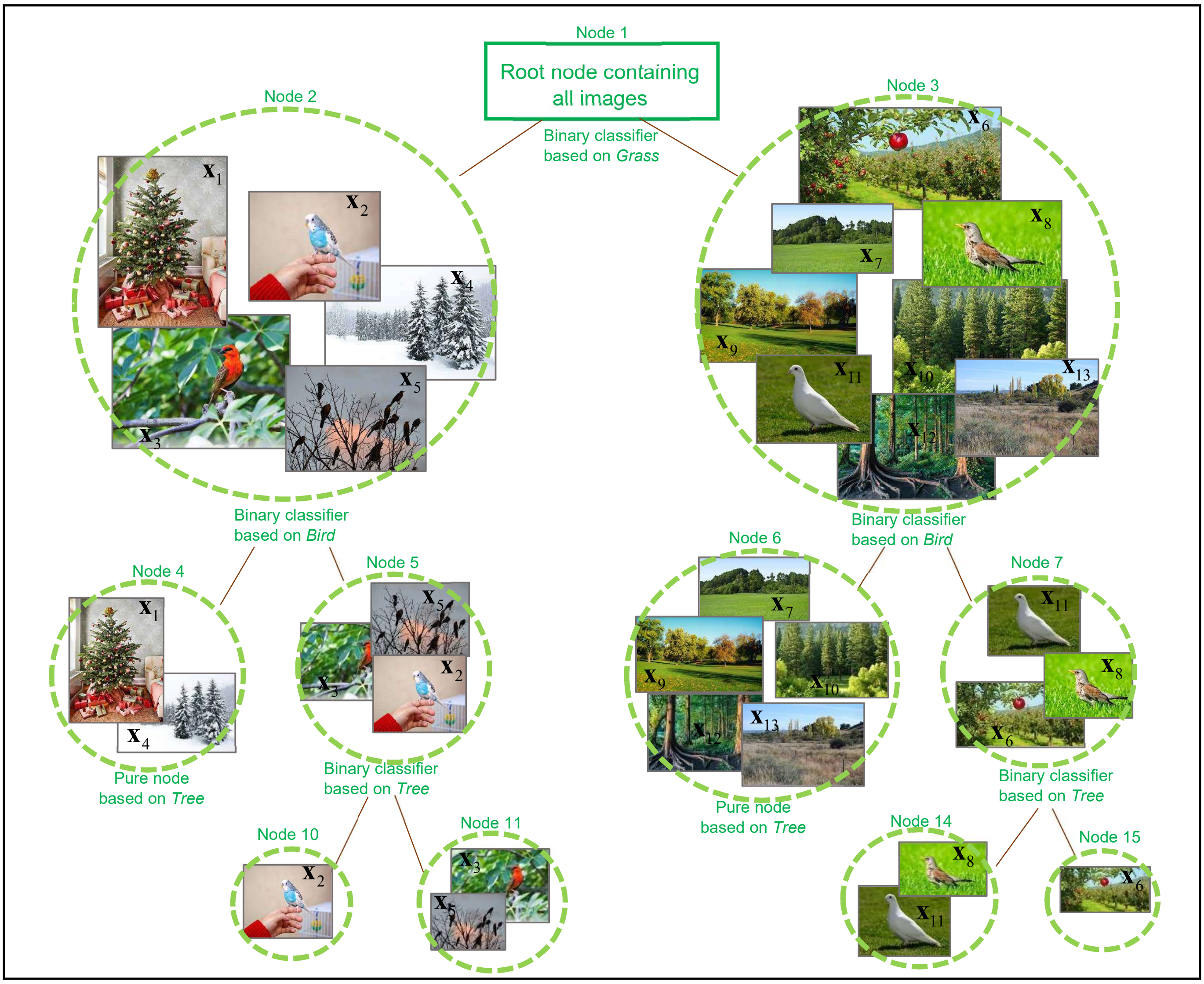


Figure 3.5 Images split into subgroups based on LDsplit tree-structure

Noticeably, LDsplit holds certain advantages over problem transformation methods. It is noteworthy that these advantages remain even in the specific case where decision trees are used as base classifiers for problem transformation methods. For example, when fitting BR with a random forest of decision trees as base classifier, no exploitation of label correlation takes place. For CC, by extending the input space of each decision tree classifier with the label relevancies of all previous classifiers in the chain, the label indicator variables become candidates for splitting the nodes. However, it could be that these label indicator variables are never chosen as splitting variables, so that the splitting rules are dominated by the original input variables. In this case the CC model resembles a BR model. In another example, **LP** suffers from the disadvantage that the transformation of the multi-label data to a multi-class structure may lead to a dataset with many classes and few observations per class. Therefore, when fitting LP with a decision tree base classifier, this property could lead to few observations in tree-nodes and cause the splitting rules of the tree to be unstable.

As mentioned in Section 2.7.5 two well-known ensemble methods aimed at improving the respective problem transformation methods CC and LP, are ECC and RAkEL.

ECC fits an ensemble of M CC models to reduce the overall negative impact of poorer label orders and to avoid the difficult task of determining a single optimal label order. This strategy is comparable to Random LDsplit, which randomly samples M label permutations to fit the ensemble of LDsplit tree-structures. However, the implementation of an ensemble also gives rise to additional flexibility for the LDsplit model. When an ensemble is fit, it is non-compulsory for each tree to give a classification for all K labels. Instead, each tree can be fit using a subset of $m \leq K$ labels, and the M classifications can be combined to obtain a final multi-label classification for all K labels (as outlined in Section 3.2.3). A large tree that uses all the labels may overfit the data. However, by setting $m < K$, an ensemble of $K-$level trees, some of which have terminal nodes on levels far preceding level K, is prevented. A similar strategy could be applied to ECC, so that each member of the ensemble is not necessarily a fully connected chain of the K labels. However, this is not the standard procedure of fitting ECC as given in Read *et al.* (2011). By only considering a subset of m labels per tree-structure and one label per tree-level for binary splitting, LDsplit attempts to learn slowly from the data while simultaneously implicitly exploiting local higher-order label correlations. This is a distinctive property of LDsplit compared to ECC, which attempts to capture label dependencies in a global manner using all the labels and data observations when fitting each ensemble member.

For **RAkEL**, the multi-label data are divided into smaller multi-label datasets using random subsets of the labels. In Tsoumakas *et al.* (2011), LP is fit to each sub-dataset. If LDsplit tree-structures, as defined in this work, are instead fit to each sub-dataset, RAkEL$_o$ would build an LDsplit model.

LDsplit also possesses unique properties when specifically compared to other multi-label tree-based methods. Firstly, a binary split is produced for each LDsplit node using a binary classifier defined by a single label per level. Any binary base classifier can be applied. Therefore, node-splitting is not necessarily based on a single input variable and splitting-value (which is the case for traditional trees), unless a tree-stump is used as base classifier. A further unique property of LDsplit is the use of random or predetermined hierarchical label structures that implicitly incorporates higher-order label correlations by considering a single label per tree-level. Popular multi-label trees such as **ML-C4.5** and **MODT**, as well as less well-known multi-label trees, **ML-SVMDT** and **LaCova**, consider all K labels when defining a splitting rule for a node and uses no hierarchical label structure.

HOMER is a multi-label tree-based method that exploits higher-order label correlations. Similar to LDsplit, HOMER transforms the initial learning task into several easier sub-tasks consisting of fewer training observations and labels by means of a tree-shaped hierarchy. However, different to LDsplit is the fact that HOMER performs balanced $k-$means clustering on the label space to form this label hierarchy. Moreover, this recursive label clustering procedure, which is based solely on the label space, determines how training observations are split into sub-nodes. The reason for this is that the training observations used per node are all the training observations having at least one of the respective labels of the meta-label present. This allows the respective nodes at a given level of a HOMER hierarchy to not necessarily be disjoint. Node-splitting of training data is therefore not determined by a fitted single-label classifier at the parent node which incorporates input variable interaction with the original labels to create disjoint sub-nodes, such as that of LDsplit. Only after the HOMER tree-shaped hierarchy has been formed, are local multi-label classifiers fit to the training data for classification of unseen observations. Furthermore, Papanikolaou *et al.* (2018) fit one HOMER tree-shaped hierarchy to the data which includes all K labels. As illustrated in Section 2.7.3, the design of HOMER gives large importance to the first clustering split of the labels. If the multi-label classifier assigns a 0 to one of the meta-labels, an immediate final classification of 0 is assigned to all the labels in this meta-label. This is different to the slow learning approach of LDsplit where an ensemble of trees is formed, with each tree considering $m \leq K$ labels and one label per tree-level for binary splitting.

The multi-label tree-based ensemble methods of Section 2.7.4 apply bagging or random forests to form an ensemble of multi-label trees. Although not done in this work, it is possible to fit a bagged or random forest ensemble of LDsplit trees. However, for LDsplit diversity of ensemble members is achieved by means of a different label order and label subset per ensemble member.

A further difference between LDsplit and the specific ensemble method, **SOSHF**, is that multiple structured labels are mapped to a single label per node when fitting SOSHF. LDsplit on the other hand considers simple label orderings of the original labels per tree. The two methods therefore make use of dissimilar single-label problems per node.

An **ML-Forest** tree recursively splits data cases into disjoint sub-nodes so that learning models at different levels of the tree work together to reveal multiple label concepts belonging to the data (Wu *et al.*, 2016). Even though an LDsplit tree is based on a similar objective, the splitting procedures of the two methods are markedly different. Firstly, an ML-Forest tree does not include a label ordering strategy that considers one label per tree-level; instead multiple labels are used to build a BR model per node. Furthermore, node-splitting of an ML-Forest tree is most often not binary. Figure 3.6 illustrates the fundamental differences between LDsplit and ML-Forest in broad terms.

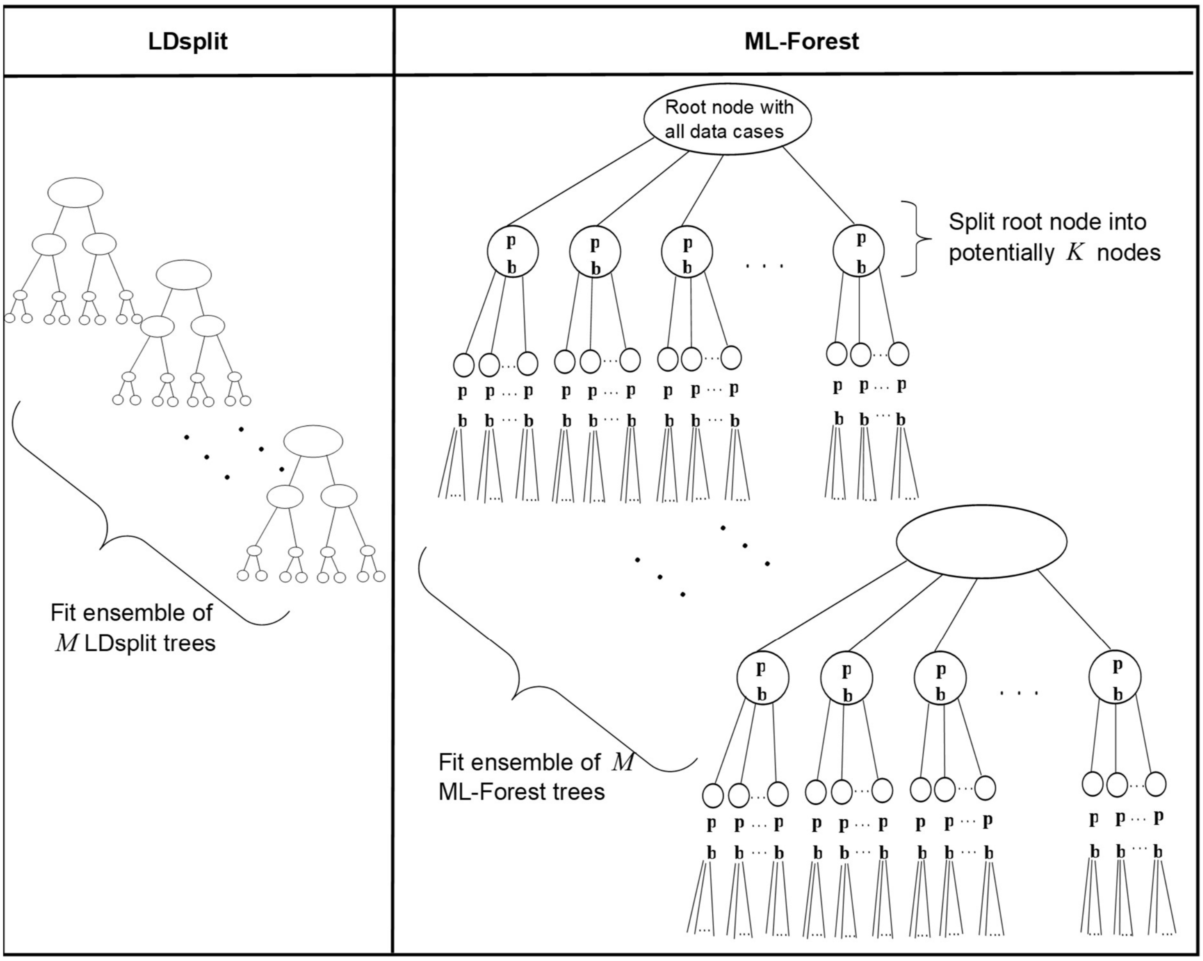


Figure 3.6 Differences between LDsplit and ML-Forest

As shown in Figure 3.6, a BR model is fit at the root node of each ML-Forest tree and each observation is classified to the class with maximum confidence score considering the K fitted classifiers. This splits the root node into potentially K disjoint nodes. Hereafter each node is once again potentially split into many child nodes. In contrast, LDsplit considers an ordered label subset per tree and uses binary splitting to create disjoint sub-nodes.

Since nodes of an ML-Forest tree are potentially split into many child nodes, a shallow tree with very few training data cases poses a risk for overfitting. Wu *et al.* (2016) attempt to combat this risk by fitting a bagged ensemble of ML-Forest trees. It may however be worthwhile to consider a similar strategy as that of LDsplit, so that each ML-Forest tree is fit considering only a subset of $m \leq K$ labels. In this way each node of an ML-Forest tree would be split into a maximum of m child nodes. Note that when applying this strategy, even in the simplest case with $m = 2$, LDsplit and ML-Forest will continue to fit different trees. This is illustrated in Figure 3.7 using a simple example.

Figure 3.7 gives an extract of a multi-label dataset consisting of $N = 12$ data cases and their corresponding label indicator variables for two labels, Y_A and Y_B. A BR model is fit to the data. This produces confidence scores for each of the data cases regarding each of the two labels. Suppose that the label ordering for the LDsplit tree is $\left[Y_A \; Y_B\right]$ and $t_{Y_A} = 0.5$. The root node of the LDsplit tree is therefore split considering only the confidence scores of Y_A. Data cases for which the confidence score of Y_A is less than 0.5 move down the left branch of the node and data cases with a confidence score exceeding 0.5 move down the right branch. If the same data are split using the ML-Forest method, a different partitioning is found. This is because an ML-Forest tree splits data cases based on their maximum confidence score. Since a different initial partitioning is found, the child nodes of the respective trees also differ.

			BR model confidence scores	
Data cases	Y_A	Y_B	Y_A	Y_B
$\mathbf{x}_1$	1	0	0.4	0.1
$\mathbf{x}_2$	1	0	0.8	0.3
$\mathbf{x}_3$	1	0	0.8	0.2
$\mathbf{x}_4$	1	0	0.8	0.1
$\mathbf{x}_5$	0	1	0.1	0.7
$\mathbf{x}_6$	0	1	0.2	0.5
$\mathbf{x}_7$	0	1	0.1	0.7
$\mathbf{x}_8$	0	1	0.2	0.8
$\mathbf{x}_9$	1	1	0.8	0.7
$\mathbf{x}_{10}$	1	1	0.9	0.6
$\mathbf{x}_{11}$	1	1	0.8	0.9
$\mathbf{x}_{12}$	1	1	0.7	0.9

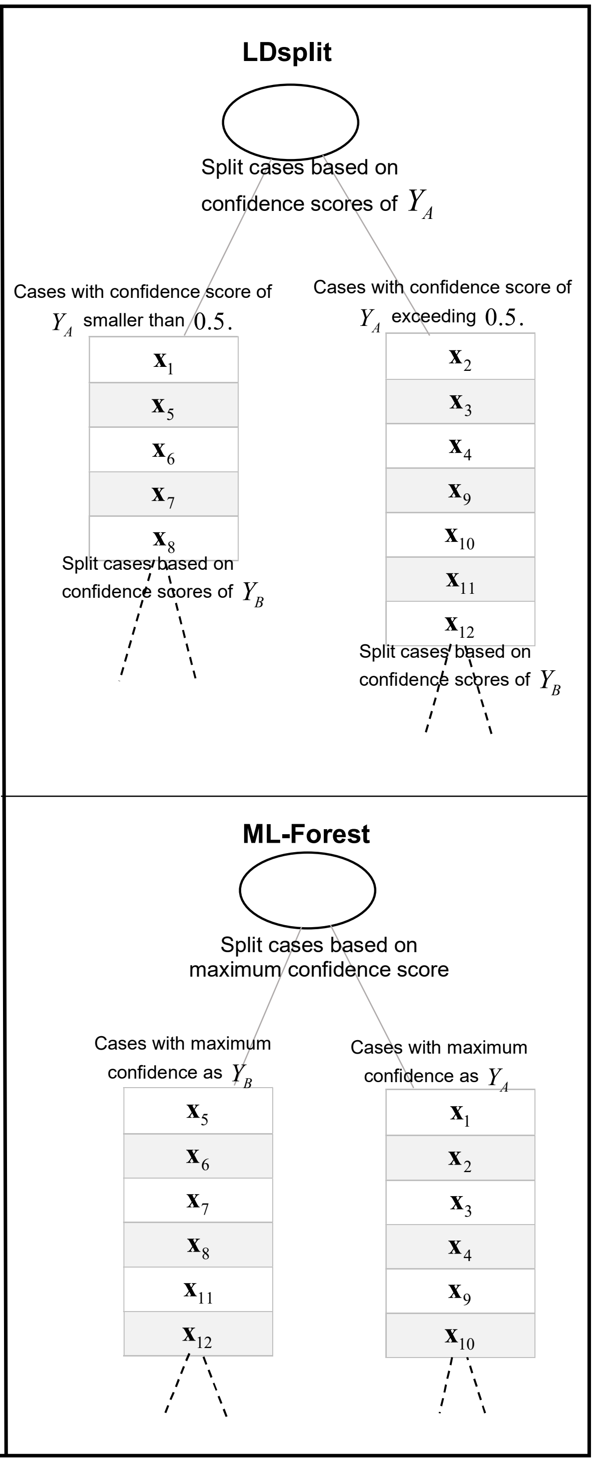


Figure 3.7 Initial split for LDsplit and ML-Forest tree

In conclusion the following summary emphasises the main contributions of LDsplit as a multi-label tree-based ensemble classifier:

- LDsplit implicitly includes possible shared information and higher-order label correlations in a simple manner. This is advantageous since it is generally agreed that exploitation of label correlation is a desirable property of a multi-label classification method.
- LDsplit exploits label correlations locally by splitting the data into subgroups. This is advantageous since label correlations are not necessarily shared globally by all observations.
- Different to traditional multi-label trees, LDsplit uses a random or predetermined hierarchical label order to implicitly exploit higher-order label correlations in a simple manner.
- By fitting an ensemble of trees, additional flexibility is achieved since each tree can be based on a subset of $m \leq K$ labels. Therefore, diversity of ensemble members is achieved through a different label order and label subset per ensemble member.
- An LDsplit ensemble learns slowly from the data by considering $m \leq K$ labels per tree and one label per tree-level to perform binary splitting of observations.
- This splitting process incorporates input variable interaction, so that the resulting hierarchy is not based on the label space alone.
- Any binary base classifier can be applied for node-splitting. This allows for further flexibility since splitting rules are not necessarily based on single input variables and splitting-values, such as for traditional trees. Binary classifiers may also be selected per label.
- Another important advantage of LDsplit is discussed in Chapter 5, namely that LDsplit can be useful for inference regarding global and local importance of variables.

3.7 Conclusion

In this chapter a new tree-based multi-label ensemble method, LDsplit, was proposed. The method fits an ensemble of trees with each tree based on an ordered label subset. By considering one label per tree-level, a simple binary classifier is used to split the data cases in the nodes. However, possible shared information and local higher-order label correlations are implicitly included in the model, since a tree-level depends on its parent levels that were formed using other labels.

Two different label ordering strategies were introduced, namely Random LDsplit and Conditional LDsplit. A detailed description of the fitting and classification procedures were given. Scaling techniques for large K were also given for both strategies to extend the contribution of this work to settings with larger K. The functions required to implement LDsplit are provided in Appendix C.

Furthermore, distinctive and favourable properties of LDsplit compared to other related multi-label learning methods were discussed in detail. LDsplit is a unique multi-label tree-based classifier since it applies an ensemble of random or predetermined hierarchical label structures and considers a single label per tree-level for binary splitting. One of the main advantages of LDsplit is that by using hierarchical label structures, higher-order label correlations are implicitly included in the model in a simple manner. Moreover, by splitting the data into subgroups using an ensemble of tree-structures, LDsplit models the dependence of labels in a local manner. Other multi-label classification models in the literature that attempt to capture label dependencies often do so in a global manner using all the labels and data observations at once. LDsplit attempts to learn slowly from the data by using only a subset of labels per tree and one label per tree-level for binary splitting. Furthermore, since input variable interaction is incorporated in the node-splitting process, the resulting hierarchy defined by each tree is not determined by the label space alone.

The next chapter empirically evaluates Random LDsplit and Conditional LDsplit using multi-label benchmark datasets.

Chapter 4: Empirical evaluation of LDsplit

4.1 Introduction

In this chapter an empirical study is performed on standard publicly available multi-label benchmark datasets. The study consists of three stages.

Firstly, the properties of the Random and Conditional LDsplit tuning parameters, m and M, are investigated in Section 4.2. This tuning parameter study is performed on the standard training parts of the benchmark datasets by means of cross-validation. The Random LDsplit and Conditional LDsplit results are discussed in Section 4.2.1 and Section 4.2.2 respectively. Based on the cross-validation results, conclusions are drawn regarding the interactions between the tuning parameters m and M, and general parameter recommendations are made. Furthermore, appropriate values for m and M are selected for each benchmark dataset based on the cross-validation results (see Table 4.1 and Table 4.2).

In the second stage, Random and Conditional LDsplit are compared in Section 4.3. This is done by directly comparing the cross-validation results achieved on the training data by the two models. This gives insight into the label ordering problem and how predictive performance can be influenced when smaller ensembles of predefined label orders are considered.

Finally, in the last stage, the predictive performances of Random LDsplit and Conditional LDsplit are compared to other well-known and related multi-label learning methods in Section 4.4. Up until this stage, only the standard training data parts of the benchmark datasets were considered by applying cross-validation. Therefore, in Section 4.4, the full set of training instances for each benchmark dataset is used to train the selected LDsplit models of Section 4.2. Hereafter each LDsplit model is used to classify the standard test datasets of each benchmark dataset. Performance is evaluated using multi-label evaluation measures as defined in Section 2.4. Since the models are trained and tested on the same standard train and test data splits, performance is directly comparable to that of the learning methods considered in Madjarov *et al.* (2012) and Wu *et al.* (2016). In addition to the learning methods of Madjarov *et al.* (2012) and Wu *et al.* (2016), four models (as recommended in the recent study by Bogatinovski *et al*. (2022)) are trained on the standard training datasets, and their classification performance evaluated using the standard test datasets. Ultimately, the classification performance of the LDsplit models is compared to that of 17 previously proposed multi-label learning models across six different benchmark datasets considering 12 evaluation measures. Furthermore, to assess the overall differences in performance of LDsplit compared to the other models, the corrected Friedman test (Iman and Davenport, 1980) and the post-hoc Nemenyi test (Nemenyi, 1963) are used. As will be seen in Section 4.4, both Random and Conditional LDsplit display very competitive results on the benchmark datasets. In some cases statistically significant improvements in predictive performance are observed (see Figure 4.3 and Figure 4.4).

The six multi-label benchmark datasets used throughout this chapter are: *Emotions, Scene, Yeast, Medical, Enron* and *Corel5k*. A summary of all the dataset characteristics is given in Table 2.2 with detailed descriptions of each dataset given in Section 2.3. These datasets were selected based on the following motivations. Firstly, these datasets are frequently used in multi-label empirical studies. Therefore, by maintaining the standard train and test data splits from the literature, the results of this chapter are reproducible and comparable to other research. This makes the empirical study of this chapter more accessible and useful to researchers. Secondly, the datasets are from different domains namely, one dataset from the music domain, one dataset from the biology domain and two datasets from both the image and text domains respectively. Lastly, the datasets differ in size and other descriptive properties as outlined in Section 2.2. It can be noted that the number of training instances, N, vary from 391 up to 4500, the number of input variables, p, vary from 72 up to 1449, the number of labels, K, vary from 6 to 374, and the average number of labels per sample vary from 1.074 up to 4.237.

In this chapter, all Random and Conditional LDsplit models are fit as described in Chapter 3. For simplicity, the minimum node-size of all LDsplit models fit in this chapter is fixed throughout at $n = 5$ and a threshold of 0.5 is used for all labels. For all Conditional LDsplit models, the conditional entropy-based ordering strategy of Jun *et al*. (2019) (Section 3.3.2) is implemented using the adaptation outlined in Section 3.3.3. Furthermore, since the *Corel5k* data has $K > 100$, the two scaling techniques described in Section 3.4 are applied when fitting all LDsplit models to this data.

Two different binary classifiers are considered to split the nodes of LDsplit tree-structures in this chapter. The first of these is an SVM as implemented in the R-package *e1071* (Meyer *et al*., 2022) and the second is a traditional binary classification decision tree as implemented in the R-package *rpart* (Therneau *et al*., 2022). The default SVM-function arguments scale the data to zero mean and unit variance, use a soft margin and a radial basis kernel with $\gamma = \frac{1}{p}$ and $C = 1$. Furthermore, the Gini index measures node impurity when a decision tree is used as base classifier for LDsplit.

Lastly, note that for the *Medical*, *Enron* and *Corel5k* data, some labels are absent for all training data cases. Since LDsplit nodes become terminal if the node is pure with respect to the current label, using such a purely absent label at any level of an LDsplit tree would immediately result in terminal nodes at this level. In this study, such labels are not allowed to be first in any label ordering. This prevents tree-structures having a root node as a terminal node. In particular, for Random LDsplit all permutations which have the first label entry equal to one of the purely absent labels, are removed. However, the same strategy cannot be implemented for Conditional LDsplit, since the conditional entropy-based label ordering method of Jun *et al*. (2019) often places such pure labels as the first entry in the label ordering. By removing all these label orders, some pure labels may be excluded from the ensemble entirely, and no classification for such labels can be produced. Instead, to overcome this, pure labels that make up the first entry of a label order are swapped with the subsequent label in the ordering for Conditional LDsplit. Those orders which also have a pure label as the second label in the ordering are removed. This gives a set of label orders with no purely absent label as the first entry in any ordering.

The following three sections outline and discuss the results of the three stages of the empirical study performed in this chapter.

4.2 Empirical evaluation of LDsplit tuning parameters

In this section the standard training datasets of the six benchmark datasets are used to examine the interaction between tuning parameters m and M of LDsplit. Based on the results, appropriate tuning parameter values are selected for each benchmark dataset for Random and Conditional LDsplit. These values are used in Section 4.4 to fit LDsplit models of which the predictive performance is compared to that of other multi-label learning methods.

For each training dataset five-fold cross-validation is applied by randomly splitting the observations of each training dataset into five approximately equal folds. With each fold in turn used as a validation set, various Random and Conditional LDsplit models are fit by varying the values of m and M. The fitted models are used to classify the validation set after which performance is evaluated using standard multi-label evaluation measures. The results of four example-based evaluation measures (Hamming loss, accuracy, F-score and subset accuracy), as well as two label-based evaluation measures (macro-F1 and micro-F1), are considered. Detailed definitions of the evaluation measures can be found in Section 2.4.

In this study, the grid of m values ranges from $m = 2$ to $m = 6$. For the *Emotions* and *Scene* data, this allows the LDsplit trees to reach their maximum number of levels, $m = K = 6$. For some of the models and datasets, it did not appear beneficial to increase m beyond 6, since the results showed signs of overfitting. For this reason, the maximum number of tree-levels considered throughout this study is $m = 6$. Furthermore, the ensemble size is limited to $M \leq 1000$ throughout. This is an appropriate upper limit since ensemble sizes are not increased beyond $M = 1000$ in Madjarov *et al.* (2012), Wu *et al.* (2016) or Bogatinovski *et al.*

(2022). Note that, unless otherwise stated, all M -values given for *Corel5k* results correspond to the user-specified $M_{\min}$ value.

The following two sections summarise and discuss the Random and Conditional LDsplit results respectively.

4.2.1 Evaluation of tuning parameters for Random LDsplit

The detailed Random LDsplit cross-validation results for each benchmark dataset can be found in Appendix A.1. For all the datasets the average performance of each m and M combination over the five cross-validation folds are provided considering each of the six evaluation measures. The standard deviation of performance over the five folds are given in brackets and the model that produces the best performance for each evaluation measure is given in bold for each dataset. The results are discussed in detail below. However, for a quick overall visual interpretation of the interaction between tuning parameters m and M for Random LDsplit, Figure 4.1 provides a summary for the *Emotions*, *Scene* and *Yeast* data when an SVM is used as base classifier. Since M label permutations are randomly sampled from L^m, a stabler representation of the Random LDsplit results over the five folds is obtained by refitting each model five times per fold. Therefore, the results in Figure 4.1 show the average performance of each m and M combination based on the five repetitions as well as the five-fold cross-validation.

The variation in performance due to the random selection of M label permutations can also be investigated. This is done as follows. A single validation fold of the *Emotions* data is considered, and the variation in performance is calculated after refitting each Random LDsplit model five times. The results for each m and M combination when an SVM is used as base classifier are given in Appendix A.2. The results for Hamming loss and F-score when $m = 5$ are given in Figure 4.2 to aid in the discussion of the general behaviour of these results.

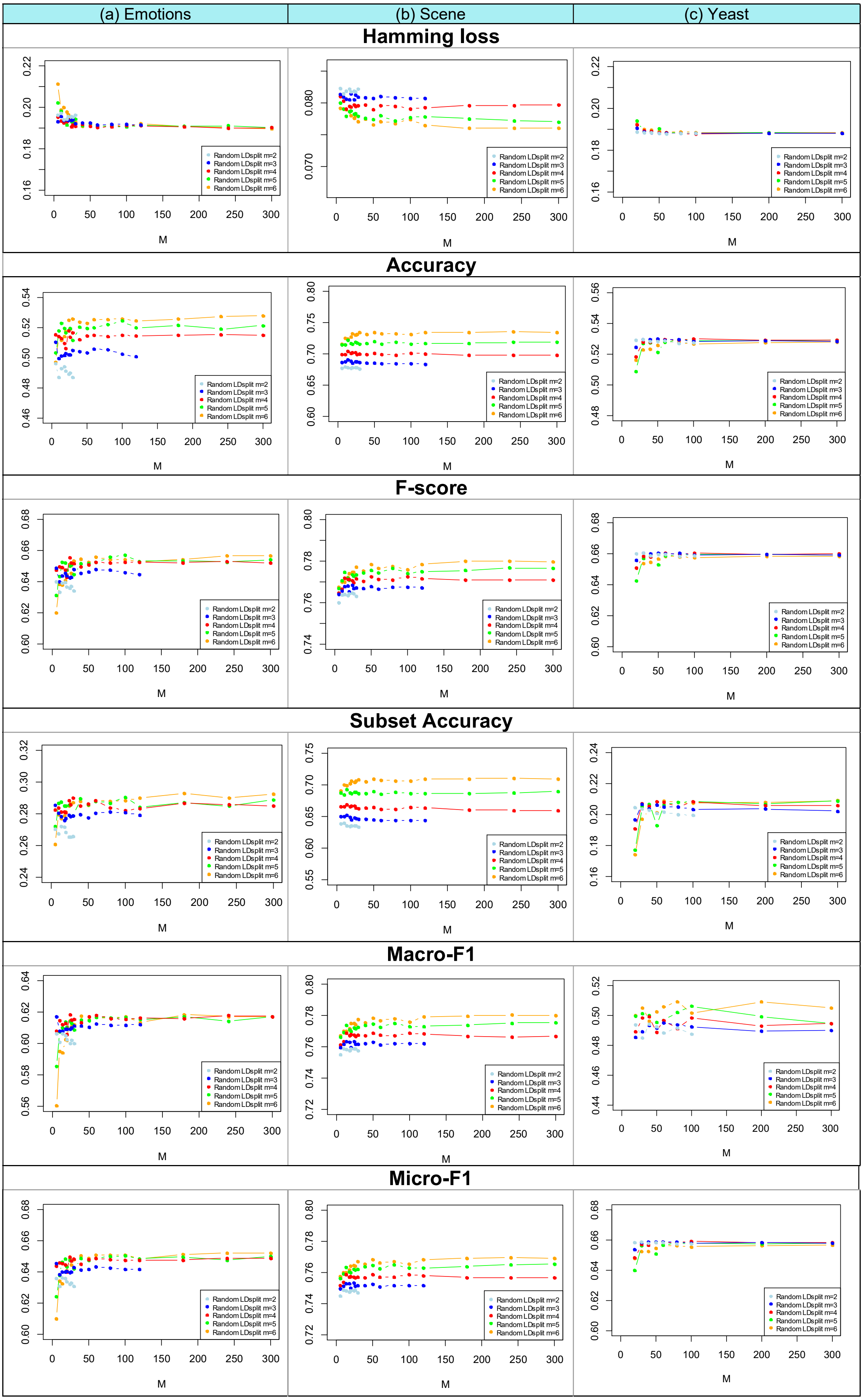


Figure 4.1 Results for different choices of m and M for Random LDsplit

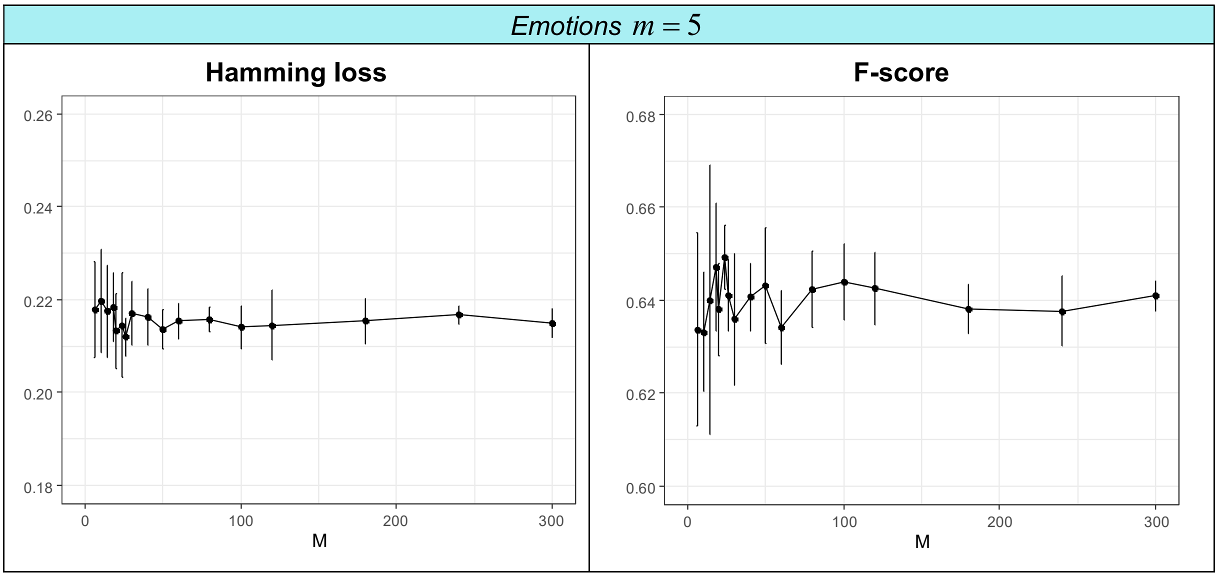


Figure 4.2 Average model performance and standard deviations based on five repeats

- **Overview of interaction between tuning parameters**

Figure 4.1 shows that for different values of m, performance stabilises after M is increased beyond approximately $M = 100$ for *Emotions*, *Scene* and *Yeast*. Increasing the value of M beyond 100 does not lead to a large increase or decrease in performance for these datasets. However, for the *Emotions* data, models with $m \le 3$ show a slight decrease in performance for accuracy and the F-score when M is large. Nonetheless, in general overfitting does not seem to be a concern for large values of M in Figure 4.1. Therefore, according to Figure 4.1, to ensure the optimal performance is achieved by Random LDsplit, M should be set large with respect to K. It could be that since labels are randomly permuted when fitting Random LDsplit, independent labels are sometimes grouped together to fit an LDsplit tree which models non-existing dependencies. However, when a large ensemble is considered, these non-existing dependencies should average out.

Figure 4.2 illustrates that the variation in model performance due to the random selection of M label orders is small and becomes less as the ensemble size increases. This result gives more incentive to set M large in general.

Regarding the choice of m, slight differences for each performance measure are observed when M is large in Figure 4.1. Subset accuracy for the *Scene* data appears to be the most sensitive to the choice of m. However, when M is not sufficiently large, the results for larger values of m are more unstable than those for smaller values of m. Consider the Macro-F1 results of the *Emotions* data in Figure 4.1 as an example. When $M = 20$, the performances of models with $m = 5$ and $m = 6$ are much worse than those with $m < 5$. This implies that for large values of m, if M is not set sufficiently large, the negative consequences are more severe compared to the case where m is small and M is not set sufficiently large.

Since computation cost increases with the value of m, smaller values of m are preferred in general, unless larger m values lead to substantial performance improvements. In this case however, the results of Figure 4.1 indicate that large m do not only lead to additional computation cost, they may also cause unstable model behaviour if M is not set sufficiently large. Furthermore, if M is in fact set sufficiently large for a large value of m, Figure 4.1 shows that the performance of the model does not improve substantially from that of a model with a smaller value of m. These findings also remain generally true when considering all the results of Appendix A.1.

Additionally, results of Appendix A.1 indicate that m can be set too large so that overfitting occurs in some cases. For example, accuracy, F-score, subset accuracy, macro-F1 and micro-F1 or the *Emotions* data become slightly unstable for Random LDsplit models with a decision tree base classifier and $m = 6$. Fitting an LDsplit model with a large value of m can result in an ensemble of trees where each tree has many terminal nodes occurring on levels preceding Level m. This could make the model more susceptible to error propagation since the low-level terminal nodes are used to determine classifications for a larger number of labels. Consequently, an error can be propagated to a greater number of labels.

In general, it appears that the choice of m has a greater effect on classification performance than the choice of M. For example, for small m, performance may improve by increasing m by 1, while increasing M by say 50 may have less influence on classification performance in comparison.

By considering the above observations for m and M, as a rule of thumb it appears best to set m small ($3 \leq m \leq 4$) while M is large with respect to K ($5K < M < 10K$) for Random LDsplit. This agrees with the hypothesis of Section 3.2.4.

This rule of thumb provides a convenient guideline for setting the values of m and M for Random LDsplit; however, the predictive performance remains dependent on the multi-label dataset at hand. Therefore, if computation time allows for it, an adequate way to determine the values of m and M is by means of cross-validation. This is the approach followed in this study. Table 4.1 gives the selected m and M values for each Random LDsplit model across the six benchmark datasets. Unless otherwise stated, models with the lowest average Hamming loss across the five cross-validation folds are selected per dataset. If the minimum Hamming loss is given by more than one model, the model with the smallest values for m and M among them is selected.

When determining an appropriate binary base classifier for LDsplit, the performance of candidate classifiers over validation folds of the training data can be considered. In this study, results from Appendix A.1 show that the Random LDsplit models with an SVM base classifier (R-LDsplit_SVM) achieve better performance than those with a decision tree base classifier (R-LDsplit_Tree), for all datasets except the *Medical* data. The less complex decision tree base classifier seems to underfit the *Emotions*, *Scene*, *Yeast*, *Enron* and *Corel5k* data. However, for the *Medical* data, R-LDsplit_Tree performs better overall. This indicates that the simpler decision tree classifier is a better choice for the bag-of-words data. Even though the *Enron* data is also summarised using a bag-of-words framework, R-LDsplit_SVM shows better predictive performance for the data. Further differences between the performance of R-LDsplit_SVM and R-LDsplit_Tree are highlighted in Section 4.4 when their predictive performance on test data is compared to that of other previously proposed methods.

A detailed discussion of the Random LDsplit cross-validation results of Appendix A.1 follows next.

- **Discussion of cross-validation results and summary of selected models**

For different values of m, Hamming loss of R-LDsplit_SVM does not vary substantially for the ***Emotions*** data. However, for models with $m \geq 4$, stable favourable performance is only achieved once $M \geq 30$. The choice of m has a larger effect on the Hamming loss of R-LDsplit_Tree , since models with $m \geq 4$ and large M markedly improve compared to those models with $m < 4$. In this case, performance of R-LDsplit_Tree, which uses a less complex base classifier than R-LDsplit_SVM, can be improved by increasing the value of m so that a more complex R-LDsplit_Tree model is fit.

For each of F-score, subset accuracy, macro-F1 and micro-F1, R-LDsplit_SVM models with $m > 2$ show only slight differences for the *Emotions* data. The differences in accuracy due to the choice of m for R-LDsplit_SVM models are more noticeable. However, models with $m \geq 5$ often achieve the best results for these measures.

As indicated in Table 4.1, the chosen R-LDsplit_SVM model for the *Emotions* data has $m = 5$ and $M = 100$. Not only does this model achieve the lowest Hamming loss, it also achieves the best results for accuracy, F-score, macro-F1 and micro-F1.

Accuracy, F-score, subset accuracy, macro-F1 and micro-F1 for the *Emotions* data become slightly unstable for R-LDsplit_Tree models with $m = 6$. This suggest that by increasing m, a more complex R-LDsplit_Tree model can improve performance, however overfitting can occur if m is chosen too large. As indicated in Table 4.1, the chosen R-LDsplit_Tree model for the

Emotions data has $m = 4$ and $M = 240$, as it is the smallest model with the lowest value for Hamming loss.

For the ***Scene*** data, by increasing the value of m, R-LDsplit_SVM models show marked improvements in performance across all evaluation measures. However, performance is fairly stable for different values of M. Similar behaviour is observed for R-LDsplit_Tree models. Therefore, the chosen R-LDsplit_SVM model for the *Scene* data has $m = 6$ and $M = 100$, whereas the chosen R-LDsplit_Tree model has $m = 6$ and $M = 300$.

Similar to the *Emotions* data, for different values of m, Hamming loss of R-LDsplit_SVM does not vary substantially for the ***Yeast*** data. However, for R-LDsplit_Tree, by increasing m Hamming loss values continue to improve. Furthermore, for R-LDsplit_SVM no marked improvements in accuracy, F-score, subset accuracy, macro-F1 or micro-F1 are observed by increasing m beyond $m = 3$. It does however appear best to set $M < 500$, as performance slightly decreases for R-LDsplit_SVM models with $M \geq 500$. Consequently, the selected R-LDsplit_SVM model for the *Yeast* data has $m = 3$ and $M = 200$. For R-LDsplit_Tree the best results for accuracy, F-score, subset accuracy, macro-F1 and micro-F1 are achieved when $4 \leq m \leq 5$. Even though this is the case, the selected R-LDsplit_Tree model for the *Yeast* data has $m = 6$ and $M = 300$, since this model achieves the minimum Hamming loss.

For the ***Medical*** data, Hamming loss of R-LDsplit_SVM is especially stable. For no obvious reason, all models, except the model with $m = 6$ and $M = 500$, achieve a Hamming loss of 0.012. Since R-LDsplit_SVM with $m = 6$ and $M = 500$ achieves a Hamming loss of 0.011, it is the selected R-LDsplit_SVM model for the *Medical* data. For all other evaluation measures, performance across different values of m are very much the same once $M \geq 100$. Therefore, R-LDsplit_SVM with $m = 6$ and $M = 500$ also remains an appropriate choice when considering these evaluation measures.

Considering the performance of R-LDsplit_SVM on the *Medical* data, marked improvements in accuracy, F-score, subset accuracy, macro-F1 and micro-F1 are observed when R-LDsplit_Tree is instead fit. Hamming loss for R-LDsplit_Tree is fairly stable, however more variation in performance is observed compared to that of R-LDsplit_SVM. Apart from accuracy and F-score, performance across different values of m for R-LDsplit_Tree is stable once $M \geq 100$. The R-LDsplit_Tree model with $m = 4$ and $M = 500$ is selected for the *Medical* data. This model is the second smallest model which achieves the minimum value of Hamming loss. It is preferred above the smallest model, as it achieves high performance over all evaluation measures and only requires a small increase in computation cost.

Hamming loss cannot be used to select the R-LDsplit_SVM model for the ***Enron*** data, since, for no obvious reason, the same value of 0.047 is found for all the fitted models. The standard error over the five cross-validation folds is also constant for all models at a low value of 0.001. Even though the other evaluation measures show a similar pattern of stable behaviour, these measures are used to select an appropriate R-LDsplit_SVM model for the *Enron* data. The R-LDsplit_SVM model with $m = 3$ and $M = 100$ is selected since it is a small model that gives good performance across all evaluation measures, and additionally achieves the best performance for subset accuracy and micro-F1.

When R-LDsplit_Tree models are fit to the *Enron* data, performance generally weakens compared to R-LDsplit_SVM. However, apart from this difference, the observations regarding the values of m and M are similar to that of R-LDsplit_SVM. Performance across the different evaluation measures are very stable regardless of the values of m and M. Since Hamming loss shows some variation over m, the smallest model which produces the lowest value for Hamming loss is selected. This model has $m = 6$ and $M = 80$.

Since ***Corel5k*** has $K = 374$, only $M_{\min} \geq 500$ were considered. However, the results appear very stable with regards to M and m. Therefore, for R-LDsplit_SVM and R-LDsplit_Tree, the number of levels per tree-structure is selected as $m = 2$ and the ensemble size is set to $M_{\min} = 500$. This implies that only pairwise label correlations are considered for the *Corel5k* data in Section 4.4.

Table 4.1
Random LDsplit models selected for each dataset based on five-fold cross-validation

Dataset	Model	m	M
Emotions	Random LDsplit (SVM)	5	100
	Random LDsplit (Decision Tree)	4	240
Scene	Random LDsplit (SVM)	6	100
	Random LDsplit (Decision Tree)	6	300
Yeast	Random LDsplit (SVM)	3	200
	Random LDsplit (Decision Tree)	6	300
Medical	Random LDsplit (SVM)	6	500
	Random LDsplit (Decision Tree)	4	500
Enron	Random LDsplit (SVM)	3	100
	Random LDsplit (Decision Tree)	6	80
Corel5k	Random LDsplit (SVM)	2	500
	Random LDsplit (Decision Tree)	2	500

4.2.2 Evaluation of tuning parameters for Conditional LDsplit

The Conditional LDsplit results for the respective benchmark datasets are summarised in Appendix A.3. The results in Appendix A.3 are the average performance over the five cross-validation folds, with the standard deviation of performance over the five folds given in brackets. Once again, the model that produces the best performance for each evaluation measure is given in bold.

For the *Emotions* and *Scene* data, where $K = 6$, L^{order} contains a small number of elements. Consequently, for each value of m, only the results of one Conditional LDsplit model are reported by setting $M = \binom{6}{m}$ for $m = 2,...,6$. For the *Yeast*, *Medical*, *Enron* and *Corel5k* data, L^{order} contains a larger number of elements. Therefore, multiple Conditional LDsplit models are reported for each value of m by increasing the value of M. The cross-validation results of Appendix A.3 are used to select an appropriate Conditional LDsplit model for each dataset. The selected models are summarised in Table 4.2. Once again, unless otherwise stated, models with the lowest average Hamming loss across the five cross-validation folds are selected per dataset. If the minimum Hamming loss is given by more than one model, the model with the smallest values for m and M among them is selected.

Note that the remarks of this study could be dependent on the label ordering strategy used for Conditional LDsplit. In this case, these remarks pertain to the conditional entropy-based ordering strategy of Jun *et al*. (2019).

- **Overview of interaction between tuning parameters**

Similar to Random LDsplit, results from Appendix A.3 show that the Conditional LDsplit models with an SVM base classifier (C-LDsplit_SVM) achieve better performance compared to those with a decision tree base classifier (C-LDsplit_Tree), for all datasets except the *Medical* data. In Section 4.4, further differences between the performance of C-LDsplit_SVM and C-LDsplit_Tree are highlighted.

Results from Appendix A.3 show that even though the *Emotions* and *Scene* data are similar in label space size (both having $K = 6$ labels), model behaviour is different depending on the value of m. For the *Emotions* data, models with $m \leq 4$ perform similarly. However, when $m \geq 5$ performance starts to deteriorate compared to the best performing models over all the evaluation measures. Slightly different behaviour is observed for the *Scene* data as models with $m = 2$ and $m = 3$ are not the top performers, especially in terms of accuracy and subset accuracy. This motivates that cross-validation may be the best method for determining m in practice for Conditional LDsplit.

In most cases; however, if a dataset displays an improvement in performance due to a larger value of m, this improvement is small in comparison to the large deterioration observed for some datasets when m is chosen too large. For example, performance in terms of all evaluation measures greatly deteriorate when $m = 6$ for the *Yeast* data. For the *Medical* data, once $m \geq 4$, a deterioration in performance is observed for all evaluation measures except macro-F1.

Furthermore, as is the case for Random LDsplit, overfitting does not seem to be a concern for large M for Conditional LDsplit. In Appendix A.3, the *Yeast* data models with $M \geq 100$ generally show better performance than models having $M = 40$. For the *Medical* data, models that have $m \leq 5$ and $M \geq 300$ generally show better performance than models having $m \leq 5$ and $M = 100$. For *Enron* and *Corel5k*, performance appears generally stable regardless of the value of M.

Considering the above remarks on m and M for Conditional LDsplit, the same general rule of thumb may be followed as that for Random LDsplit, namely that m should be set small ($3 \leq m \leq 4$) while M is large compared to K. For smaller datasets, it may be possible to set M to its maximum value of $\binom{K}{m}$. It is however noted in Section 4.3 that a similar level of performance than Random LDsplit can be achieved by a Conditional LDsplit model with a smaller value of M.

The rule of thumb is a convenient guideline for setting the values of m and M in practice. However, as computation time allows for it in this study, cross-validation is used to determine the appropriate model per dataset. Some remarks considering the chosen Conditional LDsplit models are given below, after which a summary of the selected models per dataset is given in Table 4.2.

- **Discussion of cross-validation results and summary of selected models**

As mentioned, for both C-LDsplit_SVM and C-LDsplit_Tree, models with $m \leq 4$ perform similarly on the ***Emotions*** data. However, when $m \geq 5$ performance starts to deteriorate compared to the best performing models over all the evaluation measures. Note that if $m = 6$, only a single tree-structure is fit using one ordering of all the labels. This model does not produce good results for the *Emotions* data, especially when a decision tree is used as base classifier. Consequently, as given in Table 4.2, the selected C-LDsplit_SVM model and C-LDsplit_Tree model for the *Emotions* data both have $m = 3$ and $M = 20$.

Conversely, for the ***Scene*** data models with $m = 2$ and $m = 3$ are not the top performers, especially in terms of accuracy and subset accuracy. In this case, for both C-LDsplit_SVM and C-LDsplit_Tree, the model that fits a single tree-structure of one ordering of all the labels achieves the best performance for accuracy and subset accuracy. Overall it appears best to set $m \geq 4$ when a Conditional LDsplit model is fit to the *Scene* data. Therefore, the selected C-LDsplit_SVM model and C-LDsplit_Tree model for the *Scene* data both have $m = 4$ and $M = 15$.

Note that in Table 4.2, the C-LDsplit_SVM model selected for the ***Yeast*** data is the model with $m = 4$ and $M = 100$. Although this model is not the smallest of those that produce the minimum Hamming loss for the *Yeast* data, it achieves high performance for all evaluation measures with only a small increase in computation cost. A similar argument holds for the C-LDsplit_SVM model selected for the ***Medical*** data and ***Enron*** data. For all other models, the smallest models are selected that give the lowest average Hamming loss across the five cross-validation folds.

Table 4.2
Conditional LDsplit models selected for each dataset based on five-fold cross-validation

Dataset	Model	m	M
Emotions	Conditional LDsplit (SVM)	3	20
	Conditional LDsplit (Decision Tree)	3	20
Scene	Conditional LDsplit (SVM)	4	15
	Conditional LDsplit (Decision Tree)	4	15
Yeast	Conditional LDsplit (SVM)	4	100
	Conditional LDsplit (Decision Tree)	4	1000
Medical	Conditional LDsplit (SVM)	2	500
	Conditional LDsplit (Decision Tree)	2	100
Enron	Conditional LDsplit (SVM)	3	100
	Conditional LDsplit (Decision Tree)	6	100
Corel5k	Conditional LDsplit (SVM)	2	500
	Conditional LDsplit (Decision Tree)	2	500

4.3 Comparison of Random LDsplit and Conditional LDsplit

In this section Random LDsplit and Conditional LDsplit are compared using the five-fold cross-validation results obtained for the training datasets of *Emotions*, *Scene*, *Yeast*, *Medical* and *Enron*. Since the *Corel5k* results appear stable regardless of fitting Random or Conditional LDsplit, the dataset is not considered in this section. Note that a comparison in performance between Random and Conditional LDsplit as based on test data is drawn in Section 4.4.

In this section, for all datasets except the *Medical* data, comparison is drawn between models that use an SVM base classifier. This is because an SVM is found to be the appropriate base classifier for these datasets in Section 4.2. For the *Medical* data comparison is drawn between models that use a decision tree base classifier.

For each value of m $(2 \leq m \leq 6)$, the results of one Conditional LDsplit model is compared to that of two Random LDsplit models. The first of these Random LDsplit models has a similar ensemble size as the Conditional LDsplit model, whereas the second Random LDsplit model fits a larger ensemble. This is done for two reasons. Firstly, to evaluate the difference in performance of Random LDsplit and Conditional LDsplit when ensembles of similar sizes are fit, and secondly, to evaluate if a Conditional LDsplit model with a smaller ensemble can produce similar or better predictive performance than a large Random LDsplit model that uses the same value for m. Note that since the size of a Random LDsplit model is upper bounded by $\frac{K!}{(K-m)!}$, for the *Emotions* and *Scene* data, the ensemble size of the second Random LDsplit model remains relatively small.

The results for all five datasets are summarised in Appendix A.4. To ease the interpretation of these results, for each value of m, the model that produces the best performance for each evaluation measure is given in bold and the percentage decrease in performance is reported in brackets for the other two models. Furthermore, for each evaluation measure, the model that gives the best performance across all values of m is highlighted.

For the ***Emotions*** data, the three models compared for $m = 2$ achieve very similar performance for all the evaluation measures; however, Conditional LDsplit does achieve the best performance for four of the six evaluation measures. When $m = 3$, the Conditional LDsplit model outperforms the Random LDsplit model of similar ensemble size and achieves the same or better performance than the large Random LDsplit model. Performance of Conditional LDsplit deteriorates once $m \geq 4$, so that the Conditional LDsplit models are generally outperformed by the Random LDsplit models for $m \geq 4$.

Similar to the *Emotions* data, for the ***Scene*** data the models compared at $m = 2$ and $m = 3$ achieve very similar performance for all the evaluation measures; however, Conditional LDsplit achieves slightly better performance for all six evaluation measures. For $m = 4$, the Conditional LDsplit model continues to outperform the Random LDsplit model of similar ensemble size and achieves the same or better performance than the large Random LDsplit model. It appears that accuracy and subset accuracy benefit the most from the implementation of a label ordering strategy on the *Scene* data. For $m = 4$ a 2.4% decrease in accuracy and 4.1% decrease in subset accuracy are observed when a Random LDsplit model is fit instead of a Conditional LDsplit model of the same size. Furthermore, by increasing the ensemble size of the Random LDsplit model, this decrease in accuracy and subset accuracy only amount to 2.1% and 3.8% respectively. Therefore, despite the larger ensemble for Random LDsplit, the Conditional LDsplit model achieves better performance for accuracy and subset accuracy. A similar level of accuracy and subset accuracy as the Conditional LDsplit model with $m=4$ is only achieved by a Random LDsplit model with $m \geq 5$. Nonetheless, accuracy and subset accuracy of Conditional LDsplit models fit to the *Scene* data continue to increase as m increases. A deterioration in other evaluation measures does however occur once $m \geq 5$ for Conditional LDsplit.

For the ***Yeast*** data, even though the models compared at $m=2$ achieve similar performance, Conditional LDsplit achieves the best performance for all six evaluation measures. For $3 \leq m \leq 5$ the Conditional LDsplit models are competitive with the Random LDsplit models; however, the large Random LDsplit models achieve the best results across most of the evaluation measures. Once $m=6$, performance of Conditional LDsplit dramatically deteriorates compared to that of the Random LDsplit models with $m=6$. It is noteworthy that for $m=4$, a 6.3% decrease in subset accuracy is observed when a Random LDsplit model is fit instead of a Conditional LDsplit model of the same size. By increasing the ensemble size of the Random LDsplit model, this decrease in subset accuracy is reduced to 1.8%. Therefore, despite the larger ensemble for Random LDsplit, the Conditional LDsplit model achieves the best value for subset accuracy when $m=4$. This gives another example where subset accuracy benefits from the implementation of a label ordering strategy in LDsplit.

The performance of Conditional LDsplit dramatically deteriorates once $m \geq 3$ for the ***Medical*** and ***Enron*** data. It is only the macro-F1 measure of Conditional LDsplit that remains competitive with the Random LDsplit models fit to these datasets.

In conclusion, the results of Appendix A.4 indicate that if an appropriate value of m is selected for Conditional LDsplit (in most cases a small value of $m \leq 3$), the model can produce better results than a Random LDsplit model of the same size. Moreover, a level of performance similar to or even better than that of Random LDsplit can be achieved by a Conditional LDsplit model with a smaller ensemble. As stated in Section 4.2, overfitting does not seem to be a concern for large values of M and it is therefore recommended to set M large to reach high levels of performance. Unfortunately, this strategy could be problematic if computation cost is a concern. However, it seems that the Conditional LDsplit model may hold an advantage here. By implementing the conditional entropy-based ordering strategy of Jun *et al.* (2019), L^{order} is limited to $\binom{K}{m}$ label orders and contains fewer elements than the larger collection of random label orders of L^m, $\frac{K!}{(K-m)!}$. Therefore, by fitting a Conditional LDsplit model, the maximum value of M is reduced. However, not only is the maximum value of M reduced, a level of performance similar to or even better than that of Random LDsplit can be achieved by a Conditional LDsplit model with a smaller ensemble. Some datasets also show that subset accuracy benefits from the implementation of a label ordering strategy in LDsplit.

4.4 Comparison of LDsplit models to other multi-label learning methods

In this section a comparative study is conducted on the six benchmark datasets. The aim of the study is to experimentally determine if LDsplit is a competitive multi-label learning method in general by comparing the performance of LDsplit to that of other previously proposed learning methods.

Up until this stage, only the training datasets of the six benchmark datasets have been used in this chapter. By using the appropriate values for m and M as determined in Section 4.2, a single R-LDsplit_SVM, C-LDsplit_SVM, R-LDsplit_Tree and C-LDsplit_Tree model are fit to the standard training datasets of each of the six benchmark datasets. Note that the entire set of training data is used to fit these models in each case. The fitted models are used to classify the standard test data after which performance is evaluated using standard multi-label evaluation measures as defined in Section 2.4. Different methods might favour the optimisation of certain evaluation measures. Therefore, for an unbiased view of performance, one needs to consider multiple evaluation measures in a comparative study. In this study the results of six example-based evaluation measures (Hamming loss, accuracy, precision, recall, F-score and subset accuracy), as well as six label-based evaluation measures (macro-precision, macro-recall, macro-F1, micro-precision, micro-recall and micro-F1) are considered.

A summary of the fitted LDsplit models is given in Table 4.3. For *Corel5k* the $M_{\min}$ values are given with the actual value of M in brackets. Table 4.3 also includes the time taken in minutes to train (*i.e.* learn the model on the training data) and test (*i.e.* classify the test data) each model. The execution times are dependent on the machine used for fitting the model, the values of m and M, as well as the binary classifier used for LDsplit. Despite these dependencies, the times are provided in Table 4.3 to give practitioners an indication of the time needed to train and test LDsplit models. Note that the machine used to train and test each model has CPU at 3 GHz and 50 GB RAM.

Table 4.3
LDsplit models with train and test time in minutes for each benchmark dataset

Dataset	**Model**	m	M	Training time (in minutes)	Testing time (in minutes)
Emotions	Random LDsplit (SVM)	5	100	0.600	0.050
	Conditional LDsplit (SVM)	3	20	0.083	0.017
	Random LDsplit (Decision Tree)	4	240	1.033	0.300
	Conditional LDsplit (Decision Tree)	3	20	0.050	0.017
Scene	Random LDsplit (SVM)	6	100	10.233	1.283
	Conditional LDsplit (SVM)	4	15	1.300	0.150
	Random LDsplit (Decision Tree)	6	300	14.833	1.517
	Conditional LDsplit (Decision Tree)	4	15	0.533	0.050
Yeast	Random LDsplit (SVM)	3	200	17.167	1.267
	Conditional LDsplit (SVM)	4	100	11.650	0.850
	Random LDsplit (Decision Tree)	6	300	12.433	0.917
	Conditional LDsplit (Decision Tree)	4	1000	23.533	1.067
Medical	Random LDsplit (SVM)	6	500	6.100	0.417
	Conditional LDsplit (SVM)	2	500	2.867	0.167
	Random LDsplit (Decision Tree)	4	500	13.700	3.267
	Conditional LDsplit (Decision Tree)	2	100	1.550	0.35
Enron	Random LDsplit (SVM)	3	100	21.350	1.283
	Conditional LDsplit (SVM)	3	100	22.817	1.383
	Random LDsplit (Decision Tree)	6	80	7.517	0.783
	Conditional LDsplit (Decision Tree)	6	100	8.883	0.817
Corel5k	Random LDsplit (SVM)	2	500 (593)	307.950	3.850
	Conditional LDsplit (SVM)	2	500 (516)	271.450	3.450
	Random LDsplit (Decision Tree)	2	500 (600)	16.100	3.133
	Conditional LDsplit (Decision Tree)	2	500 (518)	13.883	2.583

The comparative study compares the predictive performance of the LDsplit models in Table 4.3 to that of a collection of previously proposed multi-label learning methods. To present a comparative study of a high standard, the collection of previously proposed multi-label models should be carefully constructed. Firstly, based on their previous usage in the multi-label literature, one must ensure that the most well-known methods in the community are included in the comparative study. Secondly, as LDsplit is a new tree-based ensemble method, it is important that it is compared to a range of previously proposed tree-based methods. Lastly, it is important that state-of-the-art methods are included which have been shown to produce high predictive performance on average for a range of different problems. A recent study by Bogatinovski *et al.* (2022) compares the average performance of 26 multi-label models on 42 multi-label benchmark datasets. The authors identify RF-PCT, BR with a random forest base classifier, ECC with a decision tree base classifier, EBR with a decision tree base classifier and AdaBoost.MH as the best-performing methods across the spectrum of evaluation measures. Therefore, to ensure that the comparative study is unbiased, the collection of previously proposed multi-label models should include these methods.

By taking the above-mentioned criteria into consideration, the predictive performance of LDsplit is compared to that of 17 multi-label models as summarised in Table 4.4. The LDsplit models are therefore compared to the problem transformation methods BR, CC, CLR and QWML, the algorithm adaptation methods ML-kNN and AdaBoost.MH, tree-based methods such as ML-C4.5, MODT/PCT, HOMER, RFML-C4.5, RF-PCT and ML-Forest as well as other related ensemble methods namely EBR, ECC and RAkEL. For each of the 17 models, the base classifiers and parameter instantiation of the models are given in Table 4.4.

Note that although different cross-validation splits are used on the training data to determine the parameter instantiation for LDsplit and models of Madjarov *et al.* (2012), the final models are trained and tested on the same standard train and test data splits. Therefore, the LDsplit results are directly comparable to that of the learning methods considered in Madjarov *et al.* (2012) and ML-Forest in Wu *et al.* (2016). The results found by Madjarov *et al.* (2012) and Wu *et al.* (2016) are therefore used in the comparative study (as appropriately referenced in Table 4.4). In addition, BR with a random forest base classifier, ECC with a decision tree base classifier, EBR with a decision tree base classifier and AdaBoost.MH are fit to the standard training data of each benchmark dataset, since these models are not included in the study by Madjarov *et al.* (2012). The first three of these models are fit by implementing the functions *br()* and *ecc()* of the R-package *utiml* (Rivolli, 2021). These functions conveniently allow for both a random forest and decision tree base classifier. The function *adaboost()* in the R-package *JOUSBoost* (Olson, 2017) is used to fit AdaBoost.MH. The fitted models are used to classify the standard test data of each benchmark dataset after which their performance is evaluated using the standard multi-label evaluation measures. In Table 4.4, these four models are appropriately referenced as “Own”.

For each of the models in Table 4.3 and Table 4.4, performance across the 12 multi-label evaluation measures are summarised in Table 4.5, Table 4.6, Table 4.7, Table 4.8, Table 4.9 and Table 4.10 for the *Emotions*, *Scene*, *Yeast*, *Medical*, *Enron* and *Corel5k* data respectively. The model that produces the best performance for each evaluation measure is given in bold per table. Since Wu *et al.* (2016) exclude micro-precision, micro-recall and micro-F1 in their reported results, all ML-Forest models are marked with NA for these evaluation measures in the respective tables. Similarly, since *Enron* was excluded in the study by Wu *et al.* (2016), all the evaluation measures are marked with NA for the ML-Forest model in Table 4.9.

Table 4.4
Multi-label models with their corresponding base classifiers and parameter instantiations

Method	Base classifier	Parameter instantiation	Reference
BR	SVM	10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)
BR	Random Forest	$M = 100$ and $\sqrt{p}$ inputs considered per node, as recommended by Bogatinovski *et al.* (2022)	Own
CC	SVM	10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)
CLR	SVM	10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)
QWML	SVM	10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)
ML-kNN	-	Number of neighbours per dataset determined from the values 6 to 20 with step 2.	Madjarov *et al.* (2012)
AdaBoost.MH	Decision stump	$M = 300$, as recommended by Bogatinovski *et al.* (2022)	Own
ML-C4.5	-	Uses sub-tree raising as a post-pruning strategy with pruning confidence set to 0.25	Madjarov *et al.* (2012)
MODT/PCT	-	Uses a pre-pruning strategy that employs the F-test	Madjarov *et al.* (2012)
HOMER	SVM	2-6 clusters were considered in experiments and the best results reported. For SVM: 10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)
RFML-C4.5	ML-C4.5	$M = 100$ fully grown trees used with minimum node size 10 and $\lfloor \log_2(p+1) \rfloor$ inputs considered per node	Madjarov *et al.* (2012)
RF-PCT	MODT/PCT	$M = 100$ fully grown trees used with minimum node size 10 and $\lfloor 0.1p+1 \rfloor$ inputs considered per node	Madjarov *et al.* (2012)
ML-Forest	SVM	$M = 50$ trees are fit, with λ randomly selected in the range $(0.9, 0.95)$ for each tree. For SVM: linear kernel is used with Penalty= 2^{-5}.	Wu *et al.* (2016)
EBR	BR, Decision tree	$M = 100$ BR models as recommended by Bogatinovski *et al.* (2022)	Own
ECC	CC, SVM	$M = 10$ random CC models. For SVM: 10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)
ECC	CC, Decision Tree	$M = \min(2K, 50)$ random CC models as recommended by Bogatinovski *et al.* (2022)	Own
RAkEL	LP, SVM	RAkEL$_o$ is fit. $M = \min(2K, 100)$ and $k = \frac{K}{2}$. For SVM: 10-fold CV with radial basis kernel and candidates: Gamma= $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ Penalty= $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$	Madjarov *et al.* (2012)

Table 4.5

Predictive performance of 21 models fit to the *Emotions* data

	Emotions											
	Hamming loss	Accuracy	Precision	Recall	F-score	Subset-accuracy	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
R-LDsplit_SVM	0.191	0.558	0.741	0.636	0.685	**0.317**	0.751	0.626	0.673	0.743	0.639	0.687
C-LDsplit_SVM	0.191	0.550	**0.755**	0.630	**0.687**	**0.317**	0.756	0.625	**0.676**	0.747	0.637	0.687
R-LDsplit_Tree	0.211	0.470	0.727	0.531	0.613	0.252	0.737	0.539	0.615	0.742	0.549	0.631
C-LDsplit_Tree	0.226	0.472	0.661	0.556	0.604	0.223	0.680	0.558	0.608	0.690	0.569	0.624
BR (SVM)	0.257	0.361	0.550	0.409	0.469	0.129	0.721	0.378	0.440	0.684	0.406	0.509
BR (RF)	0.202	0.508	0.724	0.585	0.647	0.287	0.751	0.583	0.644	0.742	0.591	0.658
CC (SVM)	0.256	0.356	0.551	0.397	0.461	0.124	0.581	0.364	0.420	0.698	0.393	0.503
CLR (SVM)	0.257	0.361	0.538	0.410	0.465	0.144	0.677	0.381	0.443	0.685	0.409	0.512
QWML (SVM)	0.254	0.373	0.548	0.429	0.481	0.149	0.660	0.398	0.458	0.680	0.431	0.528
ML-kNN	0.294	0.319	0.502	0.377	0.431	0.084	0.518	0.334	0.385	0.584	0.376	0.457
AdaBoost.MH	0.224	0.505	0.680	0.613	0.645	0.208	0.674	0.599	0.630	0.677	0.609	0.641
ML-C4.5	0.247	0.536	0.606	0.703	0.651	0.277	0.602	0.702	0.630	0.607	0.712	0.655
MODT/PCT	0.267	0.448	0.577	0.534	0.554	0.223	0.628	0.533	0.568	0.607	0.539	0.571
HOMER	0.361	0.471	0.509	**0.775**	0.614	0.163	0.464	**0.775**	0.570	0.471	**0.782**	0.588
RFML-C4.5	0.198	0.488	0.625	0.545	0.583	0.272	**0.828**	0.532	0.620	**0.783**	0.551	0.647
RF-PCT	**0.189**	0.519	0.644	0.582	0.611	0.307	0.802	0.569	0.650	**0.783**	0.589	0.672
ML-Forest	0.256	0.429	0.662	0.489	0.524	0.168	0.604	0.477	0.525	NA	NA	NA
EBR (Tree)	0.204	0.507	0.717	0.592	0.649	0.267	0.740	0.593	0.642	0.730	0.604	0.661
ECC (SVM)	0.281	0.432	0.580	0.533	0.556	0.168	0.531	0.508	0.500	0.579	0.531	0.554
ECC (Tree)	0.195	**0.560**	0.721	0.656	**0.687**	0.307	0.719	0.648	0.674	0.722	0.664	**0.692**
RAkEL (SVM)	0.282	0.419	0.564	0.491	0.525	0.208	0.547	0.462	0.488	0.586	0.489	0.533

Table 4.6

Predictive performance of 21 models fit to the *Scene* data

	Scene											
	Hamming loss	Accuracy	Precision	Recall	F-score	Subset-accuracy	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
R-LDsplit_SVM	0.077	0.728	0.839	0.728	0.780	0.692	0.838	0.718	0.771	0.840	0.708	0.769
C-LDsplit_SVM	**0.076**	0.733	0.840	0.735	**0.784**	**0.695**	0.839	0.724	0.775	0.839	0.716	**0.773**
R-LDsplit_Tree	0.099	0.551	0.869	0.551	0.674	0.523	0.859	0.540	0.654	0.869	0.533	0.661
C-LDsplit_Tree	0.116	0.532	0.773	0.545	0.639	0.492	0.753	0.536	0.619	0.758	0.530	0.624
BR (SVM)	0.079	0.689	0.718	0.711	0.714	0.639	0.844	0.703	0.765	0.843	0.694	0.761
BR (RF)	0.091	0.568	**0.905**	0.574	0.702	0.537	0.887	0.567	0.682	0.899	0.560	0.690
CC (SVM)	0.082	0.723	0.758	0.726	0.742	0.685	0.817	0.716	0.762	0.814	0.708	0.757
CLR (SVM)	0.080	0.686	0.714	0.712	0.713	0.633	0.835	0.704	0.762	0.835	0.695	0.758
QWML (SVM)	0.081	0.683	0.711	0.709	0.710	0.630	0.832	0.701	0.759	0.832	0.692	0.756
ML-kNN	0.099	0.629	0.661	0.655	0.658	0.573	0.784	0.647	0.692	0.691	0.634	0.661
AdaBoost.MH	0.087	0.661	0.833	0.704	0.763	0.596	0.807	0.696	0.747	0.800	0.690	0.741
ML-C4.5	0.141	0.569	0.592	0.582	0.587	0.533	0.635	0.573	0.596	0.619	0.570	0.593
MODT/PCT	0.129	0.538	0.565	0.539	0.551	0.509	0.682	0.529	0.593	0.512	0.521	0.516
HOMER	0.082	0.717	0.746	0.744	0.745	0.661	0.807	0.734	0.768	0.804	0.727	0.764
RFML-C4.5	0.116	0.388	0.403	0.388	0.395	0.372	**0.963**	0.381	0.514	**0.960**	0.572	0.717
RF-PCT	0.094	0.541	0.565	0.541	0.553	0.518	0.919	0.533	0.658	0.930	0.523	0.669
ML-Forest	0.097	0.719	0.757	0.744	0.740	0.656	0.749	0.730	0.738	NA	NA	NA
EBR (Tree)	0.098	0.548	0.875	0.563	0.685	0.511	0.854	0.556	0.667	0.860	0.550	0.671
ECC (SVM)	0.085	**0.735**	0.770	**0.771**	0.771	0.665	0.785	**0.757**	0.770	0.773	**0.751**	0.762
ECC (Tree)	0.106	0.644	0.743	0.672	0.706	0.589	0.746	0.659	0.697	0.728	0.659	0.692
RAkEL (SVM)	0.077	0.734	0.768	0.740	0.754	0.694	0.835	0.727	**0.777**	0.831	0.721	0.772

Table 4.7

Predictive performance of 21 models fit to the *Yeast* data

	Yeast											
	Hamming loss	Accuracy	Precision	Recall	F-score	Subset-accuracy	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
R-LDsplit_SVM	**0.187**	0.530	0.726	0.604	0.660	0.207	0.730	0.368	0.476	0.737	0.598	0.660
C-LDsplit_SVM	**0.187**	0.539	0.721	0.620	0.667	0.226	0.717	0.379	0.479	0.728	0.613	0.666
R-LDsplit_Tree	0.201	0.478	0.733	0.535	0.619	0.128	0.759	0.296	**0.480**	0.732	0.531	0.616
C-LDsplit_Tree	0.204	0.484	0.715	0.557	0.626	0.140	0.577	0.330	0.472	0.712	0.551	0.622
BR (SVM)	0.190	0.520	0.722	0.591	0.650	0.190	0.628	0.355	0.392	0.733	0.587	0.652
BR (RF)	0.194	0.501	0.732	0.563	0.636	0.156	**0.776**	0.319	0.411	0.739	0.557	0.635
CC (SVM)	0.193	0.527	0.727	0.600	0.657	**0.239**	0.602	0.357	0.390	0.726	0.588	0.650
CLR (SVM)	0.190	0.524	0.719	0.601	0.655	0.195	0.614	0.361	0.392	0.729	0.595	0.655
QWML (SVM)	0.191	0.523	0.718	0.600	0.654	0.192	0.614	0.361	0.394	0.727	0.595	0.654
ML-kNN	0.198	0.492	0.732	0.549	0.628	0.159	0.600	0.308	0.336	0.736	0.543	0.625
AdaBoost.MH	0.210	0.492	0.684	0.581	0.628	0.156	0.527	0.364	0.458	0.684	0.573	0.624
ML-C4.5	0.234	0.480	0.620	0.608	0.614	0.158	0.377	0.375	0.370	0.618	0.603	0.610
MODT/PCT	0.219	0.440	0.705	0.490	0.578	0.152	0.479	0.269	0.293	0.698	0.492	0.577
HOMER	0.207	**0.559**	0.663	**0.714**	**0.687**	0.213	0.471	**0.466**	0.447	0.647	**0.702**	**0.673**
RFML-C4.5	0.205	0.453	0.738	0.491	0.589	0.129	0.533	0.257	0.283	0.747	0.491	0.593
RF-PCT	0.197	0.478	**0.744**	0.523	0.614	0.152	0.674	0.286	0.322	**0.755**	0.521	0.617
ML-Forest	0.199	0.501	0.717	0.573	0.609	0.166	0.393	0.320	0.333	NA	NA	NA
EBR (Tree)	0.197	0.497	0.728	0.558	0.632	0.161	0.716	0.319	0.413	0.733	0.555	0.631
ECC (SVM)	0.207	0.546	0.667	0.673	0.670	0.215	0.391	0.388	0.350	0.662	0.655	0.658
ECC (Tree)	0.201	0.501	0.708	0.573	0.633	0.181	0.738	0.326	0.403	0.712	0.569	0.633
RAkEL (SVM)	0.192	0.531	0.715	0.615	0.661	0.201	0.480	0.352	0.359	0.720	0.602	0.656

Table 4.8

Predictive performance of 21 models fit to the *Medical* data

	Medical											
	Hamming loss	Accuracy	Precision	Recall	F-score	Subset-accuracy	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
R-LDsplit_SVM	0.011	0.740	0.865	0.797	0.830	0.639	0.774	0.397	0.827	0.826	0.787	0.806
C-LDsplit_SVM	0.011	0.740	0.861	0.800	0.829	0.636	0.735	0.399	0.833	0.820	0.789	0.804
R-LDsplit_Tree	0.010	0.761	0.876	0.808	0.841	0.669	0.868	0.364	**0.853**	0.838	0.784	0.810
C-LDsplit_Tree	0.010	0.760	0.870	0.810	0.839	0.666	0.828	0.369	0.815	0.830	0.787	0.808
BR (SVM)	0.077	0.206	0.211	0.735	0.328	0.000	0.399	0.423	0.361	0.225	0.725	0.343
BR (RF)	0.015	0.530	**0.914**	0.531	0.672	0.471	**0.927**	0.197	0.635	**0.913**	0.526	0.668
CC (SVM)	0.077	0.211	0.217	0.754	0.337	0.000	0.391	0.428	0.371	0.229	0.739	0.350
CLR (SVM)	0.017	0.656	0.695	0.795	0.742	0.486	0.288	0.307	0.281	0.669	0.782	0.721
QWML (SVM)	0.012	0.658	0.697	0.801	0.745	0.480	0.285	0.324	0.286	0.667	0.787	0.722
ML-kNN	0.017	0.528	0.575	0.547	0.560	0.462	0.267	0.163	0.192	0.807	0.522	0.634
AdaBoost.MH	**0.009**	**0.785**	0.883	**0.835**	**0.858**	**0.700**	0.778	**0.525**	0.783	0.844	**0.816**	**0.830**
ML-C4.5	0.013	0.730	0.797	0.740	0.768	0.646	0.263	0.249	0.250	0.796	0.720	0.756
MODT/PCT	0.023	0.228	0.285	0.227	0.253	0.177	0.018	0.022	0.020	0.826	0.227	0.356
HOMER	0.012	0.713	0.762	0.760	0.761	0.610	0.287	0.282	0.282	0.807	0.742	0.773
RFML-C4.5	0.022	0.250	0.284	0.251	0.267	0.216	0.190	0.040	0.058	0.884	0.237	0.374
RF-PCT	0.014	0.591	0.635	0.599	0.616	0.538	0.269	0.176	0.207	0.885	0.569	0.693
ML-Forest	0.012	0.759	0.845	0.782	0.795	0.654	0.367	0.327	0.336	NA	NA	NA
EBR (Tree)	0.010	0.757	0.871	0.801	0.835	0.667	0.868	0.363	0.852	0.839	0.775	0.806
ECC (SVM)	0.014	0.611	0.662	0.642	0.652	0.526	0.266	0.179	0.203	0.834	0.624	0.714
ECC (Tree)	0.011	0.696	0.878	0.714	0.787	0.622	0.886	0.267	0.783	0.864	0.701	0.774
RAkEL (SVM)	0.012	0.673	0.730	0.679	0.704	0.607	0.269	0.183	0.210	0.881	0.600	0.714

Table 4.9

Predictive performance of 21 models fit to the *Enron* data

	Enron											
	Hamming loss	Accuracy	Precision	Recall	F-score	Subset-accuracy	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
R-LDsplit_SVM	0.046	0.436	0.740	0.486	0.587	0.147	0.417	0.133	0.414	0.726	0.453	0.558
C-LDsplit_SVM	0.045	0.439	0.743	0.488	0.589	0.149	0.458	0.128	0.398	0.729	0.456	0.561
R-LDsplit_Tree	0.049	0.397	0.681	0.461	0.550	0.085	0.557	0.106	0.362	0.665	0.453	0.539
C-LDsplit_Tree	0.049	0.417	0.675	0.475	0.558	0.118	0.433	0.114	0.347	0.671	0.445	0.535
BR (SVM)	0.045	0.446	0.703	0.497	0.582	0.149	0.258	0.120	0.143	0.721	0.464	0.564
BR (RF)	0.046	0.441	0.731	0.493	0.589	0.135	0.399	0.142	0.366	0.714	0.467	0.564
CC (SVM)	0.064	0.334	0.464	0.507	0.484	0.000	0.260	0.146	0.153	0.492	0.472	0.482
CLR (SVM)	0.048	0.459	0.650	0.557	0.600	0.117	0.205	0.139	0.149	0.652	0.532	0.585
QWML (SVM)	0.048	0.388	0.624	0.453	0.525	0.097	0.242	0.120	0.143	0.687	0.438	0.535
ML-kNN	0.051	0.319	0.587	0.358	0.445	0.062	0.170	0.075	0.087	0.684	0.353	0.466
AdaBoost.MH	0.048	0.407	0.707	0.477	0.570	0.111	0.357	0.154	0.363	0.686	0.459	0.550
ML-C4.5	0.047	0.418	0.623	0.487	0.546	0.140	0.142	0.107	0.115	0.613	0.440	0.512
MODT/PCT	0.058	0.196	0.415	0.229	0.295	0.002	0.023	0.030	0.026	0.601	0.246	0.349
HOMER	0.051	0.478	0.616	0.610	0.613	0.145	0.241	0.163	0.167	0.597	0.585	0.591
RFML-C4.5	0.053	0.374	0.690	0.398	0.505	0.124	0.245	0.082	0.102	0.768	0.366	0.496
RF-PCT	0.046	0.416	0.709	0.452	0.552	0.131	0.233	0.100	0.122	0.738	0.422	0.537
ML-Forest	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
EBR (Tree)	0.046	0.456	0.701	0.521	0.598	0.135	0.594	0.126	0.424	0.698	0.494	0.578
ECC (SVM)	0.049	0.462	0.652	0.560	0.602	0.131	0.249	0.129	0.140	0.642	0.532	0.582
ECC (Tree)	0.046	0.448	0.686	0.504	0.581	0.147	0.602	0.103	0.466	0.705	0.471	0.565
RAkEL (SVM)	0.045	0.428	0.708	0.469	0.564	0.136	0.222	0.097	0.115	0.743	0.435	0.548

Table 4.10

Predictive performance of 21 models fit to the *Corel5k* data

	Corel5k											
	Hamming loss	Accuracy	Precision	Recall	F-score	Subset-accuracy	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
R-LDsplit_SVM	0.009	0.070	0.580	0.073	0.130	0.008	0.709	0.029	0.345	0.646	0.076	0.135
C-LDsplit_SVM	0.009	0.069	0.603	0.072	0.128	0.008	0.734	0.028	0.344	0.663	0.074	0.133
R-LDsplit_Tree	0.009	0.022	0.524	0.022	0.043	0.000	0.487	0.009	0.312	0.554	0.023	0.045
C-LDsplit_Tree	0.009	0.022	0.524	0.022	0.043	0.000	0.487	0.009	0.312	0.554	0.023	0.045
BR (SVM)	0.017	0.030	0.042	0.055	0.047	0.000	0.052	0.023	0.021	0.061	0.057	0.059
BR (RF)	0.010	0.049	0.260	0.051	0.085	0.000	0.737	0.006	0.247	0.275	0.052	0.087
CC (SVM)	0.017	0.030	0.042	0.056	0.048	0.000	0.053	0.023	0.021	0.061	0.057	0.059
CLR (SVM)	0.012	0.195	0.329	0.264	0.293	0.010	0.059	0.039	0.042	0.338	0.258	0.293
QWML (SVM)	0.012	0.195	0.326	0.264	0.292	0.012	0.059	0.039	0.042	0.339	0.258	0.293
ML-kNN	0.009	0.014	0.035	0.014	0.021	0.000	0.031	0.006	0.010	0.730	0.015	0.030
AdaBoost.MH	0.009	0.037	0.490	0.039	0.072	0.000	0.435	0.020	0.260	0.525	0.041	0.077
ML-C4.5	0.010	0.002	0.005	0.002	0.003	0.000	0.004	0.005	0.008	0.160	0.002	0.004
MODT/PCT	0.009	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
HOMER	0.012	0.179	0.317	0.250	0.280	0.002	0.044	0.041	0.036	0.308	0.248	0.275
RFML-C4.5	0.009	0.005	0.018	0.005	0.008	0.008	0.007	0.001	0.001	0.750	0.005	0.010
RF-PCT	0.009	0.009	0.030	0.009	0.014	0.000	0.015	0.002	0.004	0.696	0.009	0.018
ML-Forest	0.010	0.119	0.268	0.142	0.172	0.010	0.052	0.030	0.035	NA	NA	NA
EBR (Tree)	0.009	0.016	0.552	0.017	0.032	0.000	0.561	0.005	0.239	0.556	0.017	0.033
ECC (SVM)	0.009	0.001	0.002	0.001	0.001	0.001	0.001	0.001	0.001	0.333	0.001	0.002
ECC (Tree)	0.009	0.006	0.455	0.006	0.012	0.000	0.756	0.002	0.168	0.478	0.006	0.012
RAkEL (SVM)	0.009	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The corrected Friedman test (Iman and Davenport, 1980) and the post-hoc Nemenyi test (Nemenyi, 1963) assess the overall differences in performance across the datasets. These tests are often used in multi-label comparative studies such as Madjarov *et al.* (2012) and Bogatinovski *et al.* (2022). The tests determine if the differences in performance of the LDsplit models are statistically significant.

The Friedman test is a non-parametric test for multiple hypotheses testing. For each dataset, the respective models are ranked according to their performance on a specific evaluation measure. The best performing model is allocated a rank of 1, the second-best performing model is allocated a rank of 2, and so on until all the models are ranked. In situations where ties occur, average ranks are assigned. The mean ranks of the models are then compared by calculating the Friedman statistic. Iman and Davenport (1980) propose a corrected F-statistic that is distributed according to the F-distribution with $A-1$ and $(A-1)(B-1)$ degrees of freedom. Here A denotes the number of models and B denotes the number of datasets. If a statistically significant difference in performance is found, the post-hoc Nemenyi test determines between which models those differences appear. To implement this test a critical distance is computed. The critical distance depends on the number of models, the number of datasets and a critical value based on the Studentised range statistic computed for a given significance level. Two classifiers are significantly different if their average ranks differ by more than the critical distance. In this work the R-package *tsutils* (Kourentzes *et al.*, 2022) is used to implement these tests as detailed by Hollander *et al.* (2014) at a confidence level of 0.95. The critical distance of the Nemenyi test is therefore calculated as:

$$q_{0.95\,,A}\sqrt{\frac{A(A+1)}{6B}}\,,$$

where the Studentised range statistic for infinite degrees of freedom divided by $\sqrt{2}$ is used to calculate $q_{0.95\,,A}$.

In this study the results of the Friedman and Nemenyi tests are summarised with mean rank diagrams. Such a diagram vertically lists the respective models such that the best ranking model is at the bottom of the diagram. The diagram gives the mean rank (denoted by a dot) and the bounded critical distance (denoted with a horizontal line to the left and right of the mean rank) of each model. If the bounded critical distances of two models cross, there is no evidence of a statistically significant difference between the models. However, if the bounded critical distances of two models do not cross, there is evidence that the two models perform significantly different for the evaluation measure.

Figure 4.3 and Figure 4.4 give the mean rank diagrams for the respective evaluation measures. Since no ML-Forest results are available for the *Enron* data, the diagrams in Figure 4.3 and Figure 4.4 are based on the Friedman and Nemenyi tests conducted using all the models of Tables 4.5 – Table 4.10 except ML-Forest. Consequently, the tests are based on the performance of $A = 20$ models across $B = 6$ datasets. To assess the overall differences in performance when ML-Forest is included in the analysis, the *Enron* data must be excluded. Consequently, Appendix A.5 gives the mean rank diagrams based on the Friedman and Nemenyi tests conducted using the performance of $A = 21$ models across $B = 5$ datasets. Since no ML-Forest results are available for micro-precision, micro-recall and micro-F1, no mean rank diagrams are given for these evaluation measures in Appendix A.5.

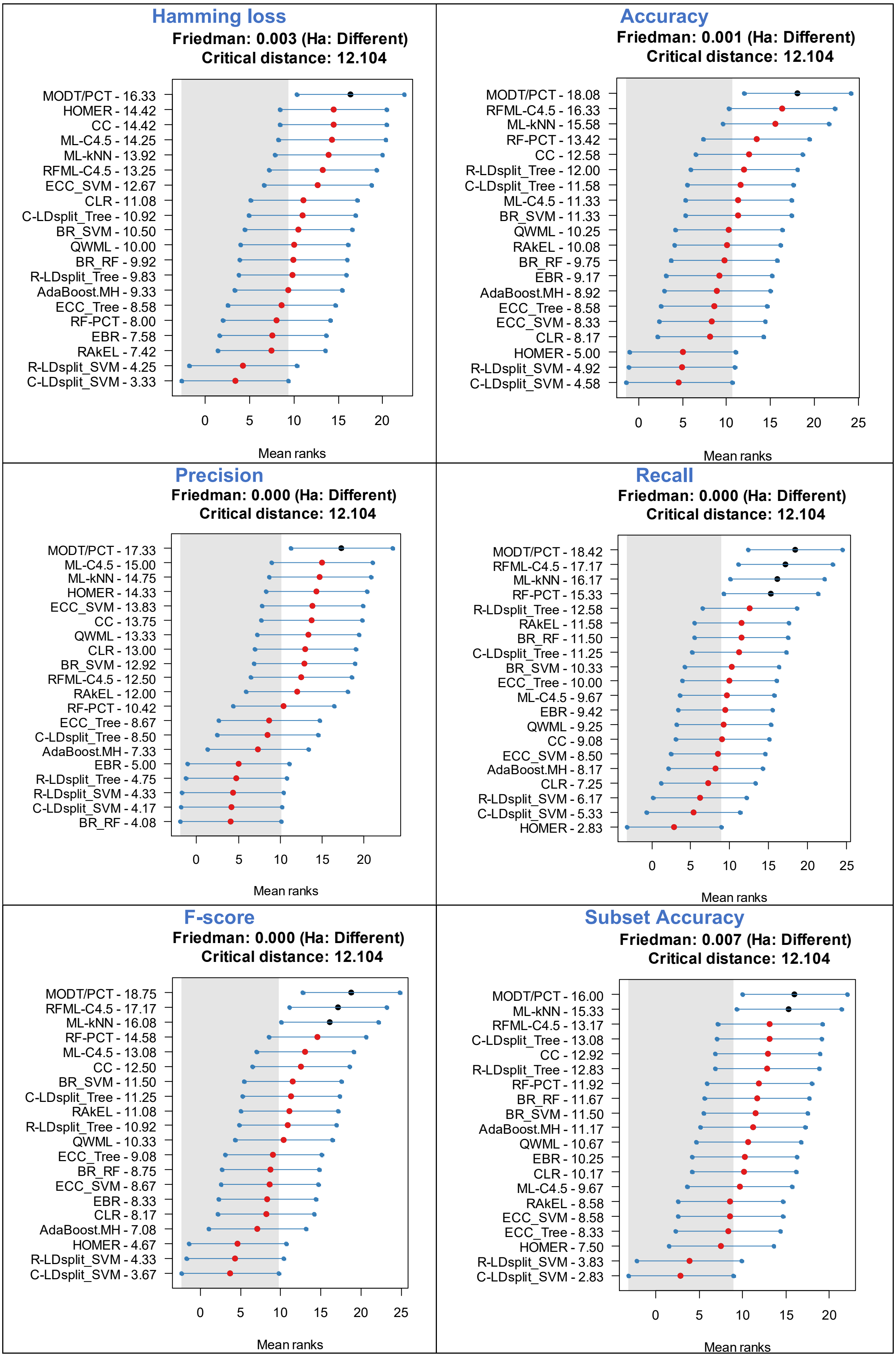


Figure 4.3 Mean rank diagrams for the example-based evaluation measures

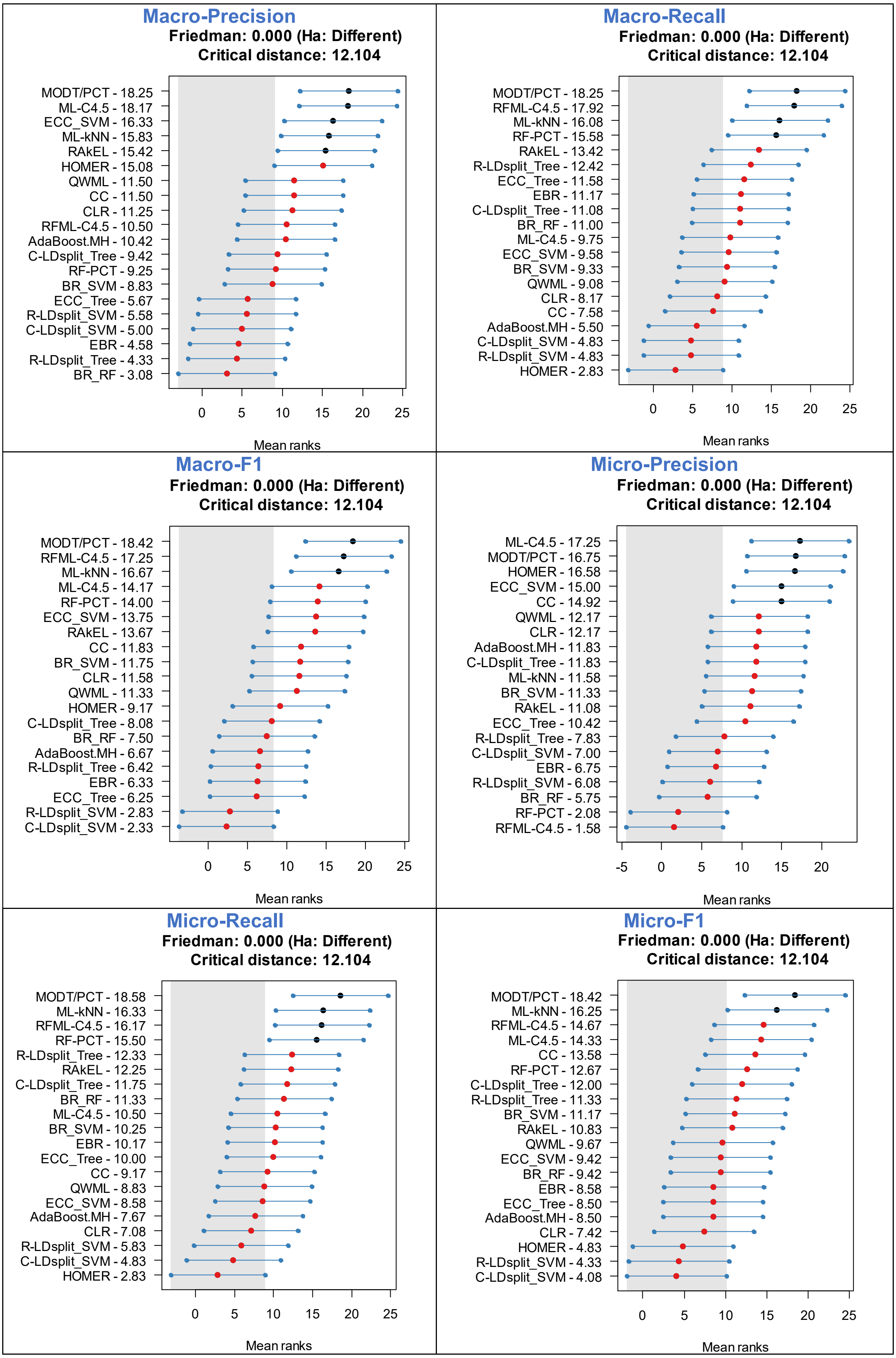


Figure 4.4 Mean rank diagrams for the label-based evaluation measures

In terms of general predictive performance across all six benchmark datasets, the Random and Conditional LDsplit models with an SVM base classifier are very satisfactory.

Figure 4.3 and Figure 4.4 illustrate that for six of the evaluation measures namely Hamming loss, accuracy, F-score, subset accuracy, macro-F1 and micro-F1, C-LDsplit_SVM and R-LDsplit_SVM are the first- and second-best performing models in terms of mean rank. It is noteworthy that for Hamming loss, subset accuracy and macro-F1, the mean ranks achieved by the LDsplit models with an SVM base classifier are much better than the mean rank of the third highest ranking model. Although these are not statistically significant differences, this illustrates that the LDsplit models are consistently highly ranked for Hamming loss, subset accuracy and macro-F1, unlike any of the other models of this study. Since subset accuracy is a very strict evaluation measure that requires the predicted set of labels to be an exact match to the true set of labels, it is interesting that the LDsplit models are consistently highly ranked for this measure.

Furthermore, C-LDsplit_SVM and R-LDsplit_SVM achieve the second- and third-best mean ranks for four evaluation measures namely precision, recall, macro-recall and micro-recall. In other words, C-LDsplit_SVM and R-LDsplit_SVM are found within the top three models in terms of mean rank for 10 of the 12 evaluation measures considered in the study. R-LDsplit_SVM and C-LDsplit_SVM are also within the top five models in terms of mean rank for macro-precision. This makes C-LDsplit_SVM and R-LDsplit_SVM the top two performing models on average across the six benchmark datasets.

The above remarks also agree with the results of Appendix A.5, since C-LDsplit_SVM and R-LDsplit_SVM are the first- and second-best performing models for Hamming loss, accuracy, F-score, subset accuracy and macro-F1 in terms of mean rank. Furthermore, C-LDsplit_SVM and R-LDsplit_SVM achieve the second- and third-best mean ranks for recall and macro-recall and are found within of the top five models for precision and macro-precision. In addition, R-LDsplit_Tree achieves the best and second-best mean rank for precision and macro-precision respectively. Note that no mean rank diagrams are available for micro-precision, micro-recall and micro-F1 in Appendix A.5. In addition, Appendix A.5 shows that ML-Forest is ranked below C-LDsplit_SVM and R-LDsplit_SVM on average for all evaluation measures. Therefore, unless otherwise stated, the remainder of this discussion refers to the mean rank diagrams of Figure 4.3 and Figure 4.4.

In general, Figure 4.3 and Figure 4.4 show that LDsplit models with an SVM base classifier achieve better performance than the previously proposed tree-based methods ML-C4.5, MODT/PCT, RFML-C4.5 and RF-PCT. In fact, when compared to MODT, C-LDsplit_SVM achieves a statistically significant improvement in performance for all evaluation measures except micro-precision. In addition, R-LDsplit_SVM achieves a statistically significant improvement in performance for all evaluation measures except micro-precision and Hamming loss when compared to MODT. Furthermore, both C-LDsplit_SVM and R-LDsplit_SVM achieve statistically significant improvements in performance compared to RFML-C4.5 for F-score, macro-recall and macro-F1. Compared to ML-C4.5, both C-LDsplit_SVM and R-LDsplit_SVM achieve a statistically significant improvement in macro-precision. Although the differences in performance between RF-PCT and LDsplit models with an SVM base classifier are not statistically significant, the LDsplit models show better mean ranks for many of the evaluation measures. For example, while the LDsplit models are found within the top three ranked models for accuracy, recall, F-score, macro-recall, and micro-recall, RF-PCT is found within the bottom four models for these measures. The ensembles of multi-label trees, RFML-C4.5 and RF-PCT, do however achieve the best mean ranks for micro-precision. In fact, RFML-C4.5 significantly outperforms ML-C4.5, PCT, HOMER, ECC_SVM and CC, while RF-PCT significantly outperforms ML-C4.5, PCT and HOMER. High precision indicates that the labels that are classified as present by the models are in truth present. However, since micro-averaging uses the sum of all true positives across all labels, micro-precision can be dominated by common labels. R-LDsplit_SVM and C-LDsplit_SVM achieve the fourth- and sixth-best mean rank values for micro-precision.

Apart from the above-mentioned statistically significant improvements in performance of LDsplit compared to tree-based methods, LDsplit also significantly outperforms the algorithm adaptation method ML-kNN. Compared to ML-kNN, R-LDsplit_SVM and C-LDsplit_SVM achieve statistically significant improvements in F-score. In addition, C-LDsplit_SVM also achieves statistically significant improvements in performance compared to ML-kNN for subset accuracy, macro-F1 and micro-F1.

The tree-based method, HOMER, achieves the best mean rank for recall, macro-recall and micro-recall while LDsplit models with an SVM base classifier achieve the second- and third-best mean ranks for these measures. Upon closer inspection, the results of Table 4.5, Table 4.7 and Table 4.9 show that HOMER achieves the highest recall, macro-recall and micro-recall for *Emotions*, *Yeast* and *Enron*, while also achieving high performance for the other datasets considering these three measures. Conversely, LDsplit models achieve high performance for precision, macro-precision and micro-precision across the different datasets. High recall indicates that a HOMER model classifies many of the originally relevant labels as present, therefore causing a small number of false negatives. At the same time, some non-relevant labels are classified as present which therefore results in a larger number of false positives and lower precision. Compared to HOMER, LDsplit models are more conservative to classify labels present. In most cases, labels that are predicted as present by an LDsplit model are truly relevant. This allows for a smaller number of false positives and higher precision than HOMER. However, this conservative approach results in a larger number of false negatives and lower recall than HOMER. Since C-LDsplit_SVM and R-LDsplit_SVM achieve the best and second-best mean ranks for F-score, macro-F1 and micro-F1, the LDsplit models allow for a better trade-off between precision and recall than HOMER in this study.

The above remarks confirm that C-LDsplit_SVM and R-LDsplit_SVM are highly competitive with state-of-the-art multi-label learning methods in terms of predictive performance. For the remainder of this section the general performance of the four LDsplit models are compared.

Since C-LDsplit_SVM and R-LDsplit_SVM are the top two performing models on average according to Figure 4.3 and Figure 4.4, it is clear that LDsplit models with an SVM base classifier generally achieve better performance than those with a decision tree base classifier. This also corresponds to the conclusion of the cross-validation study performed on the training data in Section 4.2. In Section 4.2, when a decision tree base classifier is used instead of an SVM base classifier, performance only improves for the *Medical* data. The same is true for the test data results of Table 4.8. Considering the test performance of the four LDsplit models on the *Medical* data, apart from recall, macro-recall and micro-recall, R-LDsplit_Tree achieves the highest performance for all the evaluation measures. R-LDsplit_Tree also outperforms many of the other methods on the *Medical* data. AdaBoost.MH however remains the overall best-performing model on the *Medical* data. In general, these results motivate that a cross-validation study, such as that performed in Section 4.2, can be used to select an appropriate base classifier for LDsplit. However, this is a computationally expensive strategy.

Even though an SVM base classifier is a better choice in general for LDsplit, according to the Friedman and Nemenyi tests, the LDsplit models with a decision tree base classifier are not significantly outperformed by any of the models in this study. In fact, R-LDsplit_Tree achieves a statistically significant improvement in performance for macro-precision compared to the tree-based methods MODT and ML-C4.5.

Figure 4.3 and Figure 4.4 show that the general difference in performance between Random and Conditional LDsplit models with an SVM base classifier is very small. Considering all the evaluation measures, the mean ranks of R-LDsplit_SVM and C-LDsplit_SVM do not differ by more than 1. C-LDsplit_SVM however achieves the better mean rank for most of the evaluation measures. According to the Friedman and Nemenyi tests, C-LDsplit_SVM also significantly outperforms more models for more evaluation measures than R-LDsplit_SVM. As seen in Table 4.3, this good performance of C-LDsplit_SVM is achieved by a smaller LDsplit model which requires less training time than R-LDsplit_SVM in most cases. Only for the *Enron* data did C-LDsplit_SVM take longer to learn than R-LDsplit_SVM. However, this additional time is less than 1.5 minutes. Therefore in practice, C-LDsplit_SVM may not only lead to slightly better predictive performance than R-LDsplit_SVM, the model may also require less computation time.

Random and Conditional LDsplit models with a decision tree base classifier show some differences in performance for precision, macro-precision and micro-precision. However, these differences are not statistically significant. Apart from these measures, the mean ranks of R-LDsplit_Tree and C-LDsplit_Tree do not differ by more than 2 in Figure 4.3 and Figure 4.4.

The mean ranks of R-LDsplit_SVM and R-LDsplit_Tree, as well as that of C-LDsplit_SVM and C-LDsplit_Tree differ by between 4 and 9 ranks in most cases. This illustrates that the difference in performance observed for two LDsplit models with different base classifiers is more severe than the difference in performance observed for Random LDsplit and Conditional LDsplit. It can therefore be concluded that the performance of LDsplit is very dependent on the chosen binary base classifier.

4.5 Conclusion

In this chapter a three-stage empirical study was performed on six multi-label benchmark datasets.

In the first stage the standard training datasets of the six benchmark datasets were used to examine the influence of the LDsplit tuning parameters, m and M. In general, for both Random and Conditional LDsplit, overfitting does not seem to be a concern for large values of M. The Random LDsplit results show that for each value of m, performance stabilises when M is large. Furthermore, for large M, in most cases only small differences in performance occur when the number of tree-levels increases. For Conditional LDsplit the cross-validation results show that for some datasets large m can improve performance, but for other datasets large m can cause a drastic decrease in performance. Therefore, since the results are dependent on the multi-label dataset at hand, an adequate way to determine m and M in practice is by means of cross-validation. However, as a general rule of thumb, m should be set small ($3 \leq m \leq 4$) while M is large with respect to K. For smaller datasets it may be possible to set M to its maximum value of $\binom{K}{m}$ when fitting Conditional LDsplit.

In the second stage of this study, performance of Random and Conditional LDsplit were compared considering the training data. Results show that a Conditional LDsplit model with an appropriately small value of m (in most cases $m \leq 3$) can produce better results than a Random LDsplit model of the same size. Moreover, a level of performance similar or even better than that of Random LDsplit can be achieved by a Conditional LDsplit model that fits a smaller ensemble. Therefore, by fitting a Conditional LDsplit model, the maximum value of M is not only reduced, a small Conditional LDsplit ensemble can also achieve similar or better performance than a large Random LDsplit ensemble. Some datasets also show that subset accuracy benefits from the implementation of a label ordering strategy in LDsplit.

Since computation time allowed for it in this study, appropriate values of m and M were determined per dataset for Random and Conditional LDsplit by means of cross-validation. These values were used to fit four LDsplit models per training dataset namely R-LDsplit_SVM, C-LDsplit_SVM, R-LDsplit_Tree and C-LDsplit_Tree. Predictive performances of these models on the test datasets were compared to that of 17 other multi-label learning methods in the final stage of the empirical study. To assess the overall differences in performance for each of the 12 evaluation measures, the corrected Friedman and post-hoc Nemenyi tests were used.

Results show that the predictive performance of LDsplit models across all six benchmark datasets is very satisfactory. However, LDsplit models with an SVM base classifier generally achieve better performance than those with a decision tree base classifier. Even though this is the case, the LDsplit models with a decision tree base classifier are not significantly outperformed by any of the models in this study.

In terms of mean rank, C-LDsplit_SVM and R-LDsplit_SVM are the first- and second-best performing models for six of the evaluation measures and are found within the top three models for ten of the evaluation measures. This makes C-LDsplit_SVM and R-LDsplit_SVM the top two performing models on average in the empirical study. It is apparent that C-LDsplit_SVM and R-LDsplit_SVM consistently achieve good ranks for Hamming loss, subset accuracy and macro-F1 across all datasets.

Furthermore, C-LDsplit_SVM and R-LDsplit_SVM perform significantly better than the tree-based methods ML-C4.5, MODT and RFML-C4.5 as well as the algorithm adaptation method, ML-kNN, for many evaluation measures. Results also show that while LDsplit models achieve high precision in general, HOMER achieves high recall. LDsplit models are therefore more conservative to classify labels present than HOMER. However, since C-LDsplit_SVM and R-LDsplit_SVM achieve the best and second-best mean ranks for F-score, macro-F1 and micro-F1, the LDsplit models generally allow for a better trade-off between precision and recall than HOMER.

C-LDsplit_SVM achieves slightly better performance on the test data and required less training time than R-LDsplit_SVM in most cases. Even though this is the case, results show that when the tuning parameters are appropriately determined, Random and Conditional LDsplit do not differ markedly. A much larger inequality in performance is observed between the different base classifiers of LDsplit. Therefore, attention in practice should be given to ensure that an appropriate base classifier is selected for LDsplit.

LDsplit uses hierarchical label structures to implicitly exploit local higher-order label correlations of multi-label data in a simple manner. This chapter illustrates that despite its simplicity, LDsplit is highly competitive with state-of-the-art multi-label learning methods in terms of predictive performance. In the next chapter a further challenging aspect of multi-label data is considered, as described in Section 1.4.2, namely that of variable importance and variable selection within multi-label data. The next chapter outlines an additional advantage of LDsplit, namely that LDsplit can be useful for inference regarding global and local importance of variables.

Chapter 5: Multi-label variable importance and variable selection - existing methods and new approaches based on LDsplit

5.1 Introduction

It is desirable that learning methods achieve good predictive performance on unseen observations. However, another desirable property of a learning method is to produce interpretable results. Models are more interpretable if the relationship between the input variables and the response variables can be described by the model in an understandable way. For example, a traditional single-label tree (as described in Section 2.7.3) is highly interpretable since the entire model can be represented as a two-dimensional tree graphic such as that of Figure 2.8. Other models, for example a neural network, may not produce highly interpretable results even if the model can achieve good predictive performance (Hastie *et al*., 2009:352).

In the previous chapters LDsplit is introduced as a new tree-based ensemble method that implicitly exploits local higher-order label correlation to aid in achieving accurate multi-label classification. The empirical evaluation of LDsplit confirms that the approach is competitive with state-of-the-art multi-label learning methods in terms of predictive performance. In this chapter the discussion of LDsplit is extended beyond its satisfying predictive performance by considering aspects of variable importance and variable selection.

As LDsplit is a tree-based method for multi-label data, variable importance measures of single-label trees can be extended to multi-label data by using the LDsplit framework. Consequently, traditional variable importance measures for single-label trees are discussed in Section 5.2. Hereafter, to illustrate how the challenging aspects of variable importance and variable selection within multi-label data are currently addressed, approaches for variable importance and variable selection in multi-label data are described in Section 5.3 and Section 5.4 respectively.

In contrast to single-label data, multi-label data allow for a distinction between global and local importance of variables. An input variable is globally important if it is deemed important for several or all labels. However, an input variable can also be deemed locally important for a specific label. In addition to classification, an LDsplit ensemble can also provide inference regarding global and local importance of input variables. Consequently, Section 5.5 outlines ways in which an LDsplit ensemble can produce global and local input variable rankings.

Apart from interpretability, an additional contribution of the LDsplit variable ranking strategies is that measures for both global and local importance of variables are given. Few proposals for measuring global and local importance of variables are found in the multi-label literature. Furthermore, since LDsplit implicitly exploits local higher-order label correlations when an ensemble of tree-structures is fit, the variable importance measures derived for LDsplit are influenced by the correlations between labels. This is advantageous since the few previously proposed multi-label variable importance measures, such as those based on the problem transformation paradigm, mostly neglect to exploit possible label dependencies. In Section 5.6 these and other advantages of the input variable ranking strategies of LDsplit are discussed. Furthermore, it is explained how the input variable rankings of LDsplit can be used to perform variable selection. In this setting the strategy is regarded as a pre-processing method to reduce the dimensionality of a multi-label dataset. Technically, after the dimensionality of the dataset has been reduced by means of the LDsplit variable selection method, any multi-label learning algorithm can be applied to the reduced set. The LDsplit variable selection method therefore qualifies as a so-called filter approach since it reduces the input space independently of the learning algorithm used for classification. In contrast to many other filter approaches for multi-label data, the LDsplit approach advantageously utilises correlations between labels.

5.2 Traditional variable importance measures for single-label trees

Two major variable importance measures have been derived from ensembles of single-label trees (Scornet, 2020). The first is based on improvements in node impurity after splitting the tree-nodes. This measure is sometimes referred to as Mean Decrease Impurity (MDI). The second measure is based on prediction accuracy before and after input variables are permuted. This second strategy is sometimes referred to as Mean Decrease Accuracy (MDA). The following two sections summarise these two well-known variable importance measures for single-label trees.

5.2.1 Variable importance based on Mean Decrease Impurity (MDI)

Assume that for a single-label tree, T, at node t an input variable, $X_{v(t)}$, and corresponding split-point, split the node into two new nodes. The chosen splitting variable and split-point are the variable and split-point that allow for maximal estimated improvement in node impurity after the split. If the impurity of the node which contains N observations is for example denoted by ϑ, and the impurity of its two children nodes of sizes N_1 and N_2 (where $N = N_1 + N_2$) are denoted by ϑ_1 and ϑ_2, the improvement in impurity is defined by $\vartheta - \left(\frac{N_1}{N}\vartheta_1 + \frac{N_2}{N}\vartheta_2\right)$. For binary classification, examples of measures of node impurity include misclassification error, the Gini index and Shannon entropy.

Breiman *et al.* (1984) propose a measure of relative importance of input variables, X_l, $l = 1,\dots,p$, in predicting the class response in a single-label tree. In this setting, for a single-label decision tree, T, the measure of relevance for an input variable X_l is given as:

$$I_l(T) = \sum_{t=1}^{J-1} \hat{\imath}_t \,\mathrm{I}\left(v(t) = l\right).$$

Here $I_l(T)$ denotes the relative importance of variable X_l computed using the single-label tree, T. This measure sums over the $J-1$ internal nodes of T. The improvement in node impurity after the split on variable $X_{v(t)}$ is denoted by $\hat{\imath}_t$. Therefore, the relative importance of variable X_l is the sum of improvements over all internal nodes where X_l is the splitting variable. Note that the Gini index is often used in the literature as a measure of node impurity when $I_l(T)$ is computed.

If a bagged or random forest ensemble of M single-label trees is fit, T_j, $j=1,...,M$, the relative importance of each variable for each tree in the ensemble is computed and denoted by $I_l(T_j)$, where $l=1,...,p$ and $j=1,...,M$. In this case the relative importance of variable X_l across the ensemble is simply the average of the relative importance computed for each tree. Therefore:

$$I_l = \tfrac{1}{M}\sum_{j=1}^{M} I_l(T_j).$$

5.2.2 Variable importance based on Mean Decrease Accuracy (MDA)

During the bootstrap stage of fitting a bagged or random forest ensemble of single-label trees, a so-called Out-Of-Bag (OOB) set is formed per tree. The OOB set of a tree includes all the observations in the dataset not included in the bootstrap sample used to fit the tree. An OOB observation can be used to compute an OOB error. This is done as follows. For each observation $\mathbf{w}_i = (\mathbf{x}_i, y_i)$, $i=1,...,N$ obtain the classification given by the ensemble by averaging only those trees corresponding to bootstrap samples in which $\mathbf{w}_i$ did not appear (Hastie *et al.*, 2009:593). This classification is then compared to the true classification of $\mathbf{w}_i$ and a classification error (for example, misclassification error) is calculated.

As outlined in Breiman (2001), OOB observations can also be used to define a measure of variable importance. To do so the following is done for each tree in the ensemble.

For a fitted tree, the OOB set of the tree is dropped into the root node. The observations filter down the tree by following the splitting rules until each observation reaches a terminal node. The classifications defined by the terminal nodes are compared to the true classifications of the OOB set and a classification error rate is calculated. Now, to calculate the importance of X_l for this tree, the values of X_l in the OOB set are permuted (*i.e.* rearranged) while $X_1,...,X_{l-1},X_{l+1},...,X_p$ are left unchanged. This permuted OOB set is dropped into the root node of the fitted tree to obtain the classifications of the OOB set after X_l has been permuted. Once again, these classifications are compared to the true classifications of the OOB set and a classification error rate is calculated.

For each variable, X_l , $l=1,...,p$, the difference in classification accuracy, before and after the variable is permuted, is computed per tree. Breiman (2001) defines the importance of variable X_l as the average difference in accuracy computed over the ensemble.

The rationale is that by permuting X_l, its relationship with the response is broken. Therefore, a large increase in the classification error of the OOB set after X_l is permuted, indicates that X_l is an important variable for accurate classification.

In Section 1.4.2 the two related concepts of variable importance and variable selection are outlined. The section also highlights that, in comparison to single-label data, additional complexity arises for variable importance and variable selection within multi-label data. In the following two sections brief summaries are given of previous approaches to variable importance and variable selection for multi-label data. First, approaches for variable importance in multi-label data are given in Section 5.3 and thereafter the discussion is extended to variable selection in Section 5.4. After this literature review is complete, Section 5.5 proposes ways in which an LDsplit ensemble can produce global and local input variable rankings.

5.3 Approaches for variable importance within multi-label data

In this dissertation, input variables that provide some useful information for the classifier are defined as relevant, so that input variables that do not provide any useful information are regarded as irrelevant. In some cases an input variable may also be redundant. This means that although the variable provides some useful information for the classifier, the information is already contained in one or more of the other input variables in the dataset. In multi-label data a distinction can also be made between globally relevant variables and locally relevant variables. An input variable is globally relevant if it is deemed important for several or all labels. However, an input variable can also be deemed locally relevant for a specific label.

In practice the true relevancies of input variables are unknown. Different approaches therefore exist that aim to evaluate input variables in terms of their importance. Spolaôr *et al*. (2013) state that the importance of input variables can be evaluated in two main ways, namely individual evaluation and subset evaluation, as shown in Figure 5.1.

With individual evaluation each input variable is assessed individually and assigned a weight according to its degree of class prediction. The degree of class prediction can be measured by different relevancy measures. Examples of such relevancy measures are given in Section 5.3.1 below. The weight or relevance score of an input variable assesses how dependent the response values are on the input variable (Petković *et al.*, 2020). The relevance scores of input variables therefore provide a measure of variable importance. Variables that obtain a larger relevance score are regarded as more important. An input variable ranking is produced by ranking all input variables from most to least important based on their relevance scores. Subset evaluation, on the other hand, defines the evaluation measures for a subset of input variables, instead of evaluating each input variable individually. Consequently, the approach is more computationally expensive than individual evaluation. However, the disadvantage of the individual evaluation approach is that it does not provide an explicit distinction between relevant, irrelevant and redundant variables within the variable ranking.

In Section 5.5 input variable ranking strategies based on LDsplit are given. Consequently, the remainder of this section focuses on previously proposed strategies for input variable ranking within multi-label data. Petković *et al.* (2020) state that similar to multi-label classification, approaches for input variable ranking within multi-label data can be divided into problem transformation approaches and algorithm adaptation approaches. Problem transformation approaches transform the multi-label data so that a single-label variable ranking method can be applied. This strategy is discussed in more detail in Section 5.3.1 below. On the other hand, algorithm adaptation methods are directly applied to the multi-label data to produce an input variable ranking. Examples of such methods are given in Section 5.3.2.

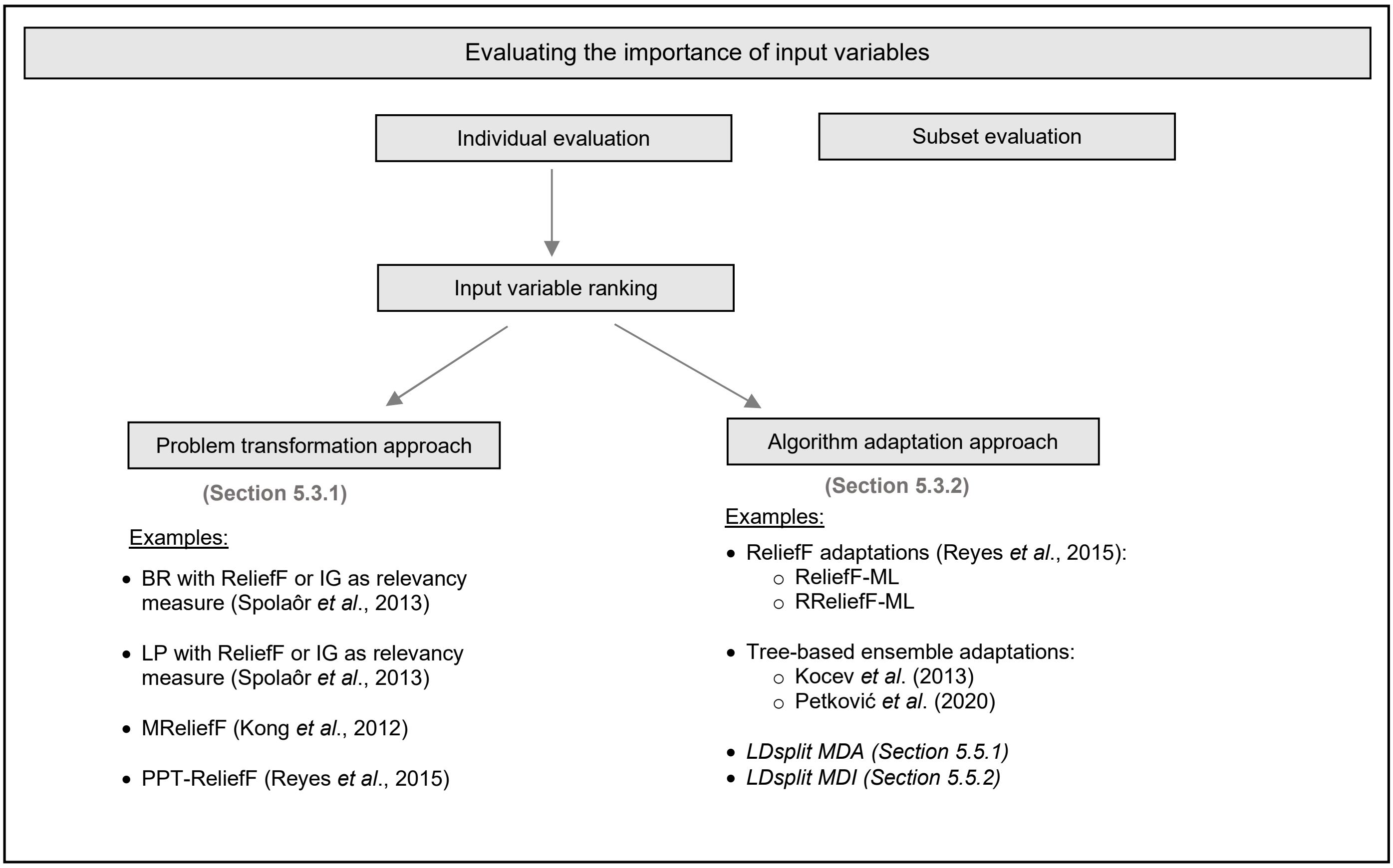


Figure 5.1 Approaches for evaluating the importance of input variables

5.3.1 Variable importance measures based on problem transformations

Variable importance measures for multi-label data often follow a problem transformation approach. Firstly, the multi-label data are transformed so that single-label variable importance measures can be implemented. Spolaôr *et al*. (2013) apply BR and LP (described in Section 2.7.2) to the multi-label data, whereas Kong *et al*. (2012) decompose the multi-label problem into a set of pair-wise single-label problems, and Reyes *et al*. (2015) use the Pruned Problem Transformation (PPT) method proposed in Read (2008). The relevance of each input variable for the single-label response is then quantified. Several previously proposed relevancy measures can be used to quantify the importance of an input variable for classification. Some of the most popular of these measures include ReliefF and Information Gain (IG). These measures are described below. Further examples include the Fisher score (Sun *et al*., 2021), chi-square score (Chen and Chen, 2011), Gini index (as described in Section 5.2.1 (Breiman *et al*., 1984)), mutual information (Doquire and Verleysen, 2013), category contribution (Zhang and Duan, 2019) and rough sets (Liu *et al*., 2018). Most measures do not make use of dependencies amongst the input variables (Petković *et al*., 2020). However, it is worth mentioning that ReliefF has the advantage of taking dependencies among input variables into account.

- **Single-label relevancy measures**

In this section two well-known relevancy measures are described, namely ReliefF and IG.

The original Relief algorithm (Kira and Rendell, 1992) is limited to binary classification problems. **ReliefF** (Kononenko, 1994) extends the Relief algorithm to the multi-class scenario. For each input variable, X_l, $l = 1,...,p$, ReliefF gives a weight in the interval $[-1 , 1]$ as output. The larger the weight assigned to the input variable, the more useful the input variable according to the ReliefF algorithm (Reyes *et al*., 2015). Suppose that a randomly selected data instance, $\mathbf{x}_r$, has $Y = y$. By using Euclidean distance, the t nearest neighbours of $\mathbf{x}_r$ are identified that have $Y = y$. This set of nearest neighbours are referred to as "hits". Similarly, for all other classes (excluding the class for which $Y = y$) the t nearest neighbours of $\mathbf{x}_r$ are identified. This set of neighbours are referred to as “misses”. Robnik-Šikonja and Kononenko (2003) state that for most purposes t can be safely set to 10.

Suppose that $\mathbf{x}_r$ has $X_l = x$. In essence, ReliefF considers X_l important for Y if the hit-values of $\mathbf{x}_r$ have X_l values that are close to x, and the miss-values of $\mathbf{x}_r$ have X_l values that are far away from x. In other words, X_l is rewarded if it gives different values for misses and similar values for hits, whereas X_l is penalised if it gives similar values for misses and different values for hits. By randomly selecting different instances, $\mathbf{x}_r$, $r = 1,...,R$, the ReliefF algorithm is iterated for a user-specified number of times, R. The detailed ReliefF algorithm is available in Robnik-Šikonja and Kononenko (2003).

Turning to **IG**, this is a single-label relevancy measure often used in papers related to multi-label variable importance and selection (Spolaôr *et al.*, 2013). Let D denote a single-label classification dataset with discrete input variables X_l, $l = 1,..., p$ and let the response variable Y consist of G disjoint classes. Then the entropy of D is calculated as:

$$entropy(D) = -\sum_{g=1}^{G} p_g(Y)\log p_g(Y).$$

In the above expression $p_g(Y)$ denotes the probability (relative frequency) of the g^{th} class. The IG for each of X_l, $l = 1,..., p$ for Y is obtained by calculating the difference between the entropy of the dataset D and the weighted sum of the entropy of each subset $D_{(v)} \subseteq D$, where $D_{(v)}$ is composed of the data cases where X_l has the value v (Zdravevski *et al.*, 2015). In other words, if X_l consists of 20 distinct values, the weighted sum is applied to 20 different $D_{(v)}$ datasets (Spolaôr *et al.*, 2013). Therefore, IG for X_l, calculated on the single-label classification dataset D, is defined by:

$$IG\left(D, X_l\right) = entropy(D) - \sum_{X_l = v} \frac{\left|D_{(v)}\right|}{|D|} entropy\left(D_{(v)}\right).$$

In the above expression $|D|$ denotes the number of data cases in D, and $\left|D_{(v)}\right|$ denotes the number of data cases in $D_{(v)}$. Furthermore note that $entropy\left(D_{(v)}\right)$ gives the entropy calculated over all data cases that have $X_l = v$. A large value of $IG\left(D, X_l\right)$ signifies that X_l is important for D.

This measure could be useful when the data are represented in the bag-of-words framework.

- **The BR approach**

When BR is used to transform the multi-label data, K binary classification datasets are formed, one for each of the K labels, $Y_1, Y_2, ..., Y_K$. For each of the p input variables, $X_1, X_2, ..., X_p$, the relevance of each of the labels, $Y_1, Y_2, ..., Y_K$, is calculated by means of a relevancy measure such as ReliefF or IG. Results can be stored in a $p \times K$ matrix $\mathbf{F}$ where entry (l, k) gives the relevance of X_l for Y_k as determined by the relevancy measure. The matrix $\mathbf{F}$ can be used to construct global and local input variable rankings.

Let $\mathbf{f}_k$ denote the k^{th} column-vector of $\mathbf{F}$. A **local** input variable ranking for Y_k can be obtained by arranging the entries of $\mathbf{f}_k$ in descending order. If for example $\mathbf{f}_k = \left[f_{1,k}\ f_{2,k}\ f_{3,k}\ f_{4,k}\right]^{\mathrm{T}}$ is arranged as $\left[f_{2,k}\ f_{1,k}\ f_{4,k}\ f_{3,k}\right]^{\mathrm{T}}$ the local input variable ranking for Y_k is X_2, X_1, X_4, X_3 with respective importance values of $f_{2,k}, f_{1,k}, f_{4,k}, f_{3,k}$.

The simplest way to obtain a **global** input variable ranking from $\mathbf{F}$ is to aggregate the importance obtained by each variable for the respective labels by taking the average importance. An average importance value is obtained for each input variable by computing the p row averages of $\mathbf{F}$. These averages reflect the relevancies of the input variables with respect to all of the labels, and can be ordered to produce a global input variable ranking. Other aggregation strategies are discussed in Spolaôr and Tsoumakas (2013).

Suppose the ReliefF algorithm is used as the relevancy measure for the BR approach. In this case the ReliefF algorithm is iterated R times for each of Y_k, $k=1,....,K$. In each case the ReliefF algorithm gives as output a weight in the interval $[-1\ ,\ 1]$ for each input variable X_l, $l=1,...,p$. The results are stored in the $p \times K$ matrix $\mathbf{F}$ so that entry (l,k) gives the relevance of X_l for Y_k.

Similarly, if IG is used as the relevancy measures for the BR approach, $IG(D_k\ ,X_l)$ can be computed for each of X_l, $l=1,...,p$ for each label Y_k, $k=1,...,K$. Once again the results can be stored in a $p \times K$ matrix $\mathbf{F}$.

Note that if BR is used to transform the data, the global and local input variable ranking strategies outlined above do not utilise label correlations.

- **The LP approach**

If LP is used to transform the multi-label data, the data are converted to multi-class data with a maximum of 2^K possible disjoint classes. Relevancy measures suitable for multi-class data can then be used to quantify the importance of each input variable for the multi-class response.

Such an LP strategy incorporates label correlation in the ranking strategy; however, only a **global** input variable ranking is produced. Furthermore, the same disadvantage of LP for classification (namely that classes can become sparse due to the large number of possible classes) could result in an unstable input variable ranking.

In general, the disadvantage of variable importance rankings based on the problem transformation paradigm is that high computation cost arises in settings with many input variables and labels.

5.3.2 Variable importance measures based on algorithm adaptations

Multi-label variable importance measures based on algorithm adaptations include all methods that produce an input variable ranking directly from the multi-label data. In this section examples of algorithm adaptation methods for multi-label input variable ranking are given, such as ReliefF (Reyes *et al*., 2015) and tree-based ensemble (Petković *et al*., 2020) adaptations. Compared to the range of problem transformation methods outlined in Section 5.3.1, few algorithm adaptation methods exist (Petković *et al*., 2020).

- **ReliefF adaptations**

Reyes *et al*. (2015) outline two strategies, ReliefF-ML and RReliefF-ML, that adapt the classic ReliefF algorithm (discussed in Section 5.3.1) to directly handle multi-label data.

ReliefF-ML adapts the ReliefF weighting mechanism of an input variable. Suppose that a randomly selected data observation $\mathbf{x}_r$ has $\mathbf{y}_r$ as its corresponding vector of multi-label classifications and $y_{r,k}$ gives the classification of Y_k for $\mathbf{x}_r$. For each relevant ($y_{r,k} = 1$) and irrelevant ($y_{r,k} = 0$) label of $\mathbf{x}_r$ a collection of t nearest neighbours of $\mathbf{x}_r$ is formed. Reyes *et al*. (2015) define the collections of hits $\left(H_r^k\right)$ and misses $\left(M_r^k\right)$ with respect to the data observation $\mathbf{x}_r$ and label Y_k by the following:

H_r^k : t nearest neighbours of $\mathbf{x}_r$ that have the relevant label Y_k of $\mathbf{x}_r$ as relevant label.

M_r^k : t nearest neighbours of $\mathbf{x}_r$ that have the irrelevant label Y_k of $\mathbf{x}_r$ as relevant label.

In other words, if $y_{r,k} = 1$, H_r^k contains the t nearest neighbours of $\mathbf{x}_r$ that have $Y_k = 1$, whereas if $y_{r,k} = 0$, M_r^k contains the t nearest neighbours of $\mathbf{x}_r$ that have $Y_k = 1$.

To determine if X_l is important for Y_k, ReliefF-ML does not only consider whether the hit-values of $\mathbf{x}_r$ have X_l values that are close to $x_{r,l}$ and whether the miss-values of $\mathbf{x}_r$ have X_l values that are far away from $x_{r,l}$. With ReliefF-ML, the distances of the labelsets are also considered when a weight of an input variable is calculated. Given two observations $\mathbf{x}_a$ and $\mathbf{x}_b$, the Hamming distance of their corresponding label vectors, $\mathbf{y}_a$ and $\mathbf{y}_b$, is calculating as:

$$d(a,b) = \tfrac{1}{K}\left|\mathbf{y}_a \,\Delta\, \mathbf{y}_b\right|.$$

Note that in the above expression Δ denotes the symmetric difference between the two sets. A small value of $d(a,b)$ represents a major similarity between $\mathbf{y}_a$ and $\mathbf{y}_b$. Based on the collections, H_r^k and M_r^k, Reyes *et al*. (2015) define the following probabilities:

$$P_{H_r^k} = \frac{1}{t}\sum_{s\in H_r^k} d(r,s)$$

and

$$P_{M_r^k} = \frac{1}{t}\sum_{s\in M_r^k} d(r,s) \ .$$

$P_{H_r^k}$ represents the probability that nearest neighbours of $\mathbf{x}_r$ that share Y_k as relevant belong to different sets of labels. Furthermore, $P_{M_r^k}$ represents the probability that nearest neighbours of $\mathbf{x}_r$ that have Y_k as relevant while $y_{r,k} = 0$ belong to different sets of labels. These probabilities are incorporated within the ReliefF-ML weighting mechanism of an input variable so that a variable obtains larger importance if it shows different values for observations with dissimilar labelsets and similar values for observations of similar labelsets. The detailed ReliefF-ML algorithm is given in Reyes *et al*. (2015).

RReliefF-ML is based on the ReliefF adaptation to regression problems (RReliefF). Reyes *et al*. (2015) outline in detail how the unified view of ReliefF for the importance of an input variable in classification and regression is extended to multi-label data. Different to the previous algorithms that form hit and miss collections, for each of the randomly selected instances, $\mathbf{x}_r$, $r = 1,...,R$, RReliefF-ML retrieves only t nearest neighbours for $\mathbf{x}_r$

considering the input space. Let N_r denote the set of nearest neighbours of $\mathbf{x}_r$. When calculating the weight of X_l, the distances between $\mathbf{x}_r$ and its nearest neighbours are considered with respect to both X_l and the label space by incorporating three probabilities within the RReliefF-ML weighting mechanism. Firstly, the probability that nearest instances have different values for X_l is computed as:

$$\frac{1}{t}\sum_{r=1}^{R}\sum_{s\in N_r}\delta\left(x_{r,l},x_{s,l}\right).$$

Here $\delta\left(x_{r,l},x_{s,l}\right)$ calculates the difference between the values of X_l considering $\mathbf{x}_r$ and one of its nearest neighbours $\mathbf{x}_s \in N_r$. Reyes *et al*. (2015) do not specify if the sign of the difference is included in the calculation. Secondly, the probability that nearest instances have different sets of labels are computed as:

$$\frac{1}{t}\sum_{r=1}^{R}\sum_{s\in N_r}d\left(r,s\right).$$

Here $d\left(r,s\right)$ calculates the Hamming distance between the label vectors $\mathbf{y}_r$ and $\mathbf{y}_s$. Finally, the probability that nearest instances have different values for X_l and have different sets of labels are computed as:

$$\frac{1}{t}\sum_{r=1}^{R}\sum_{s\in N_r}\delta\left(x_{r,l},x_{s,l}\right).\,d\left(r,s\right).$$

With RReliefF-ML, an input variable receives higher importance if it has different values for observations with dissimilar labelsets and similar values for observations with similar labelsets.

- **Tree-based ensemble adaptations**

Kocev *et al*. (2013) use a random forest of PCTs to obtain an input variable ranking directly from multi-label data. Their work is motivated by the success of input variable rankings attained from traditional single-label trees, such as those outlined in Section 5.2.

By fitting a random forest of M PCTs the OOB set of each PCT is used to calculate an importance value for each input variable X_l , $l = 1,...,p$. For each PCT in the ensemble an MDA strategy (as given in Section 5.2.2) is used as outlined next.

Each PCT in the ensemble, PCT_j, $j = 1,...,M$, is used to classify all observations in the OOB set of the PCT. These classifications are compared to the true classifications of the OOB set and an OOB-error is computed for each PCT, $Err_j(OOB)$, $j = 1,...,M$. Any multi-label evaluation measure can be used to calculate the error. In the experiments of Kocev *et al.* (2013), Err_j is computed using several different evaluation measures namely: accuracy, micro-precision, micro-recall, micro-F1, macro-precision, macro-recall, and macro-F1.

To compute the importance of X_l for PCT_j , the values of X_l in the OOB set of the PCT are permuted while $X_1,...,X_{l-1},X_{l+1},...,X_p$ are left unchanged. PCT_j is used to classify the permuted set and these classifications are compared to the true classifications to obtain an OOB-error, $Err_j(OOB(X_l))$. The importance of X_l is calculated as the increase of the OOB-error after X_l has been permuted, *i.e.* $Err_j(OOB(X_l)) - Err_j(OOB)$ gives the importance of X_l calculated using PCT_j. Note that the importance of X_l is calculated by including the sign of the difference of the evaluation measure after permuting X_l. This process is repeated for all input variables for each of the M PCTs in the ensemble. The overall importance of X_l is calculated as the average importance obtained for X_l across the M PCTs in the ensemble.

Note that the above approach of Kocev *et al*. (2013) produces a global input variable ranking. Kocev *et al*. (2013) make no mention of the possibility of local importance of input variables. Consequently, no local input variable ranking strategy is given by Kocev *et al*. (2013).

Petković *et al.* (2020) extend the work of Kocev *et al*. (2013). In their work three different ensemble strategies are used: random forests, bagged ensembles, and so-called extra trees. Extra trees are similar to random forests. For both random forests and extra trees only a random subset of input variables is considered as candidates for splitting the data at each node of the tree. In addition, extra trees only consider one random splitting-value per input variable per node. Furthermore, extra-trees do not use bootstrapping.

Apart from the three ensemble strategies, Petković *et al.* (2020) also consider three different methods for determining the importance of input variables within the different ensembles. The first is the MDA PCT strategy as proposed by Kocev *et al*. (2013) and explained above. For the second strategy, Petković *et al.* (2020) apply an MDI strategy (as given in Section 5.2.1) by using the average of the normalised Gini indices of the respective labels as a measure of node impurity when calculating the improvement in node impurity. Petković *et al.* (2020) propose a third importance strategy which calculates the importance of an input variable based on the number of times the variable is selected as the splitting variable across the ensemble. In the simplest case, the importance of X_l is given as the number of times X_l is selected as a splitting variable across the ensemble.

The approaches by Petković *et al.* (2020) discussed above give global input variable rankings. Petković *et al.* (2020) also make no mention of the possibility of local importance of input variables.

In general, the identified shortcoming of the algorithm adaptation methods for variable importance is that these methods do not consider both the global and local importance of input variables.

5.4 Approaches for variable selection within multi-label data

In the previous section approaches for variable importance within multi-label data are discussed. In this section attention shifts to the related concept of variable selection within multi-label data.

The aim of variable selection is to find a smaller set of input variables that describe the dataset as well as the full original set of input variables (Liu and Motoda, 2007). With the reduction in dimensionality, the learning process of a model is not only faster, predictive performance of the model may also improve and important interpretations can be made. Traditional variable selection techniques can generally be categorised into three approaches: wrapper, embedded and filter (Pereira *et al*., 2018). The differences between the three approaches pertain to the interaction of the variable selection process and the learning algorithm used for classification. These approaches are explained in the following section.

5.4.1 Categorisation of traditional variable selection techniques

The **wrapper** approach identifies input variables that are best suited for the specific learning algorithm that is implemented. Wrapper methods need to fit the classifier many times to assess the quality of the selected input variables. Examples of wrapper methods include the iterative methods of forward selection and backward elimination (Kohavi and John, 1997). Forward selection starts with an empty model and iteratively adds the input variable that most improves the performance of the learning method (Hastie *et al*., 2009:58). In contrast, backward elimination starts with a full model that contains all input variables and iteratively removes those input variables that has the least impact on the fit. The candidate for dropping is the variable with the smallest Z-score (Hastie *et al*., 2009:59). Wrapper approaches suffer from the disadvantage of being computationally expensive.

With the **embedded** approach, variable selection is embedded within the specific learning algorithm as part of the training process. For example, when a traditional decision tree is fit, an appropriate input variable is determined to split each node of the tree.

Filter approaches are independent of the learning algorithm. Irrelevant variables are filtered out by considering other general characteristics of the data. The importance values obtained from an input variable ranking strategy can for example be used to select appropriate input variables. This can be done by selecting a fixed number of the most important variables according to their importance values (Petković *et al*., 2020). Another possibility is to apply a threshold strategy. In this case, all input variables with importance values exceeding the threshold are selected, while the other input variables are discarded. It is however not a straightforward task to determine an appropriate number of variables to select or threshold to apply. In addition, a disadvantage of the filter approach is that redundant variables are not easily identified since these variables are likely to have similar importance values. However, the main advantage of the filter approach is that it is relatively fast and simple to implement.

These three approaches to variable selection also apply to multi-label data. Similar to multi-label classification strategies and multi-label variable ranking strategies, methods for variable selection in multi-label data can also be divided into problem transformation approaches and algorithm adaptation approaches (Pereira *et al*., 2018). With a problem transformation approach the multi-label data are divided into one or more single-label datasets by using any problem transformation method (such as those outlined in Section 2.7.2). A wrapper, embedded or filter approach appropriate for single-label data can then be applied to the transformed data. The results found in this way can either be combined to obtain one reduced multi-label dataset on which a multi-label classifier is fit, or a single-label classifier can be applied to (each of) the reduced single-label dataset(s). With an algorithm adaptation approach the variable selection method is directly applied to the multi-label data. Generally, such a technique would then be categorised as either a wrapper, embedded or filter approach depending on its interaction with the learning algorithm used for classification.

The input variable rankings of LDsplit (given in Section 5.5) can be used to perform variable selection. In this setting an LDsplit variable ranking strategy is used as a pre-processing method for variable selection. No problem transformation method is applied before the LDsplit strategy is implemented. The method can therefore be considered as an algorithm adaptation method for variable selection. Furthermore, any multi-label classifier can be applied to the reduced input variable set, so that the LDsplit variable selection strategy is independent of the learning algorithm used for classification. Consequently, the proposed method also categorises as a filter approach. Filter approaches for variable selection within multi-label data are therefore of interest in this work and are discussed in more detail in the next section. Note that the filter approach for multi-label variable selection is the most commonly used approach (Spolaôr and Tsoumakas, 2013).

5.4.2 The filter approach for multi-label variable selection

A variable importance ranking (either obtained by using a problem transformation strategy as explained in Section 5.3.1, or by using an algorithm adaptation strategy as explained in Section 5.3.2) can be used to filter input variables from a multi-label dataset.

Spolaôr *et al.* (2013) apply both BR and LP transformations and consider both ReliefF and IG as relevancy measures, thus giving rise to four global input variable rankings (obtained in the manner outlined in Section 5.3.1). In their work a conservative threshold value of 0.01 is used throughout to perform variable selection. In other words, for each ranking all input variables with a global importance value exceeding 0.01 are selected while the remaining variables are filtered out. Empirical results on benchmark datasets show that methods with ReliefF often produce a smaller subset of input variables than the ones that use IG (Spolaôr *et al.*, 2013). The satisfactory performance of ReliefF could be owed to the fact that the measure incorporates input variable interaction. In general, the fixed threshold value of 0.01 causes large variation in the number of input variables selected for the respective measures and datasets (Spolaôr *et al.*, 2013).

Reyes *et al*. (2015) compare various ReliefF algorithms and use their respective variable rankings to perform variable selection. In their empirical work, the four problem transformation methods listed in Figure 5.1 for input variable ranking are considered, as well as two algorithm adaptation methods namely ReliefF-ML and RReliefF-ML. Consequently, for each of the benchmark datasets considered by Reyes *et al*. (2015), six input variable rankings are obtained by applying the respective algorithms. To perform variable selection, Reyes *et al*. (2015) use the strategy outlined in Ruiz *et al*. (2005). Therefore, for each input variable ranking, the top 100 ranked input variables are selected to which a wrapper approach is applied to determine an appropriate subset of input variables. The wrapper approach is applied to the entire set of input variables for datasets that have $p \leq 100$. Hereafter, a multi-label classifier is fit to the full set of input variables as well as to each of the six reduced sets. The classification performance of the respective models are then compared based on four multi-label evaluation measures. Results show that the proposed methods of Reyes *et al*. (2015), namely PPT-ReliefF, ReliefF-ML and RReliefF-ML, generally perform better than the model that uses the full set of input variables.

In general, it is not a straightforward task to select an appropriate threshold for filter approaches or alternatively to determine the number of variables to be selected. As a simple example, the median importance can be used as a threshold. This would regard the top 50% of input variables as relevant. However, the (unknown) true percentage of relevant input variables may be more or less than 50%. Both Spolaôr *et al.* (2013) and Reyes *et al*. (2015) ignore this concern since they simply specify a threshold or a fixed number of variables to select. Ideally, a selection method should also be tested on simulated data, where the true number of relevant variables is known.

Work by Bi *et al*. (2003), Stoppiglia *et al*. (2003), and Tuv *et al*. (2008) consider using independent probe variables to determine appropriate cut-off points for filter approaches. The idea of independent probes is to augment the input space with several randomly generated variables that are independent of the response variable(s). These variables are referred to as probe variables. Probes can be generated using a probability distribution such as the standard normal distribution (Dreyfus and Guyon, 2006). However, Tuv *et al.* (2008) argue that permuting (*i.e.* rearranging) the values of input variables is a better strategy, as it takes into account the possible special structure of the input variables. Note that this is a similar strategy as that used for MDA of traditional single-label trees (Section 5.2.2). These previous proposals for generating probe variables for the purpose of variable selection do not consider multi-label data. However, Sandrock and Steel (2016) give an approach for generating independent probes for the purpose of variable selection for multi-label data. In order to apply their method, a $p \times K$ matrix is required, for which entry (l,k) gives the local importance of variable X_l for label Y_k. The argument is that an effective variable ranking strategy would rank relevant variables higher than the probe variables, so that the probe variables assist in determining an appropriate cut-off point.

5.5 Variable importance within LDsplit

As seen in Section 5.3, many of the current methods for input variable ranking within multi-label data are based on the problem transformation paradigm. However, these methods mostly neglect to exploit label dependencies fully. A further shortcoming identified in the literature is that few proposals consider both global and local importance of variables. In fact, no local importance measures could be found in the literature that incorporate label correlations. Consequently, this section outlines how an LDsplit ensemble can be used to address these shortcomings.

As LDsplit is a tree-based method for multi-label data, variable importance measures of single-label trees, as described in Section 5.2, can be extended to multi-label data by using the LDsplit framework. Examples of such extensions are given in Section 5.5.1 and Section 5.5.2 below.

5.5.1 LDsplit MDA variable importance

This section outlines how the single-label MDA method of random forests (Section 5.2.2) can be extended to multi-label data by means of an LDsplit ensemble.

Suppose the multi-label training dataset is given by $\{(\mathbf{x}_i, \mathbf{y}_i),\ i = 1,2,\ldots,N\}$, with N $p-$component input observations, $\mathbf{x}_i$, and their corresponding $K-$component multi-label response vectors, $\mathbf{y}_i$.

To implement an MDA strategy, each of the M tree-structures in the LDsplit ensemble, T_j, $j=1,...,M$, requires an OOB set. These sets can be obtained by basing the fit of each tree-structure on a random subsample of the N training observations. Denote the proportion of observations sampled from $\{(\mathbf{x}_i, \mathbf{y}_i), i=1,2,...,N\}$ to fit T_j by $prop_{IB}$, *i.e.* the In-Bag (IB) proportion. Those observations not included in the subsample used to fit T_j, form the OOB set of T_j denoted by $(OOB)_j$, where $j=1,...,M$. In other words, the proportion of observations that form the OOB set per tree-structure is given by $prop_{OOB}=1-prop_{IB}$. Therefore, apart from one additional step which is to subsample $prop_{IB}(N)$ training observations before fitting each of T_j, $j=1,...,M$, the LDsplit ensemble is fit as described in Chapter 3. For simplicity the Random LDsplit algorithm is used in this work; however, either of the Random or Conditional LDsplit algorithms could be implemented.

The LDsplit framework allows for the development of both a global and local MDA input variable ranking. These are described next.

- **Global importance**

To determine a global importance value for each input variable, X_l, $l=1,...,p$, proceed as follows.

For the fitted $m-$level tree-structure T_j, find the posterior probabilities of labels $P_{j,1},...,P_{j,m}$ for $(OOB)_j$ by implementing the LDsplit classification strategy outlined in Section 3.2.3. These posterior probabilities can be transformed to a set of multi-label classifications for $(OOB)_j$ over the labels, $P_{j,1},...,P_{j,m}$, by applying the predetermined thresholds, $t_{P_{j,1}},...,t_{P_{j,m}}$. By doing so, classifications are easily compared to the true set of classifications of $(OOB)_j$ over $P_{j,1},...,P_{j,m}$ by using any multi-label evaluation measure (for example Hamming loss).

Now permute the values of X_l within $(OOB)_j$ while $X_1,...,X_{l-1},X_{l+1},...,X_p$ are left unchanged. Use T_j and thresholds $t_{P_{j,1}},...,t_{P_{j,m}}$ to find the multi-label classifications of the labels $P_{j,1},...,P_{j,m}$ for this permuted set of $(OOB)_j$. This set of classifications is compared to the true set of classifications of $(OOB)_j$ by once again using the chosen multi-label evaluation measure.

The difference in the value of the evaluation measure after permuting each of X_l, $l=1,...,p$, reflects the variable importance of X_l given by T_j. The reasoning is that a large decrease in performance in terms of the evaluation measure after X_l has been permuted shows that X_l has a large influence on the success of T_j and that X_l is arguably important for T_j.

Doing this for all M tree-structures give M variable importance values for each input variable X_l, $l=1,...,p$. The global importance of X_l is given by the average of these values over the M trees.

Note that the value of the evaluation measure is not necessarily worse after a variable is permuted. Due to randomness, it is possible for the value of the evaluation measure to slightly improve after a noise variable is permuted. In this case the recorded difference is negative. To prevent such information from being lost, the importance of a variable is calculated by including the sign of the difference of the evaluation measure after permuting the variable.

The resulting global variable importance values of X_l, $l=1,...,p$, can be ranked from most to least important. They can also be transformed to positive values and can be represented on a chosen scale (for example: $0-100$).

- **Local importance**

A local importance value for each input variable and label combination is determined by tracking input variable behaviour over the ensemble for each label.

Again tree-structure T_j and thresholds, $t_{P_{j,1}},...,t_{P_{j,m}}$, are used to find the multi-label classifications of labels $P_{j,1},...,P_{j,m}$ for $(OOB)_j$ as described in Section 3.2.3. However, since the overall multi-label performance of T_j is not of interest in this case, a multi-label evaluation measure is not used to compare the classifications given by T_j to the true classifications of $(OOB)_j$. Instead, each of $P_{j,1},...,P_{j,m}$ is considered separately, and the proportion of correctly classified observations over the set of $(OOB)_j$ is noted for each label.

To find the local importance of X_l, $l=1,...,p$, permute X_l within $(OOB)_j$ while $X_1,...,X_{l-1},X_{l+1},...,X_p$ are left unchanged, and use T_j and thresholds $t_{P_{j,1}},...,t_{P_{j,m}}$ to find the multi-label classifications of labels $P_{j,1},...,P_{j,m}$ for the permuted set. The set of classifications is compared to the true set of classifications of $(OOB)_j$ by finding the proportion of correct classifications for each of the labels $P_{j,1},...,P_{j,m}$.

A large decrease in the proportion of correct classifications of a label after X_l is permuted, indicates that X_l has a large influence on the successful classification of this label. This implies that X_l should be regarded as important for such a label.

Therefore, for tree-structure T_j, the difference in proportion of correct classifications for each of the labels $P_{j,1},...,P_{j,m}$ is calculated after each of X_l, $l=1,...,p$, is permuted. The results can be represented in a $p \times m$ matrix where entry (l,s) gives the difference in proportion of correct classification of label $P_{j,s}$, $s=1,...,m$, after input variable X_l is permuted. Such a matrix is constructed for each of T_j, $j=1,...,M$. Since $P_{j,s} \in \{Y_1,...,Y_K\}$ the local importance of variable X_l for label Y_k is calculated as the average importance over all matrix entries that link variable X_l and label Y_k. The results are stored in a $p \times K$ matrix, $\mathbf{F}$, where entry (l,k) gives the local importance of variable X_l for label Y_k. To identify important input variables per label, a variable importance ranking can be constructed for each column of $\mathbf{F}$.

It is important to note that the local variable importance method outlined above implicitly considers label correlation, even though proportions of correct classifications are calculated separately per label. This is due to the label dependency of the fitted LDsplit ensemble.

5.5.2 Other approaches for variable importance within LDsplit

One main advantage of the LDsplit MDA variable importance method is that it allows any binary base classifier within LDsplit. Therefore, LDsplit MDA can provide variable importance rankings for any fitted LDsplit model, irrespective of the base classifier used. It is therefore the preferred variable importance strategy for LDsplit in this work. A direct adaptation of the MDI approach of Section 5.2.1 for LDsplit may for example require a stump (*i.e.* a tree containing only one split-point) as binary base classifier.

Even though the LDsplit MDA variable importance method is preferred in this work, this section briefly describes two additional approaches which could also be considered for LDsplit.

- **LDsplit MDI variable importance**

Node-splitting in a traditional single-label classification tree is based on one variable per node that gives maximum estimated improvement in node impurity after the split. This allows the relative importance of variable X_l in a traditional single-label tree to be defined as the sum of improvements over all internal nodes where X_l is the splitting variable.

Since each split of an LDsplit tree-structure is based on a binary classification problem involving one label, it is possible to calculate the improvement in node impurity per split based on this label. However, for LDsplit, the binary base classifier generally defines each split using all the input variables. A general variable importance approach that appropriately allocates the improvement in node impurity per split across all the input variables, may be complicated to define. A simpler way in which the MDI approach could be adapted for LDsplit is to use a stump as binary base classifier. In this way each splitting rule of LDsplit would be defined by a single input variable (given by the fitted stump) and label (given by the corresponding label of the tree-level) combination. This gives rise to a simple global, as well as a simple local variable importance measure.

In this setting, the **global** importance of variable X_l is given by the sum of improvements over all internal nodes where X_l is used for the splitting rule. On the other hand, the **local** importance of variable X_l for label Y_k is given by the sum of improvements over the collection of splitting rules that combine variable X_l and label Y_k.

- **Conditional permutation schemes for LDsplit variable importance**

Strobl *et al.* (2008) state that correlation between input variables affect the original random forest variable importance measures discussed in Section 5.2, since random forests show a preference for correlated variables. Strobl *et al.* (2008) reason that the random forest measures can be considered as measures of marginal importance even though what is of interest in most applications is the conditional effect of each variable. A positive value of the importance could correspond to deviation of independence between X_l and Y or deviation of independence between X_l and $X_1,...,X_{l-1},X_{l+1},...,X_p$. Strobl *et al.* (2008) therefore introduce a conditional permutation scheme with the aim of only measuring the impact of X_l on Y under a given correlation structure between X_l and the other input variables. This scheme permutes X_l only within groups of observations with $X_1,...,X_{l-1},X_{l+1},...,X_p =$ $x_1,...,x_{l-1},x_{l+1},...,x_p$ to preserve the correlation structure between X_l and the other input variables.

Apart from other aspects that are investigated in the synthetic study of Chapter 6, the effect of correlated input variables on the LDsplit MDA approach is also considered. The results prove it unnecessary to develop an LDsplit conditional permutation scheme.

5.6 Advantages of LDsplit MDA variable importance

This section highlights various advantages of the proposed LDsplit MDA method given in Section 5.5.1.

Many multi-label classification applications require interpretable results. As mentioned, in the medical domain, successful early interventions may prevent or delay diseases (Cheng *et al*., 2019). Furthermore, a good understanding of important input variables can also be beneficial if the data collection is difficult or expensive. In such a scenario the available resources can be directed at only the most important variables. Therefore, input variable ranking for multi-label data is an important area of research as it provides insight regarding the importance of the input variables.

Since variables can be globally as well as locally relevant, it is advantageous that the LDsplit MDA strategy provides both a global and local input variable ranking. As indicated in Section 5.3, few previously proposed variable importance measures for multi-label data consider this distinction. Moreover, both the global and local LDsplit MDA measures are influenced by label correlations since they are dependent on a fitted LDsplit ensemble which implicitly exploits local higher-order label correlations. When BR is for example applied with an appropriate relevancy measure, the resulting global and local input variable rankings are not influenced by label correlations. No local importance measure could be found in the literature that utilises label correlations.

In practice, the $p \times K$ local importance matrix, $\mathbf{F}$, produced by LDsplit MDA can be used to identify important input variables per label. This can be done by constructing a variable ranking for each column of $\mathbf{F}$. Depending on the multi-label problem at hand, it may be insightful to identify sets of input variables that are important for subsets of labels. Furthermore, insight may be found if some labels demonstrate distinct behaviour by displaying a ranking of important variables very different to that of other labels. Additionally, $\mathbf{F}$ can be used to compare the importance of a particular input variable across the different labels. In this case, a variable may be regarded as important for several labels but may show a higher importance towards a few labels in particular. Other tree-based ranking methods for multi-label data, such as those proposed by Kocev *et al*. (2013) and Petković *et al.* (2020), only provide global input variable rankings. The authors do not consider the possibility of local importance of input variables. Therefore, the above examples of insight regarding the local importance of input variables can not be obtained from those measures.

It is important to realise that by basing the fit of each LDsplit tree-structure in the ensemble on a random subsample of the training observations, the LDsplit MDA method can be implemented whilst simultaneously fitting an LDsplit model useful for classification of unseen data. This allows for computation efficiency as the LDsplit MDA method is not a stand-alone variable importance method. The LDsplit MDA method can provide valuable insight towards the importance of the different input variables, while the fitted LDsplit model can classify unseen data. Since the variable importance values obtained with LDsplit MDA are dependent on the fitted LDsplit model, the measures allow for better interpretability of the LDsplit model used for classification.

Furthermore, the LDsplit MDA variable importance method allows any binary base classifier within LDsplit which makes the method more flexible. Since no restriction exists in terms of the choice of binary base classifier, LDsplit MDA can provide variable importance rankings for any fitted LDsplit model. As seen in Chapter 4, the classification performance of LDsplit appears to be very dependent on the chosen binary base classifier. One might therefore prefer to determine a good base classifier based on classification performance on validation sets. However, irrespective of the chosen base classifier, LDsplit MDA can provide variable importance rankings for the fitted model, allowing better interpretation of the model. In contrast, an LDsplit MDI strategy can for example only provide variable rankings for LDsplit models that use a stump as binary base classifier.

When LDsplit MDA is applied to an ensemble of M LDsplit tree-structures, the OOB sets, $(OOB)_j$, $j=1,...,M$, are dropped into the respective tree-structures $p+1$ times (one unpermuted set and p permuted sets for each of X_l, $l=1,...,p$ respectively). This increases the computation cost of the strategy. Fortunately, the testing times provided in Table 4.3 obtained in the empirical study of Chapter 4 show that an LDsplit ensemble generally requires little computation time to classify sets of unseen data.

Apart from the insight that the LDsplit MDA rankings can provide in terms of variable importance, the values can also be used within so-called lazy learning algorithms. Lazy learning algorithms depend on the definition of a distance function defined on the input space that determines the $k-$nearest neighbours of a query instance (Reyes *et al*., 2015). In this case the global importance values obtained for each input variable using LDsplit MDA define an input variable weight vector. This weight vector can potentially reduce the negative impact of irrelevant variables on the distance computation of the lazy learning algorithm.

Finally, the LDsplit MDA input variable rankings can be applied as a filter approach for variable selection. Spolaôr and Tsoumakas (2013) state that the filter approach is the most common approach for multi-label variable selection. However, different from many other filter approaches for multi-label variable selection, the LDsplit MDA strategy incorporates label correlation into the selection process.

The variables can be selected based on the global and local importance values in different ways. For example, the LDsplit MDA global importance ranking can be used for variable selection by selecting a fixed number of top-ranked variables, applying a threshold, or by making use of probe variables. The local importance matrix, $\mathbf{F}$, can be used for variable selection by selecting a fixed number of variables per label or by applying a threshold strategy per label. Determining an appropriate threshold per label does remain a challenge. However, since the local importance matrix, $\mathbf{F}$, defines a $p \times K$ matrix for which entry (l,k) gives the local importance of variable X_l for label Y_k, the independent probe strategy for multi-label variable selection given by Sandrock and Steel (2016) can be applied, thereby addressing the problem of threshold specification.

Note that LDsplit MDA local variable selection strategies are different from a binary relevance approach where variable selection is performed separately for each label, because $\mathbf{F}$ of LDsplit MDA incorporates label dependencies.

All the required R-functions to apply LDsplit MDA are given in Appendix D.

5.7 Conclusion

In this chapter aspects of variable importance and variable selection within multi-label data were considered.

Firstly, by investigating the current approaches for variable importance and variable selection for multi-label data, shortcomings in the literature were identified. It was found that many variable importance measures are based on problem transformations that neglect to exploit label correlations and that few proposals exist for measuring both the global and local importance of variables. Moreover, no measures for local importance of input variables could be found in the literature that incorporate label correlations. Consequently, different strategies were given in this chapter to address these shortcomings by using an LDsplit ensemble.

Input variable ranking strategies were proposed for LDsplit by adapting traditional variable importance measures of single-label trees. Advantageously, both global and local measures could be established due to the LDsplit structure. Furthermore, the global and local measures are influenced by label correlations since they are dependent on a fitted LDsplit ensemble which implicitly exploits local higher-order label correlations.

The LDsplit MDA variable importance method is the preferred strategy of this work since it allows any binary base classifier within LDsplit. LDsplit MDA can therefore provide a variable importance ranking for any fitted LDsplit model used for classification. The strategy therefore allows for better interpretation of LDsplit models. This is advantageous since interpretability can be a sought-after property in many multi-label classification scenarios.

Finally, the LDsplit MDA input variable rankings may potentially be applied as a filter approach for variable selection. For either the global or local input variable rankings this can be done by selecting a fixed number of top-ranked variables, applying a threshold or by making use of probe variables. Advantageously, different from many other filter approaches for multi-label variable selection, LDsplit MDA incorporates label correlation into the selection process.

In the next chapter multi-label synthetic data are generated to evaluate the performance of the global and local LDsplit MDA measures. A short benchmark dataset application is also given.

Chapter 6: Empirical properties and application of LDsplit MDA

6.1 Introduction

In this chapter the performance of the LDsplit MDA strategy is investigated.

An ideal way to assess a variable importance measure is by generating synthetic data. In this way the relationship between the input variables and the response can be specified, and the variable importance measure can be accurately assessed since the true input variable relationship with the response is known. However, as mentioned in Section 1.4.3, few proposals for generating synthetic multi-label data exist in the literature due to the additional challenges faced when generating the data.

In Section 6.2 the algorithm of Sandrock and Steel (2017) is used to generate synthetic multi-label data to conduct an extensive study of the performance of LDsplit MDA. The algorithm of Sandrock and Steel (2017) is used as it has the option of specifying locally and globally relevant input variables. It is therefore possible to evaluate the proposed global and local variable importance measures of LDsplit MDA.

In Section 6.3, a short benchmark dataset application is given by applying LDsplit MDA to the benchmark dataset, *Emotions*. In this case it is unknown which variables are in truth globally and locally most important since the benchmark dataset is not simulated. However, this section illustrates how LDsplit MDA makes an LDsplit model more interpretable in terms of variable importance. Furthermore, since *Emotions* is a simple music dataset where each piece of music is labelled with the emotions it provokes, it may be possible to assess if the results appear reasonable based on intuition and background knowledge of the respective emotion labels.

6.2 LDsplit MDA synthetic study

In this section the performance of LDsplit MDA is investigated by conducting an extensive study on synthetic multi-label data.

Firstly, in Section 6.2.1, the algorithm of Sandrock and Steel (2017) is described for generating multi-label synthetic data. In Section 6.2.2 the design and configuration of the synthetic study are outlined. The global and local LDsplit MDA importance results are given and discussed in Section 6.2.3 and Section 6.2.4 respectively. Some remarks on LDsplit MDA as a method for variable selection are given in Section 6.2.5 before concluding remarks of the synthetic study are given in Section 6.2.6.

6.2.1 Generating synthetic multi-label data

Consider a multi-label data pair, $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i$ represents a vector of the input variables $X_1, X_2, ..., X_p$ and $\mathbf{y}_i$ represents a vector of the binary labels $Y_1, Y_2, ..., Y_K$. To generate $(\mathbf{x}_i, \mathbf{y}_i)$, one can either start by generating $\mathbf{x}_i$ from its marginal distribution followed by generating $\mathbf{y}_i$ from its conditional distribution given $\mathbf{x}_i$, or $\mathbf{y}_i$ can be generated first from its marginal distribution followed by generating $\mathbf{x}_i$ from its conditional distribution given $\mathbf{y}_i$. The algorithm of Sandrock and Steel (2017) starts by generating the label matrix, $\mathbf{Y}$, consisting of N $K-$component label vectors. This is done using the auto-regressive or AR(1) approach of Oman (2009) for generating correlated Bernoulli variables. The equi-correlation amongst the label variables is user-specified and given by ρ_Y. The K label densities for each of $Y_1, Y_2, ..., Y_K$ are also user-specified. Sandrock and Steel (2017) denote $P(Y_k = 1)$ by p_k, where $k = 1, ..., K$.

Once the label space has been generated, a conditional multivariate normal distribution is used by Sandrock and Steel (2017) to generate the input matrix, $\mathbf{X}$, consisting of N $p-$component input observations. The mean vector of the multivariate normal distribution depends on the label space, whereas the covariance matrix is constant. The equi-correlation between the (relevant) input variables is given by ρ_X.

To ensure the mean vector specification reflects the dependency between the labels and the input variables, a $p \times K$ binary matrix $\mathbf{A}$ is introduced. The interpretation of this matrix is that if entry (l,k) is equal to 1, X_l is relevant for Y_k, otherwise this entry of the matrix equals 0. In this case "relevant" means that the distribution of X_l will be different if $Y_k = 1$ from that when $Y_k = 0$. The dependence is modelled by taking:

$$\mu_l(\mathbf{y}_i) = E(X_l | \mathbf{y}_i)$$

$$= c\sum_{k=1}^{K} a_{l,k}\, y_{i,k}\,, \quad l = 1,\ldots,p\,.$$

Therefore, by setting $a_{l,k} = 0$ when X_l is irrelevant for Y_k, $\mu_l(\mathbf{y}_i)$ does not depend on $y_{i,k}$ in such cases. On the other hand setting $a_{l,k} = 1$ when X_l is relevant for Y_k allows $\mu_l(\mathbf{y}_i)$ to depend on $y_{i,k}$, increasing by a positive quantity c if $y_{i,k}$ changes from 0 to 1.

Since c controls the extent to which $y_{i,k}$ changing from 0 to 1 influences $\mu_l(\mathbf{y}_i)$, c is used to regulate the strength of the signal of the input variables for $\mathbf{Y}$. The signal is defined as $s^2 = \left(\sum_{l=1}^{p} Var\left[E(X_l | \mathbf{Y})\right]\right)^2$. Using the fact that $E(X_l | \mathbf{y}_i) = c\sum_{k=1}^{K} a_{l,k} y_{i,k}$, $l = 1,\ldots,p$ and by completing some algebraic extensions, Sandrock and Steel (2017) give the signal as an expression which is dependent on c, $\mathbf{A}$ and ρ_Y. Since it is undesirable for the signal to depend on ρ_Y, Sandrock and Steel (2017) calculate c in such a way that the user-specified signal, s^2, is independent of ρ_Y.

The proposal is to calculate c using:

$$c = s \div \sum_{l=1}^{p} \left\{ \sum_{k=1}^{K} a_{l,k} p_k + \rho_Y \sum_{\substack{k=1 \\ k \neq r}}^{K} \sum_{r=1}^{K} a_{l,k} a_{l,r} \left[p_k \left(1 - p_k\right) p_r \left(1 - p_r\right) \right]^{\frac{1}{2}} + \sum_{\substack{k=1 \\ k \neq r}}^{K} \sum_{r=1}^{K} a_{l,k} a_{l,r} p_k p_r - \left(\sum_{k=1}^{K} a_{l,k} p_k \right)^2 \right\}^{\frac{1}{2}}$$

where $p_k = P\left(Y_k = 1\right)$, $k = 1, ..., K$.

It appears that a general problem when generating correlated Bernoulli variables is that the realised correlations differ from those specified once the univariate probabilities are not equal. This seems to be inevitable (Oman, 2009). Note that for all cases where the univariate label probabilities are set to be unequal, this issue is taken into account when the results of the synthetic study are discussed in Section 6.2.3 and Section 6.2.4. In general, this problem illustrates one of the many challenges faced when generating synthetic multi-label data. It also highlights two challenging research areas not covered in this work, namely imbalanced multi-label data generation and variable selection within imbalanced data. Research on variable selection within imbalanced data is limited and the few existing works mainly focus on single-label data. Dfuf *et al.* (2020), Chen *et al.* (2019) and Yin *et al.* (2013) give information regarding variable selection within imbalanced single-label data, and Xie *et al.* (2017) give more information regarding variable selection within imbalanced multi-label data.

6.2.2 Design and configurations of synthetic study

To investigate the performance of the LDsplit MDA approach (Section 5.5.1), the method outlined in Section 6.2.1 above is used to generate several synthetic multi-label datasets. All the generated datasets have $K = 6$ labels, $N = 2000$ observations and $p = 10$ input variables. Of the 10 input variables, only the first six are relevant, so that variables X_7, X_8, X_9 and X_{10} are noise variables. The four irrelevant variables are generated from a standard normal distribution independently of the label variables. The following matrix, $\mathbf{A}$, is used to specify the input variable relevancies that are fixed throughout:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Defining $\mathbf{A}$ in this way reflects the irrelevance of variables $X_7,...,X_{10}$. It also specifies that X_1 is locally only important for Y_1, X_2 is locally important for both Y_1 and Y_2, X_3 is locally important for Y_1, Y_2 and Y_3 and so forth.

Since X_6 is locally important for all six labels, X_6 is regarded as globally most important, followed by $X_5, X_4, ..., X_1$.

To investigate the performance of the LDsplit MDA approach in different settings, a total of 96 configurations are considered, as given in Table 6.1. The influence that the label densities, label correlation (ρ_Y), input variable correlation (ρ_X), and the strength of the signal of the data (s^2) have on performance, are examined. The influence of the choice of LDsplit tuning parameters m and M are also considered.

Table 6.1
Configurations of the 96 cases considered

				Label densities			
				$[0.2\ \ 0.2\ \ 0.2\ \ 0.2\ \ 0.2\ \ 0.2]$		$[0.2\ \ 0.3\ \ 0.4\ \ 0.2\ \ 0.3\ \ 0.4]$	
m	M	signal	ρ_X	$\rho_Y = 0$	$\rho_Y = 0.6$	$\rho_Y = 0$	$\rho_Y = 0.6$
$m = 3$	50	$s^2 = 10$	$\rho_X = 0$	Case 1	Case 2	Case 3	Case 4
			$\rho_X = 0.6$	Case 5	Case 6	Case 7	Case 8
		$s^2 = 100$	$\rho_X = 0$	Case 9	Case 10	Case 11	Case 12
			$\rho_X = 0.6$	Case 13	Case 14	Case 15	Case 16
	100	$s^2 = 10$	$\rho_X = 0$	Case 17	Case 18	Case 19	Case 20
			$\rho_X = 0.6$	Case 21	Case 22	Case 23	Case 24
		$s^2 = 100$	$\rho_X = 0$	Case 25	Case 26	Case 27	Case 28
			$\rho_X = 0.6$	Case 29	Case 30	Case 31	Case 32
$m = 4$	50	$s^2 = 10$	$\rho_X = 0$	Case 33	Case 34	Case 35	Case 36
			$\rho_X = 0.6$	Case 37	Case 38	Case 39	Case 40
		$s^2 = 100$	$\rho_X = 0$	Case 41	Case 42	Case 43	Case 44
			$\rho_X = 0.6$	Case 45	Case 46	Case 47	Case 48
	100	$s^2 = 10$	$\rho_X = 0$	Case 49	Case 50	Case 51	Case 52
			$\rho_X = 0.6$	Case 53	Case 54	Case 55	Case 56
		$s^2 = 100$	$\rho_X = 0$	Case 57	Case 58	Case 59	Case 60
			$\rho_X = 0.6$	Case 61	Case 62	Case 63	Case 64
$m = 5$	50	$s^2 = 10$	$\rho_X = 0$	Case 65	Case 66	Case 67	Case 68
			$\rho_X = 0.6$	Case 69	Case 70	Case 71	Case 72
		$s^2 = 100$	$\rho_X = 0$	Case 73	Case 74	Case 75	Case 76
			$\rho_X = 0.6$	Case 77	Case 78	Case 79	Case 80
	100	$s^2 = 10$	$\rho_X = 0$	Case 81	Case 82	Case 83	Case 84
			$\rho_X = 0.6$	Case 85	Case 86	Case 87	Case 88
		$s^2 = 100$	$\rho_X = 0$	Case 89	Case 90	Case 91	Case 92
			$\rho_X = 0.6$	Case 93	Case 94	Case 95	Case 96

For each case given in Table 6.1, the following steps are performed. Fifty multi-label datasets are generated based on the configurations, after which a Random LDsplit model with an SVM binary base classifier is fit to each by setting m and M as specified. The default SVM-function arguments of the R-package *e1071* (Meyer *et* al., 2022) are used that scales the data to zero mean and unit variance and use a radial basis kernel with $\gamma = \frac{1}{10}$ and $C = 1$. The fit of each LDsplit tree-structure is based on a random subsample of two thirds of the total generated observations, allowing one third of the observations to be used as the OOB set of the tree, *i.e.* $prop_{OOB} = \frac{1}{3}$. For simplicity, the LDsplit minimum node-size is fixed throughout at $n = 5$ and a threshold of 0.5 is used for all labels. After the specified Random LDsplit model has been fit to a synthetic dataset, the LDsplit MDA approach is applied to the ensemble. For the global importance calculations, Hamming loss is used as the multi-label evaluation measure. This produce a global importance value for each input variable as well as a local importance $p \times K$ matrix, $\mathbf{F}$, for each of the 50 synthetic datasets of a case.

In the next section the global importance results are summarised and discussed. Hereafter the local importance results are summarised and discussed in Section 6.2.4.

6.2.3 Global importance results and remarks

Figure 6.1, Figure 6.2 and Figure 6.3 summarise the average global importance results of variables $X_1,...,X_{10}$ for Cases $1-32$, $33-64$ and $65-96$.

In these figures, for each input variable the value associated with a case is found as follows. First the average global importance of the input variable across the 50 synthetic datasets of that case is calculated. After having done this for all 10 input variables, across all 96 cases, the average values are transformed by assigning the lowest a value of 0 and shifting the others accordingly. Figure 6.1, Figure 6.2 and Figure 6.3 show the results for $m = 3$, $m = 4$ and $m = 5$ respectively. To ease the general interpretation, for each variable, its mean importance over the 32 scenarios represented in the figure is indicated in red.

By representing the global importance results in this way, the average global importance values achieved by the different input variables can be compared per case. This allows the opportunity to examine per case if the LDsplit MDA method can detect the true global importance relationship between input variables. In addition, by comparing performance across different cases, insight is found into how data signal, input variable correlation, label correlation, label densities and the choice of m and M can affect performance.

For each input variable the standard deviation of global importance across the 50 synthetic datasets of the case was also calculated. The results indicate that the global importance obtained for each input variable does not vary substantially across the 50 synthetic datasets of the case. These standard deviations range between 0.0002 and 0.0113.

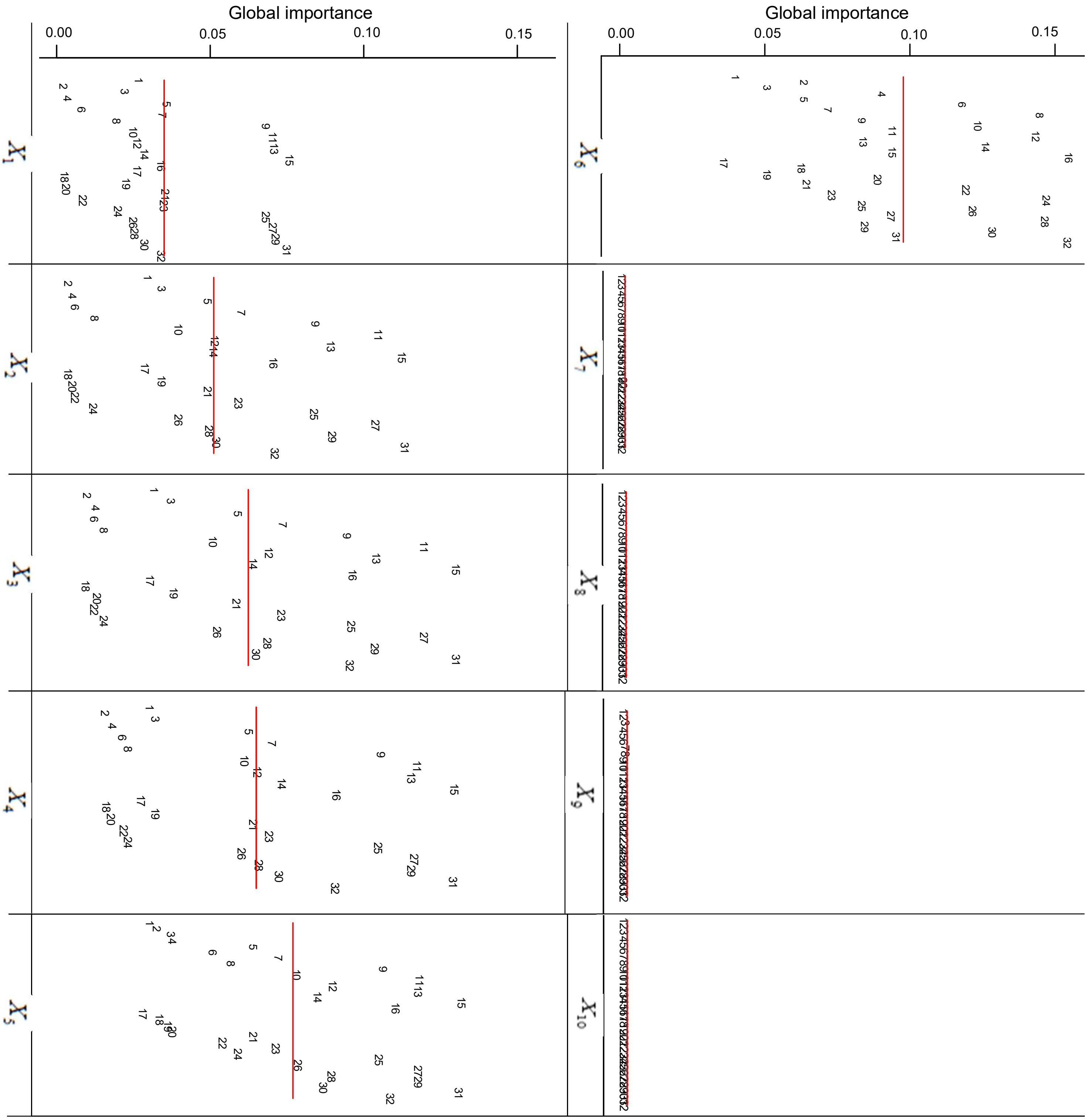


Figure 6.1 Average global variable importance for Cases $1-32$

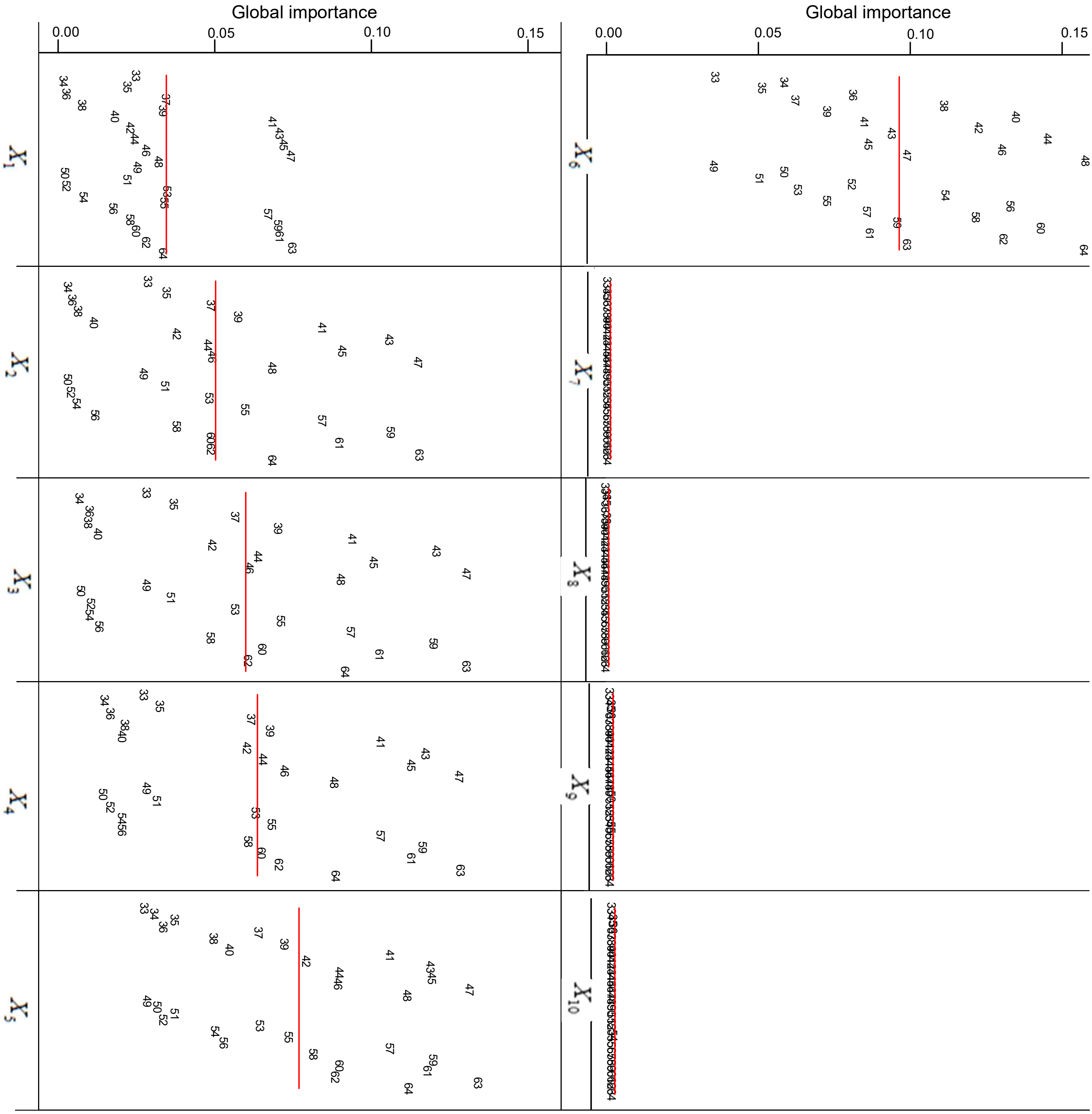


Figure 6.2 Average global variable importance for Cases $33-64$

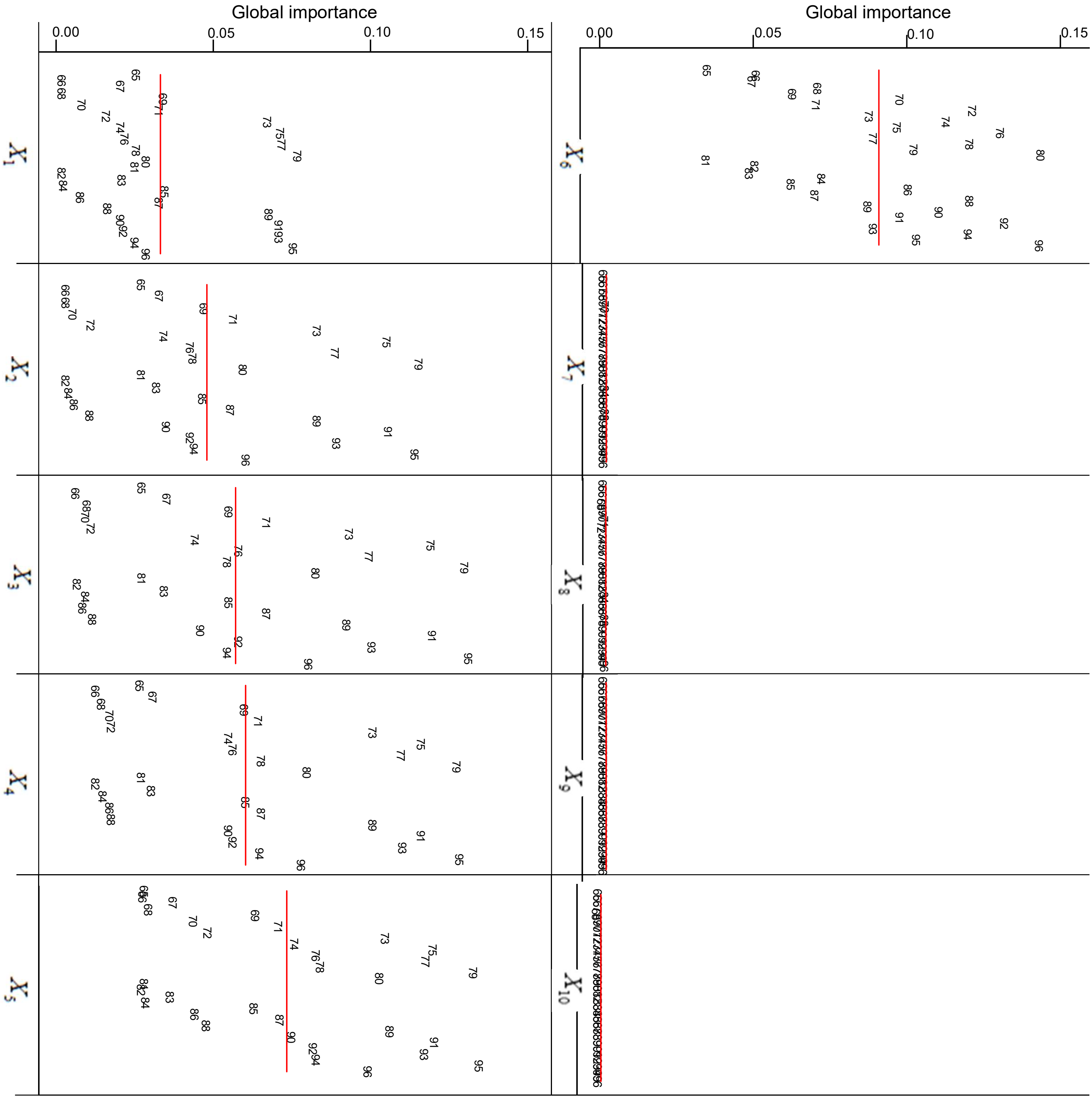


Figure 6.3 Average global variable importance for Cases $65-96$

Since the data signal, input variable correlation, label correlation, and label densities are unknown or fixed in real-world and benchmark multi-label datasets, an ideal conclusion of this study would be that the LDsplit MDA method allows for good performance in most settings. A discussion of the global LDsplit MDA variable importance results is given next.

The mean lines of Figure 6.1, Figure 6.2 and Figure 6.3 reflect the true global relevancies of the input variables, since importance increases from X_1 to X_6 and no importance is found for the irrelevant variables $X_7,...,X_{10}$. This confirms good general performance of the LDsplit MDA method. Moreover, Figure 6.1, Figure 6.2 and Figure 6.3 demonstrate that the LDsplit MDA method has no trouble in distinguishing between relevant and irrelevant input variables, as little to no importance is found for irrelevant variables $X_7,...,X_{10}$ across all 96 cases. Due to this result, the remainder of the global importance discussion below is based on the relevant variables, $X_1,...,X_6$.

Upon closer inspection it appears that the choice of M and m for LDsplit does not have a large effect on the global importance performance of LDsplit MDA. Results of Figure 6.1, Figure 6.2 and Figure 6.3 show similar behaviour for the varying values of M and m. However, a change in the number of levels of the LDsplit tree-structures, m, does give some small changes in importance results. Variable X_6 appears to be the most affected. It is unclear why X_6 shows this behaviour. Results show that the global LDsplit MDA method is particularly insensitive to the choice of the number of trees in an ensemble, M. Advantageously, these observations indicate that good global importance performance of LDsplit MDA is not dependent on a particular choice of m and M.

Since different values of m and M give very similar global results, the remarks of the following paragraphs are supported by examples using cases found in Figure 6.1, although the same conclusions can be made using the corresponding cases in Figure 6.2 and Figure 6.3.

Results show that an increase in input variable correlation, ρ_X, has a very small effect on the global importance performance of the LDsplit MDA method. Cases with smaller input variable correlation give relatively smaller global importance across variables than cases with higher input variable correlation. However, the ranking of the input variables remains the same. This can be seen by comparing Cases 2 and 6 (as well as many other examples). Case 6 leads to higher importance values for all relevant variables compared to Case 2. However, the ranking of the variables are the same for both cases, *i.e.* that variable importance increases from X_1 to X_6. This result leads to the conclusion that LDsplit MDA can capture the true global importance of variables even when input variable correlation is high.

Label correlation appears to have a larger effect on global performance of LDsplit MDA than input variable correlation has. The results suggest that it is easier for LDsplit MDA to capture the true global importance relationship between variables when label correlation is higher. Results of cases with high label correlation reflect the true global relationship between variables, since the global importance values of these cases generally increase from X_1 to X_6. This is true for datasets with $s^2 = 10$ and $s^2 = 100$. However, datasets with high label correlation as well as high signal show clearer distinction between relevant global variables. When Case 2 and Case 10 are for example compared, it is noticeable that X_1, X_2 and X_3 display very similar global importance when the signal is lower ($s^2 = 10$).

When label correlation is low, the ability of LDsplit MDA to reflect true global importance becomes more volatile. For some cases with low label correlation as well as low signal, the method struggles to reflect the true global importance distinction between relevant variables. Examples of such cases are Case 1, Case 5 and Case 7, which give similar global importance for all relevant variables $X_1, ..., X_6$. It is therefore possible to distinguish between globally relevant and globally irrelevant variables in settings such as this; however, it may be difficult to conclude which of the relevant variables are globally more relevant. As data signal increases, cases with low label correlation give more distinction between relevant variables; however, the true global relationship between variables is not reflected as X_6 is often not identified as globally most important in these settings. Examples of such cases are Case 9, Case 13 and Case 15 which give larger global importance to X_5, X_4 and X_3. It is unclear why this behaviour occurs.

The previous two paragraphs outline the influence that data signal together with changes in label correlation have on the behaviour of LDsplit MDA. However, if only the data signal is considered, the following is observed. Generally, cases with higher data signal show relatively higher importance towards all relevant variables. Therefore, higher data signal leads to the method giving better distinction between relevant and irrelevant variables, as the differences between the importance values of relevant and irrelevant variables become larger.

Varying label densities seem to cause no substantial difference in global performance of LDsplit MDA. Only slight differences in results can be identified once label densities are different. To demonstrate this, compare Case 10 to Case 12 as well as Case 14 to Case 16. For Case 10 and Case 14, global importance values increase from X_1 to X_6, however for Case 12 and Case 16 a slight fall in importance is observed from X_3 to X_4.

6.2.4 Local importance results and remarks

The local LDsplit MDA variable importance approach gives a $p \times K$ matrix, $\mathbf{F}$, as output, where entry (l,k) gives the local importance of variable X_l for label Y_k. In this synthetic study, each of the 96 cases considered have 50 $\mathbf{F}$ matrices storing the local importance results for each of the 50 synthetic datasets of a case.

The heatmaps of Appendix B.1 – B.6 are constructed to investigate the local importance performance of LDsplit MDA on the synthetic data and to illustrate how performance is influenced by the different configurations outlined in Table 6.1. Since the data are synthetic and matrix $\mathbf{A}$ is defined as given in Section 6.2.2, it is known which of the variables are in truth relevant per label and how many variables are in truth relevant per label. Therefore, the local importance heatmaps in Appendix B.1 – B.6 are found as follows. For each of the 50 $\mathbf{F}$ matrices found for a case, the six, five, four, three, two and one most important variables for Y_1, Y_2, Y_3, Y_4, Y_5 and Y_6 respectively, are marked “relevant”. The $(l,k)^{th}$ entry of a heatmap in Appendix B.1 – B.6 gives the proportion of time variable X_l is considered “relevant” for label Y_k across the 50 local importance matrices of that case. This allows the opportunity to investigate if LDsplit MDA can identify the truly locally important variables per label conditioned on the fact that the correct number of relevant variables are chosen per label. Even though the number of relevant variables per label is unknown in practice, this may be an acceptable way to represent the local results for interpretation and evaluation of LDsplit MDA in the different settings. Figure 6.5 given in Section 6.2.5 shows how successful locally relevant variables can be identified using LDsplit MDA once a simple median threshold is applied per label. More on this later.

A discussion of the LDsplit MDA local importance results of Appendix B.1 – B.6 is given next. For convenience, Table 6.2 gives a selection of heatmaps that illustrate some changes observed due to the different configurations.

Table 6.2 (a)
Comparison of selection of local importance results

Influence of

ρ_Y

Case 1

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.04	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.56	0.00
X_5	0.96	0.80	0.80	0.92	1.00	0.00
X_6	0.92	0.20	0.16	0.08	0.44	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 2

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	1.00	0.98	0.00	0.00	0.00	0.00
X_3	0.98	1.00	1.00	0.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.02	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 9

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.66	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.96	0.96	0.34	1.00	1.00	0.00
X_6	0.86	0.04	0.00	0.00	0.00	1.00
X_7	0.08	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 10

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.56	0.00	0.00	0.00
X_3	1.00	0.84	1.00	0.00	0.00	0.00
X_4	1.00	0.38	0.98	1.00	0.00	0.00
X_5	1.00	0.78	0.52	1.00	1.00	0.00
X_6	1.00	1.00	0.96	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

ρ_X

Case 3

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.02	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	0.08	0.00
X_5	0.88	0.74	0.90	0.90	1.00	0.00
X_6	0.72	0.26	0.08	0.08	0.92	1.00
X_7	0.16	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.12	0.00	0.00	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 7

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.88	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.98	1.00	1.00	0.96	0.00
X_5	0.98	0.74	0.12	0.96	1.00	0.00
X_6	0.78	0.22	0.00	0.02	0.04	1.00
X_7	0.14	0.02	0.00	0.00	0.00	0.00
X_8	0.06	0.02	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.02	0.00	0.00	0.00	0.00

m

Case 1

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.04	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.56	0.00
X_5	0.96	0.80	0.80	0.92	1.00	0.00
X_6	0.92	0.20	0.16	0.08	0.44	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 33

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.06	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.98	0.98	1.00	0.48	0.00
X_5	0.90	0.66	0.78	0.82	1.00	0.00
X_6	0.74	0.36	0.18	0.18	0.52	1.00
X_7	0.16	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 65

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.04	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.96	0.96	1.00	1.00	0.54	0.00
X_5	0.96	0.60	0.74	0.86	1.00	0.00
X_6	0.66	0.44	0.22	0.14	0.46	1.00
X_7	0.10	0.00	0.00	0.00	0.00	0.00
X_8	0.08	0.00	0.00	0.00	0.00	0.00
X_9	0.12	0.00	0.00	0.00	0.00	0.00
X_{10}	0.12	0.00	0.00	0.00	0.00	0.00

Case 11

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.98	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	1.00	0.00
X_5	0.86	0.90	0.02	0.98	1.00	0.00
X_6	0.70	0.06	0.00	0.00	0.00	1.00
X_7	0.16	0.02	0.00	0.00	0.00	0.00
X_8	0.16	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.02	0.00	0.00	0.00	0.00
X_{10}	0.12	0.00	0.00	0.00	0.00	0.00

Case 43

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.96	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.90	0.04	1.00	1.00	0.00
X_6	0.86	0.10	0.00	0.00	0.00	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 75

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.84	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.16	0.98	1.00	0.00
X_6	0.96	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Table 6.2 (b)
Comparison of selection of local importance results

Influence of

M

Case 2

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	1.00	0.98	0.00	0.00	0.00	0.00
X_3	0.98	1.00	1.00	0.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.02	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 18

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	0.98	1.00	0.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 34

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.94	0.10	0.00	0.00	0.00	0.00
X_2	0.90	0.88	0.00	0.00	0.00	0.00
X_3	0.92	0.92	1.00	0.00	0.00	0.00
X_4	1.00	0.98	0.98	1.00	0.00	0.00
X_5	0.98	0.98	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.02	0.00	0.00	0.00	0.00
X_8	0.04	0.06	0.00	0.00	0.00	0.00
X_9	0.08	0.02	0.00	0.00	0.00	0.00
X_{10}	0.06	0.04	0.02	0.00	0.00	0.00

Case 50

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	0.94	1.00	0.00	0.00	0.00	0.00
X_3	0.90	0.98	1.00	0.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.02	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Label densities

Case 25

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.80	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.96	0.94	0.22	1.00	1.00	0.00
X_6	0.82	0.06	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.08	0.00	0.00	0.00	0.00	0.00

Case 27

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.98	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.84	0.02	1.00	1.00	0.00
X_6	0.82	0.16	0.00	0.00	0.00	1.00
X_7	0.08	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 26

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.56	0.00	0.00	0.00
X_3	1.00	0.90	1.00	0.00	0.00	0.00
X_4	1.00	0.20	0.86	1.00	0.00	0.00
X_5	1.00	0.90	0.58	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 28

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.72	0.00	0.00
X_4	1.00	0.06	1.00	1.00	0.00	0.00
X_5	1.00	0.96	0.00	0.98	1.00	0.00
X_6	1.00	0.98	1.00	0.30	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

A heatmap of a case that perfectly reflects the true local importance of the synthetic data would have the form of Figure 6.4 below. In this way, the six most important variables identified for Y_1 are X_1, X_2, X_3, X_4, X_5 and X_6 for all 50 of the $\mathbf{F}$ matrices found for the case. Similarly the five most important variables for Y_2 are X_2, X_3, X_4, X_5 and X_6, for all 50 of the $\mathbf{F}$ matrices found for the case, and so on until the one most important variable for Y_6 is X_6 for all 50 of the $\mathbf{F}$ matrices found for the case.

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Figure 6.4 Heatmap displaying true local importance of synthetic data

In general, Appendix B.1 – B.6 show favourable local performance of LDsplit MDA as the heatmaps generally show a similar form to that of Figure 6.4.

The most striking observation in Appendix B.1 – B.6 is that the importance of variable X_6 for all the labels is not clearly identified once $\rho_Y = 0$. Heatmaps for which $\rho_Y > 0$ better reflect the true local importance of variables. As an example, consider Case 1 and Case 2. These two cases have low ρ_X and low signal. In Case 1, X_6 is not correctly identified as important for Y_2, Y_3, Y_4 and Y_5. Instead X_1 is incorrectly identified as important for Y_2, X_2 is incorrectly identified as important for Y_3, X_3 is incorrectly identified as important for Y_4, X_4 is incorrectly identified as important for Y_5, and X_5 is incorrectly identified as important for Y_6. However, an increase in ρ_Y from Case 1 to Case 2 causes X_6 to be correctly identified as important for Y_2, Y_3, Y_4 and Y_5, while X_1, X_2, X_3 and X_4 are respectively correctly identified as unimportant for labels Y_2, Y_3, Y_4 and Y_5. For all other values of signal and ρ_X the pattern remains the same; however, results appear slightly more unstable. As an example, compare Case 9 and Case 10. For Case 10, X_6 is correctly identified as important for Y_2, Y_3, Y_4 and Y_5 after ρ_Y increases from Case 9. However, in Case 10, X_1 remains incorrectly identified as important for Y_2. Furthermore, the importance proportion of X_4 for Y_2 drops from 1.00 to 0.38 from Case 9 to Case 10, even though X_4 is in truth locally important for Y_2. Favourably however, X_1, X_3 and X_4 are respectively correctly identified as unimportant for labels Y_3, Y_4 and Y_5 in Case 10 and the importance proportion of X_2 for Y_3 drops to 0.56.

It appears that the local importance performance of LDsplit MDA may be more affected by a change in input variable correlation compared to the results of Section 6.2.3 for global importance. The local importance results show that high input variable correlation together with low signal can cause a decrease in performance. Some variables that are correctly identified as important (unimportant) for a label are incorrectly identified as unimportant (important) once ρ_X increases. Examples can be seen when comparing Case 3 and Case 7 by specifically considering the local importance of variables X_1 and X_5 for Y_3 as well as the local importance of variables X_4 and X_6 for Y_5. Fortunately, for specifications where a strong signal is present in the realisation of the data, an increase in ρ_X leads to only small changes in performance.

The choice of m and M have a small influence on the local importance performance of LDsplit MDA. In some cases, the local importance results of LDsplit MDA are slightly better when $m = 3$. For example, for Y_1, results show lower importance proportions for the irrelevant variables (X_7, X_8, X_9 and X_{10}) in Case 1 than in Case 33 or Case 65. On the other hand, in some cases the local importance results of LDsplit MDA are slightly better when m is larger. For example, when comparing Case 11, 43 and 75, results for Y_1 show lower importance proportions for the irrelevant variables (X_7, X_8, X_9 and X_{10}) when m is larger. As no drastic increase or decrease in performance is observed due to the choice of m, it is recommended to set m small to allow for less computation time.

Only a few examples in Appendix B.1 – B.6 benefit from increasing M. Most cases show almost no change in performance after M increases (see Case 2 and Case 18 as an example). An example where an increase in M proved beneficial is Case 34 to Case 50. The results of Case 50 are more accurate and stable since the heatmap shows larger proportions for truly relevant variables per label and smaller proportions for truly irrelevant variables per label compared to Case 34. Due to the above observations and since no examples illustrate a drastic decrease in performance after an increase in M, it is recommended to set M large if computation time allows for this.

It is difficult to pinpoint exactly what influence differing label densities have on the local importance performance of LDsplit MDA. The local results are adequate for settings with equal label densities as well as those with differing label densities. However, once $\rho_Y > 0$, the differing label densities appear to influence local performance more. It seems as if local results of Y_3 and Y_4 suffer the most in these settings. As an example, compare Case 25 and Case 27 which both have $\rho_Y = 0$. The two heatmaps are very similar even though the label densities change from Case 25 to Case 27. However, when Case 26 and Case 28 are compared, which both have $\rho_Y > 0$, the change in label densities appears to influence the results. From Case 26 to Case 28 the importance proportion of X_2 for Y_3 incorrectly increases and the importance proportion of X_5 for Y_3 incorrectly and drastically decreases. The local results of

Y_4 also appear less accurate and stable in Case 28 with the importance proportion of X_3 incorrectly increasing and the importance proportion of X_6 incorrectly decreasing. It is important to note that the observed behaviour could be related to the way in which the synthetic data are generated. As mentioned in Section 6.2.1, the general problem when generating correlated Bernoulli variables which make up the label space is that the realised correlations differ from those specified once the univariate probabilities are not equal.

6.2.5 Using LDsplit MDA for variable selection on synthetic data

Figure 6.1, Figure 6.2 and Figure 6.3 given in Section 6.2.3 show that the global LDsplit MDA method is very effective in distinguishing between globally relevant and irrelevant variables on the synthetic data. This suggests that the global LDsplit MDA values could be useful for variable selection. Thresholding strategies or a probe variable strategy could be applied to the global LDsplit MDA values to distinguish globally relevant variables from irrelevant variables since the results show such a clear distinction between relevant and irrelevant variables.

In general, the local results of Appendix B.1 – B.6 show that LDsplit MDA does well in identifying local importance relationships between input variables and labels in the different settings. The figures also indicate that LDsplit MDA may be useful for identifying locally relevant variables per label. However in practice, the true number of relevant variables per label will not be known. In this setting a threshold or probe variable can be used per label to distinguish relevant variables from irrelevant variables. Figure 6.5 aims to show how successful locally relevant variables can be identified across the 50 synthetic datasets of a case when a simple median threshold strategy is applied. The heatmaps of Figure 6.5 are therefore found as follows.

For each local importance matrix of a case, the median importance value is found per label and all input variables that exceed this importance value per label are marked as “relevant” for the label. In other words, since $p = 10$ in the synthetic study, the top five most important variables are marked as relevant per label. With this method of determining relevancy, the

$(l,k)^{th}$ entry of a heatmap in Figure 6.5 gives the proportion of time variable X_l is considered relevant for label Y_k across the 50 local importance matrices of that case. Cases $17-32$ are represented in Figure 6.5 with Cases $1-16$, $33-48$, $49-64$, $65-80$ and $81-96$ available in Appendix B.7, Appendix B.8, Appendix B.9, Appendix B.10 and Appendix B.11 respectively.

In general, the results of Figure 6.5 confirm that LDsplit MDA could be useful for local variable selection. Keeping in mind that some labels have more than five or less than five locally relevant variables in truth, results show that the selected variables generally include those variables that are truly relevant per label. Consider for example the results of labels Y_4, Y_5 and Y_6 throughout in Figure 6.5. Apart from X_6 having a 0.96 selection proportion for Y_4 in Case 23, for all other cases the truly locally relevant variables have a perfect selection proportion for the respective labels. For labels Y_1, Y_2 and Y_3, similar to what was observed in Appendix B.1 – B.6, Figure 6.5 shows that the local relevancy of X_6 appears to be better detected once $\rho_Y > 0$.

Generally, the noise variables, $X_7,...,X_{10}$, are seldomly selected as locally relevant when using the simple median threshold strategy. Even for those labels which have less than five truly important variables (Y_3, Y_4, Y_5, Y_6), it appears that the five variables that are marked relevant with the median threshold strategy often consist of globally relevant variables, $X_1,...,X_6$. Once the data signal increases the truly globally relevant variables are consistently marked relevant and no noise variables are selected (as can be seen when for example comparing Case 20 with Case 28 in Figure 6.5). It would be interesting to investigate if results can be improved by using a more complicated thresholding strategy or probe variable strategy.

Case 17

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.36	0.00	0.18	0.24
X_2	1.00	1.00	1.00	1.00	0.70	0.52
X_3	1.00	1.00	1.00	1.00	1.00	0.60
X_4	0.92	0.96	1.00	1.00	1.00	1.00
X_5	0.76	0.72	0.94	1.00	1.00	1.00
X_6	0.28	0.32	0.70	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.06	0.18
X_8	0.00	0.00	0.00	0.00	0.06	0.18
X_9	0.04	0.00	0.00	0.00	0.00	0.16
X_{10}	0.00	0.00	0.00	0.00	0.00	0.12

Case 18

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.88	0.00	0.20	0.12	0.16	0.16
X_2	0.64	1.00	0.50	0.40	0.58	0.64
X_3	0.62	1.00	1.00	0.86	0.82	0.94
X_4	0.90	1.00	1.00	1.00	1.00	1.00
X_5	0.96	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.10	0.12	0.02	0.10
X_8	0.00	0.00	0.06	0.12	0.14	0.10
X_9	0.00	0.00	0.08	0.22	0.14	0.04
X_{10}	0.00	0.00	0.06	0.18	0.14	0.02

Case 19

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.26	0.00	0.14	0.10
X_2	1.00	1.00	1.00	0.98	0.68	0.48
X_3	1.00	1.00	1.00	1.00	1.00	0.90
X_4	0.94	1.00	1.00	1.00	1.00	0.98
X_5	0.62	0.76	1.00	1.00	1.00	1.00
X_6	0.38	0.22	0.74	1.00	1.00	1.00
X_7	0.02	0.02	0.00	0.00	0.00	0.12
X_8	0.02	0.00	0.00	0.02	0.08	0.08
X_9	0.02	0.00	0.00	0.00	0.02	0.22
X_{10}	0.00	0.00	0.00	0.00	0.08	0.12

Case 20

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.96	0.02	0.16	0.48	0.16	0.24
X_2	0.72	1.00	0.58	0.42	0.58	0.32
X_3	0.32	0.98	1.00	0.38	0.72	0.98
X_4	1.00	0.98	1.00	1.00	0.66	0.92
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.10	0.18	0.26	0.08
X_8	0.00	0.02	0.04	0.26	0.28	0.18
X_9	0.00	0.00	0.04	0.18	0.10	0.10
X_{10}	0.00	0.00	0.08	0.10	0.24	0.18

Case 21

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.20	0.24
X_2	1.00	1.00	1.00	1.00	0.80	0.60
X_3	1.00	1.00	1.00	1.00	1.00	0.96
X_4	0.94	1.00	1.00	1.00	1.00	1.00
X_5	0.64	0.90	0.88	1.00	1.00	1.00
X_6	0.38	0.10	0.12	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.08
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.12

Case 22

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.26	1.00	0.30	0.48	0.82	0.92
X_2	0.74	0.34	0.70	0.60	0.36	0.22
X_3	0.98	0.68	1.00	0.94	0.74	0.82
X_4	1.00	0.98	1.00	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.02
X_8	0.00	0.00	0.00	0.00	0.04	0.00
X_9	0.02	0.00	0.00	0.00	0.02	0.02
X_{10}	0.00	0.00	0.00	0.00	0.02	0.00

Case 23

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.04	0.14	0.28
X_2	1.00	1.00	1.00	1.00	0.86	0.62
X_3	0.98	1.00	1.00	1.00	1.00	1.00
X_4	0.84	0.98	1.00	1.00	1.00	1.00
X_5	0.74	0.72	0.94	1.00	1.00	1.00
X_6	0.40	0.28	0.06	0.96	1.00	1.00
X_7	0.02	0.02	0.00	0.00	0.00	0.02
X_8	0.00	0.00	0.00	0.00	0.00	0.06
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.02

Case 24

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.30	1.00	0.86	0.20	1.00	1.00
X_2	0.58	0.70	0.98	0.76	0.34	1.00
X_3	0.84	0.62	0.92	1.00	0.70	0.32
X_4	1.00	0.68	0.48	1.00	0.86	0.50
X_5	1.00	1.00	0.76	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.06
X_8	0.10	0.00	0.00	0.00	0.06	0.04
X_9	0.06	0.00	0.00	0.02	0.02	0.04
X_{10}	0.08	0.00	0.00	0.02	0.02	0.04

Case 25

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.10
X_2	1.00	1.00	1.00	1.00	1.00	0.90
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.86	0.94	1.00	1.00	1.00	1.00
X_6	0.14	0.06	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 26

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	0.92	1.00	1.00	1.00	1.00	1.00
X_3	0.20	0.90	1.00	1.00	1.00	1.00
X_4	0.88	0.20	1.00	1.00	1.00	1.00
X_5	1.00	0.90	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 27

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.08
X_2	1.00	1.00	1.00	1.00	1.00	0.92
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.78	0.84	1.00	1.00	1.00	1.00
X_6	0.18	0.16	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 28

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.12	0.00
X_2	1.00	1.00	1.00	1.00	0.88	1.00
X_3	0.08	1.00	1.00	1.00	1.00	1.00
X_4	0.98	0.06	1.00	1.00	1.00	1.00
X_5	0.94	0.96	1.00	1.00	1.00	1.00
X_6	1.00	0.98	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 29

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.12
X_2	1.00	1.00	1.00	1.00	1.00	0.88
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.86	1.00	1.00	1.00	1.00	1.00
X_6	0.12	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 30

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	0.88	1.00	1.00	1.00	1.00	1.00
X_3	0.26	0.96	1.00	1.00	1.00	1.00
X_4	0.86	0.10	1.00	1.00	1.00	1.00
X_5	1.00	0.94	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 31

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.06
X_2	1.00	1.00	1.00	1.00	1.00	0.94
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.70	0.90	1.00	1.00	1.00	1.00
X_6	0.22	0.10	0.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 32

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.14	0.00	0.04	0.00
X_2	0.98	1.00	1.00	1.00	0.96	1.00
X_3	0.04	1.00	1.00	1.00	1.00	1.00
X_4	1.00	0.06	1.00	1.00	1.00	1.00
X_5	0.98	0.94	0.86	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Figure 6.5 Local heatmaps for Case $17-32$ using a median threshold strategy

6.2.6 Conclusions of the LDsplit MDA synthetic study

This section highlights the most important conclusions of the LDsplit MDA synthetic study.

In general, both the global and local importance measures of LDsplit MDA display very favourable performance on the synthetic data. For all configurations of input variable correlation, label correlation, data signal and label densities, LDsplit MDA adequately displays the true global and local importance of input variables. This is especially the case with higher label correlation.

The synthetic results show that both the global and local importance measures of LDsplit MDA fare better once $\rho_Y > 0$. When $\rho_Y = 0$, the synthetic results are unstable and the true global and local importance of input variables are not clearly identified. The results of the globally most important variable, X_6, appear to suffer the most. Since LDsplit was developed with the aim of implicitly exploiting label correlation, this could contribute to the reason why the LDsplit MDA method better reflects true global and local variable importance once $\rho_Y > 0$.

Advantageously, the synthetic results show that high input variable correlation has a very small effect on the global importance performance of the LDsplit MDA method in all settings. The same cannot be said for the local importance results. In settings with low data signal, local importance performance may be sensitive to high input variable correlation. Some variables that are correctly identified as important (unimportant) for a label are incorrectly identified as unimportant (important) once ρ_X increases. Fortunately, the synthetic results show minor differences in local performance if ρ_X increases while data signal is stronger. These observations motivate why the development of an LDsplit conditional permutation importance method, such as that mentioned in Section 5.5.2, is not a large priority at this stage.

LDsplit MDA gives adequate results for settings with equal label densities as well as settings with unequal label densities. Global results are not considerably influenced by a change in label densities, while local results may be more sensitive to such changes. Because results are based on correlated Bernoulli variables which prove difficult to generate once the univariate probabilities are not equal, it would be beneficial to investigate the performance of LDsplit MDA on imbalanced multi-label data more extensively in future studies.

In general, LDsplit MDA performs adequately for all values of m and M used in the study. Therefore, LDsplit MDA can be applied to any fitted LDsplit model in order to better interpret the LDsplit model used for classification. On the other hand, if LDsplit MDA is specifically applied as a measure for better understanding the multi-label data or as a filter approach for variable selection, guidelines can be given regarding the specification of m and M to ensure optimal performance.

Results show that good global performance of LDsplit MDA is not reliant on the choice of m and M for LDsplit. Little to no difference in global performance is observed when the number of trees in the ensemble, M, is increased from 10 to 100. Although this indicates no global benefit for setting M large, it does demonstrate that overfitting is not a concern for large M. For the number of tree-levels, m, the global importance of X_6 is slightly unstable for large m; however, in general the choice of m does not appear to be a large concern for adequate global importance performance. Local performance of LDsplit MDA is slightly more sensitive to the choice of m and M. The general parameter recommendation of LDsplit MDA is therefore based on the observed local importance results.

In Section 6.2.4 some examples demonstrate improvements in local importance results once M increases. As is the case for global importance performance, no evidence of overfitting is found due to large M. Therefore, it is recommended to set M large. For the number of tree-levels, m, examples can be given for an increase in performance as well as a decrease in performance due to larger m. However, since no clear benefit is observed for large m and since some results indicate slightly unstable behaviour once m is large, it is recommended to set m small to allow for less computation time as well as favourable global and local performance of LDsplit MDA.

Advantageously, the above tuning parameter recommendations for m and M correspond with those given in Chapter 3 and Chapter 4. In other words, an LDsplit model that generally gives good classification performance could simultaneously describe the importance of the input variables well.

Furthermore, the clear distinction shown between relevant and irrelevant variables in the synthetic study confirms the potential of LDsplit MDA as a filter approach for variable selection. The favourable results obtained with the simple median thresholding strategy in Section 6.2.5 encourages further investigations that make use of more sophisticated thresholding strategies or probe variable strategies for variable selection. It should be noted, however, that in the synthetic study the irrelevant variables are generated from a standard normal distribution independently of the label variables. Therefore in practice, such a drastic distinction in importance may not necessarily exist, for example if all the input variables are more or less equally relevant.

6.3 Applying LDsplit MDA to the *Emotions* benchmark dataset

In this section the LDsplit MDA method is applied to an LDsplit ensemble fit on the standard training dataset of the benchmark dataset *Emotions*. The method is used to determine globally and locally important input variables for the *Emotions* data. This section therefore illustrates how LDsplit MDA allows an LDsplit model to be more interpretable in terms of variable importance.

To conduct this study, a Random LDsplit model with $m = 4$ and $M = 100$ is fit to the *Emotions* training dataset. The rule of thumb (Section 4.2.1) is applied to set the LDsplit tuning parameters m and M. The fit of each LDsplit tree-structure is based on a random subsample of two thirds of the training data, allowing one third of the observations to form the OOB set of the tree, *i.e.* $prop_{OOB} = \frac{1}{3}$. The minimum node-size is specified as $n = 5$ and a threshold of

0.5 is used for all labels. A radial basis function kernel SVM with $\gamma = \frac{1}{72}$ and $C = 1$ is used as binary base classifier within LDsplit. Hereafter, LDsplit MDA is applied to the fitted LDsplit ensemble. For the global importance calculations, the performance of both Hamming loss and F-score are considered. The global importance values obtained using the respective measures are appropriately transformed to a $0-100$ scale. Figure 6.6 gives the global importance ranking of the top 30 variables found by applying LDsplit MDA with Hamming loss and F-score as the respective evaluation measures. Similarly, for each label the local importance values are appropriately transformed to a $0-100$ scale. Figure 6.7 gives the top 30 locally important variables per label as determined by using the LDsplit MDA strategy.

As summarised in Table 2.2 of Section 2.3 the *Emotions* dataset consists of $N = 391$ training observations, each corresponding to a piece of music. The music pieces are described using $p = 72$ input variables as outlined in Trohidis *et al.* (2008). A complete overview of audio variable extraction is beyond the scope of this dissertation. Therefore, only a brief description of the audio input variables is given next.

Variables $X_{65} - X_{72}$ of the *Emotions* data are so-called rhythmic variables (Trohidis *et al.*, 2008). These rhythmic variables are formed by extracting periodic changes from a beat histogram (Tzanetakis and Cook, 2002). First an algorithm that identifies peaks using autocorrelation is implemented to select the two highest peaks. The amplitudes and Beats Per Minute (BPM) of the two respective peaks are computed and define variables X_{65}, X_{66}, X_{67} and X_{68}. The high-to-low ratio of these BPMs defines X_{69}. Furthermore, variables X_{70}, X_{71} and X_{72} are respectively calculated by summing the histogram bins between $40-90$, $90-140$ and $140-250$ BPMs. The remaining 64 variables, $X_1 - X_{64}$, are so-called timbre variables (Trohidis *et al.*, 2008). The Short-Term Fourier Transform (STFT) (Rabiner and Juang, 1993) is used to determine the spectral centroid, spectral roll-off and the spectral flux (Tzanetakis and Cook, 2002). Additionally, Mel Frequency Cepstral Coefficients (MFCCs)

(Hasan *et al*., 2004) are derived of which the first 13 are selected. Note that this gives rise to 16 audio representations. $X_1 - X_{64}$ are formed by calculating the mean, standard deviation, mean standard deviation and standard deviation of standard deviation over all frames of the 16 representations.

Since the *Emotions* data are not simulated, it is difficult to know which input variables are truly globally and locally important. However, it may be possible to assess if the results appear reasonable based on intuition and background knowledge of the six emotion labels: *Amazed-Surprised*, *Happy-Pleased*, *Relaxing-Calm*, *Quiet-Still*, *Sad-Lonely* and *Angry-Aggressive*. For example, one could argue that the emotions *Relaxing-Calm* and *Quiet-Still* have similarities, so that it could be expected that these labels have some locally important input variables in common. On the other hand, two intuitively contrasting emotions such as *Amazed-Surprised* and *Relaxing-Calm* may have sets of locally important variables that are very different. It is certainly possible that the same variables are important for intuitively contrasting emotions as well. The variable may for example aid in recognising distinguishing properties between the emotions.

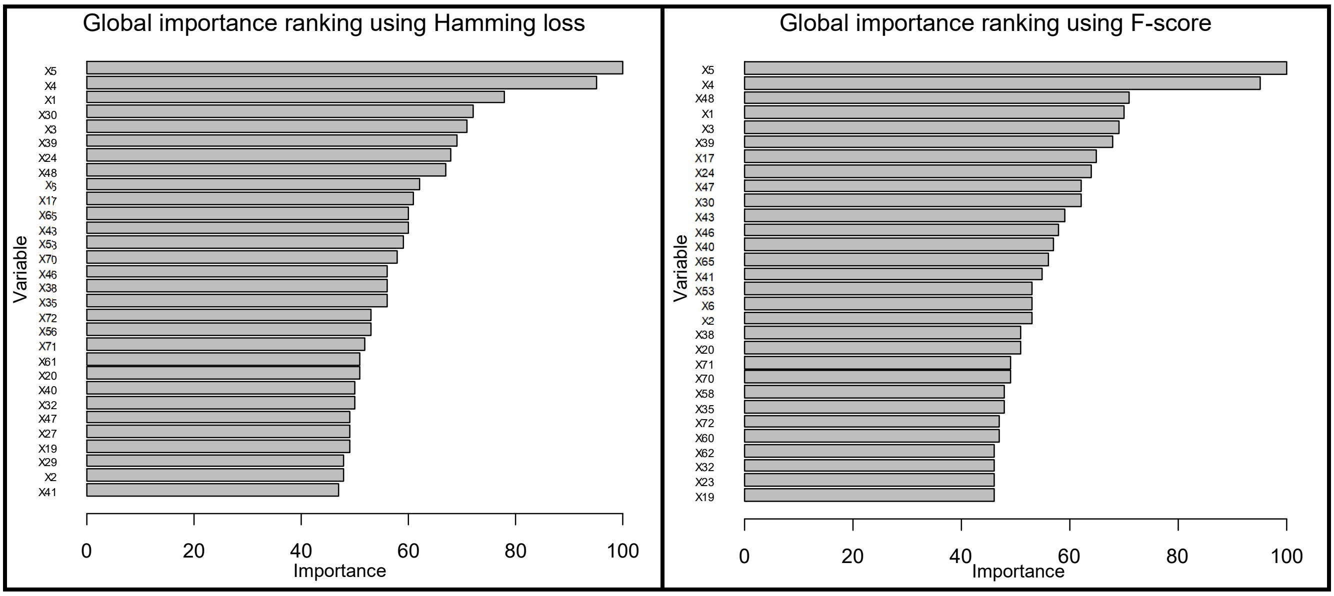


Figure 6.6 Top 30 globally important variables for *Emotions* when using LDsplit MDA

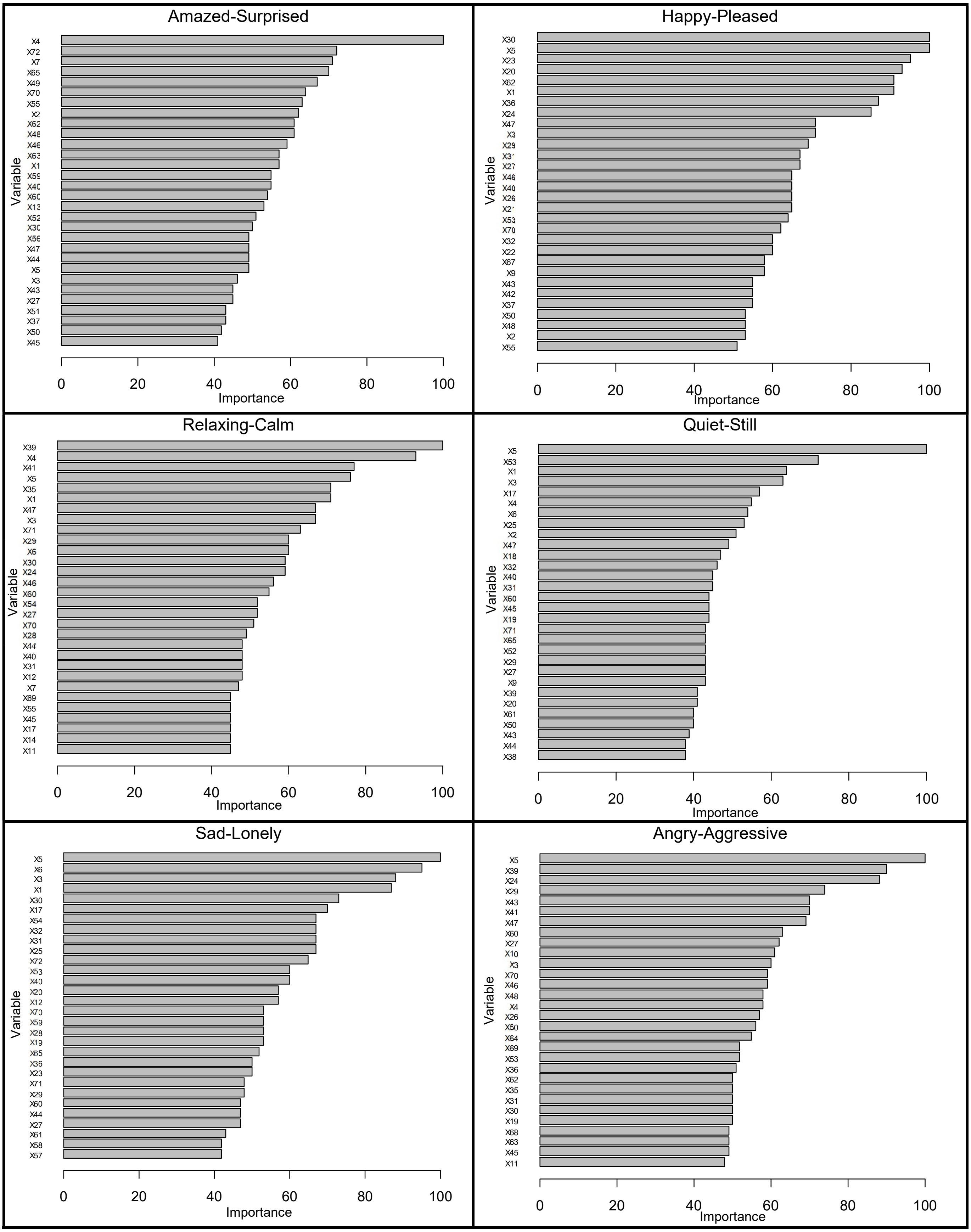


Figure 6.7 Top 30 locally important variables per label for the *Emotions* data based on LDsplit MDA

Figure 6.6 shows that regardless of whether Hamming loss or the F-score is used to calculate global importance with LDsplit MDA, the collections of globally most important variables are very similar. For both cases in Figure 6.6, variables X_5 and X_4 are found to have the highest and second highest global importance. Furthermore, Figure 6.6 shows that in both cases, X_5 and X_4 display noticeably higher global importance than the other variables of the dataset. Upon closer inspection of the two sets of top 30 variables, 26 variables are identified that appear in both sets. Furthermore, when only the top ten globally most important variables of the two respective cases are considered, nine variables appear in both sets. These variables are: $X_5, X_4, X_1, X_{30}, X_3, X_{39}, X_{24}, X_{48}$ and X_{17}.

Figure 6.7 shows that for most labels there are a handful of variables that are noticeably more important than the others. *Amazed-Surprised* and *Quiet-Still* each have one such variable. For *Angry-Aggressive* the top three variables show noticeable importance, whereas for *Sad-Lonely* the top four variables are noticeably more important. The *Happy-Pleased* label shows slightly different behaviour compared to the other labels since a total of eight variables display noticeably higher importance.

Figure 6.7 also shows that X_5 is regarded as a highly important variable for all the labels except for *Amazed-Surprised* where it is ranked in the 23^{rd} position. Recall that X_5 is identified as the globally most important variable in Figure 6.6. Since X_5 is regarded as locally important for many of the labels, it could justify why this variable is globally most important. Interestingly, X_4 is identified as the second most globally important variable in Figure 6.6; however, the variable is not found to be locally important for all the labels. The local results of Figure 6.7 show high local importance of X_4 for *Amazed-Surprised*, *Relaxing-Calm* and *Quiet-Still*, but X_4 is ranked 15^{th} for the *Angry-Aggressive* label and is not found within the top 30 locally important variables for *Happy-Pleased* and *Sad-Lonely*. Other examples of globally important variables that only show local importance for a selection of labels are X_{39}, X_{24}, X_1 and X_{30}. For example, X_{39} is identified as the locally most important variable for *Relaxing-Calm* and the second most locally important variable for *Angry-Aggressive*, while it is simultaneously ranked 24^{th} for *Quiet-Still* and not found within the top 30 locally important variables for *Amazed-Surprised*, *Happy-Pleased* and *Sad-Lonely*. This illustrates the advantage of defining both a global and local importance measure, since information such as this may have been lost if only the global importance of variables were considered.

Figure 6.8 is constructed to support Figure 6.7 and to allow for quick comparisons between locally important variables per label. In Figure 6.8 the top ten locally most important variables for each label are used to construct several Venn diagrams with the **total number** of variables per section given in the diagrams. Based on these figures some further remarks are given next.

Firstly, it is noteworthy that the collection of top ten locally important variables identified for *Amazed-Surprised* differs very much from that of the other five labels in the dataset. When comparing the respective collections of *Amazed-Surprised* and *Quiet-Still*, only X_4 and X_2 are found within both collections. Furthermore, the collections of *Amazed-Surprised* and *Relaxing-Calm* have only X_4 in common, and the collections of *Amazed-Surprised* and *Happy-Pleased* have only X_{62} in common. Moreover, none of the top ten locally most important variables of *Amazed-Surprised* correspond to that of *Sad-Lonely* or *Angry-Aggressive*. Intuitively, this could be a reasonable result if music pieces that evoke the *Amazed-Surprised* emotion are regarded as fairly different from music pieces that evoke the other emotions.

On the other hand, the collections of top ten locally most important variables identified for *Relaxing-Calm*, *Quiet-Still*, *Sad-Lonely* and *Happy-Pleased* show more overlap. For example, the collections of top ten locally important variables identified for *Quiet-Still* and *Sad-Lonely* have six variables in common and *Relaxing-Calm* and *Quiet-Still* have five locally important variables in common within their collections. Furthermore, *Happy-Pleased* has four locally important variables in common with *Relaxing-Calm* and *Quiet-Still* within their top ten locally most important variables. The last Venn diagram given in Figure 6.8 (c) is constructed to better investigate the relationship between the four emotion labels, *Relaxing-Calm*, *Quiet-Still*, *Sad-Lonely* and *Happy-Pleased*. The diagram shows that all four labels have a number of unique variables within their top ten locally most important variables as well as three variables that appear within the top ten of all four labels, namely X_1, X_3 and X_5. As seen in Figure 6.6, these three variables are ranked among the top five globally important variables. It is also noteworthy that X_1 and X_3 do not appear to be very important for the other two labels *Amazed-Surprised* and *Angry-Aggressive* (as seen in Figure 6.7).

Finally, it is interesting that, within their top ten locally most important variables, *Angry-Aggressive* has five locally most important variables in common with *Relaxing-Calm*. These variables are X_5, X_{39}, X_{41}, X_{29} and X_{47}. Note that X_5 is the globally most important variable as given by Figure 6.6 while X_{39} is regarded as the sixth globally most important variable. Furthermore, X_{39} is found locally most important for *Relaxing-Calm* and the second most important variable for *Angry-Aggressive* (Figure 6.7). However, X_{39} is arguably only important for *Relaxing-Calm* and *Angry-Aggressive* as it does not appear within the top 20 locally important variables for any of the other labels. The same is true for X_{41}. One would intuitively expect a music piece that evokes an *Angry-Aggressive* emotion to sound different from a music piece which evokes a *Relaxing-Calm* emotion. Therefore, it may be insightful to investigate in future work if X_{39} and X_{41} for example provide an important distinction between the *Angry-Aggressive* and *Relaxing-Calm* labels.

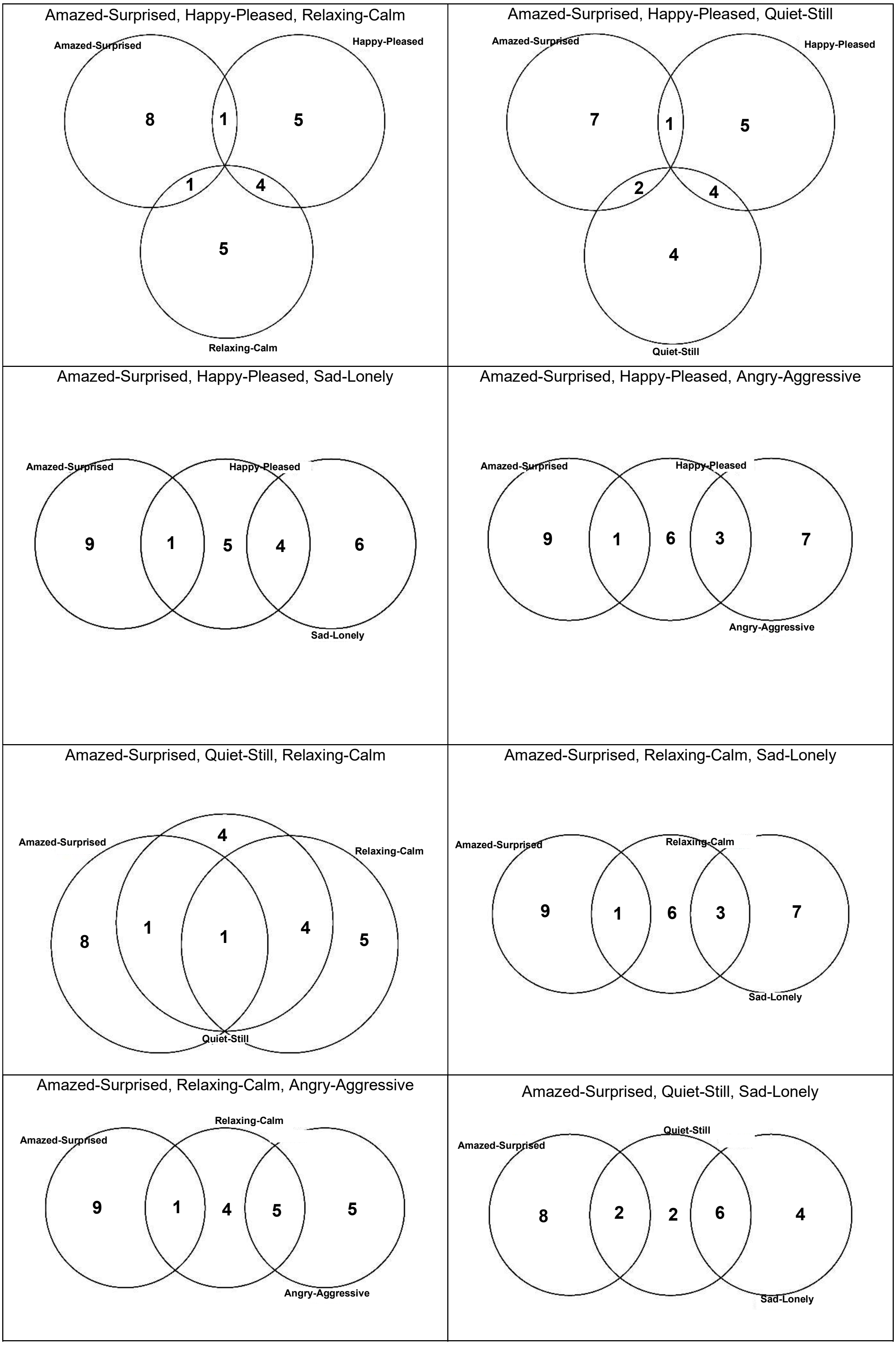


Figure 6.8 (a) Venn diagrams for top ten locally important variables of different labels

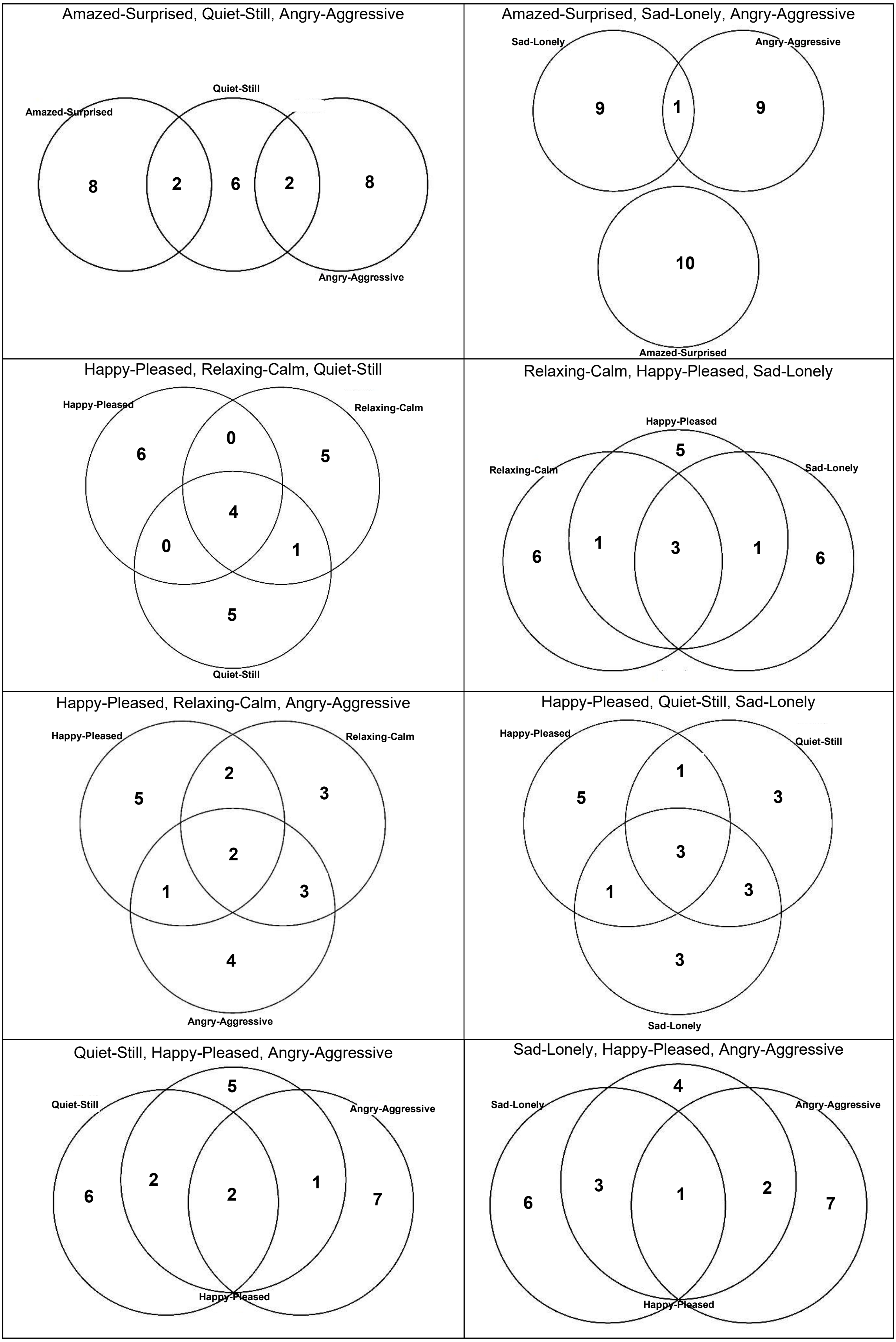


Figure 6.8 (b) Venn diagrams for top ten locally important variables of different labels

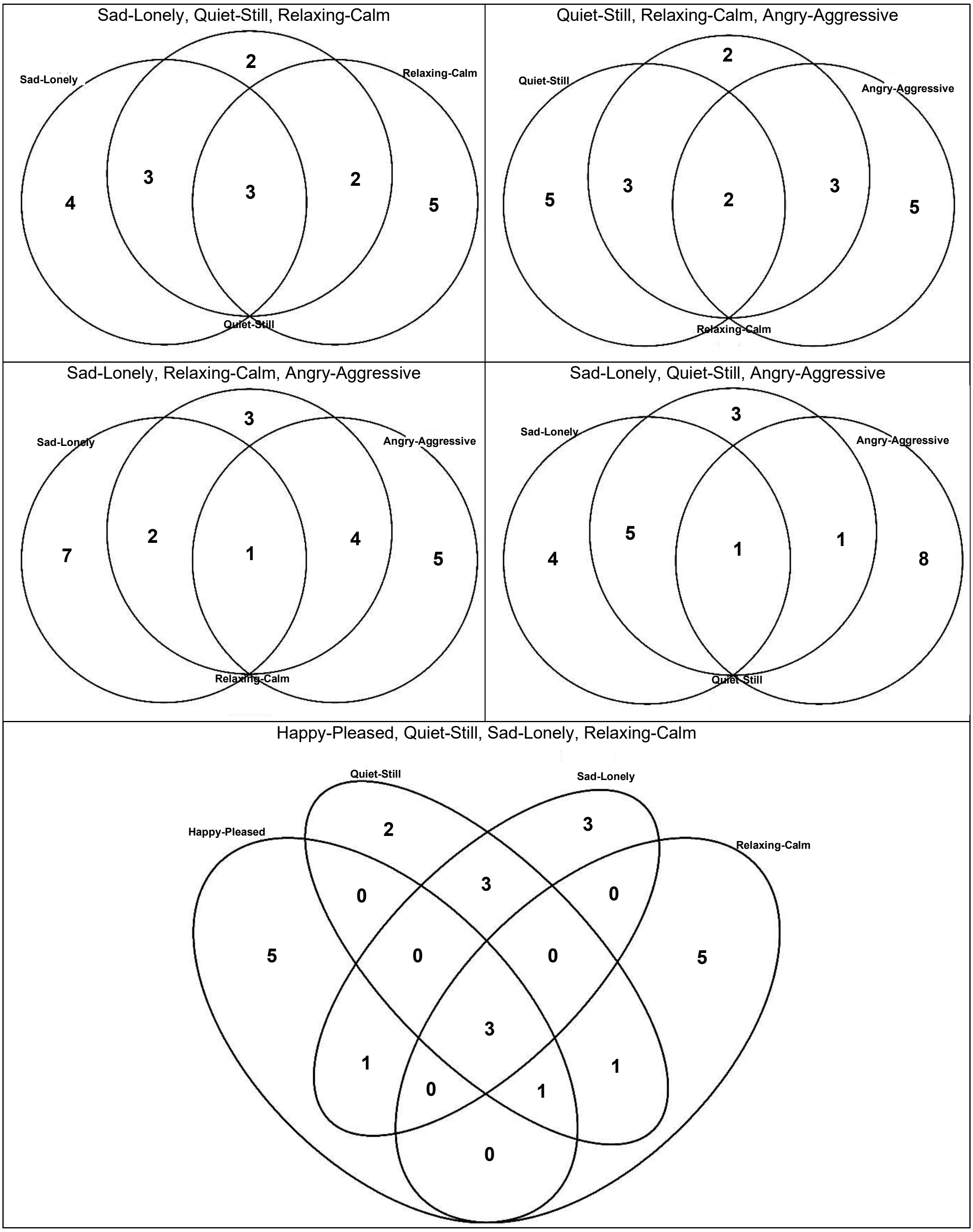


Figure 6.8 (c) Venn diagrams for top ten locally important variables of different labels

6.4 Conclusion

LDsplit MDA is a novel input variable ranking approach for multi-label data that produces both global and local input variable rankings and that exploits label correlations within the data. In this chapter, empirical properties and different applications of the newly proposed method were considered.

Firstly, the performance of LDsplit MDA was evaluated on synthetically generated multi-label data. The extensive study consisted of 96 configurations. The influence that label densities, label correlation, input variable correlation, data signal and LDsplit tuning parameters (m and M) have on performance were investigated. For each of the 96 cases, 50 multi-label synthetic datasets were generated to which a Random LDsplit model (as specified) was fit. Hereafter the LDsplit MDA approach was applied to each of the fitted LDsplit ensembles, and the results were appropriately combined per case.

Results demonstrated that both the global and local importance measures of LDsplit MDA display very favourable performance in all settings; however, cases with higher label correlations produce the best results. This could be regarded as a reasonable result since LDsplit was precisely developed to implicitly exploit label correlations. The absence of label correlations therefore weakens the performance of LDsplit MDA.

Since LDsplit MDA shows adequate performance for all values of m and M used in the study, the conclusion is made that LDsplit MDA can be applied to any fitted LDsplit model for better interpretability of the model, as good performance of LDsplit MDA is not dependent on the choice of m and M. In addition, general guidelines for the values of m and M were also provided for settings in which LDsplit MDA is merely used as a filter approach for variable selection. In this case it is recommended to set m small while M is large.

The chapter concluded with an illustration of how LDsplit MDA can be applied to a fitted LDsplit ensemble for better interpretability of non-synthetic data. For this purpose, the well-known *Emotions* dataset was used. In general, this section highlighted the advantage of obtaining both global and local input variable rankings with LDsplit MDA. By simultaneously considering both the global ranking and the local rankings of each variable, interesting patterns were identified from the data. For example, the study demonstrated that some of the globally important variables only show local importance for a selection of labels. Furthermore, the local importance rankings of some labels were very different or very similar to that of the other labels. This allowed for additional insight into the relationship between the different emotion labels.

The subsequent chapter concludes this dissertation with a summary of the main findings and contributions of the research. Opportunities for future research are also provided.

Chapter 7: Conclusions and opportunities for future research

7.1 Summary

In this dissertation the supervised learning task of multi-label classification was considered. Chapter 1 provided a brief introduction to the field. It was found that a dramatic increase in the number of available multi-label datasets in recent years and the immense diversity of the different problems have led to an increase in multi-label research. However, since multi-label data possess unique characteristics in comparison to single-label data, multi-label learning poses additional challenges. Therefore, despite the observed increase in multi-label research, some shortcomings within the literature could be identified. Emphasis was placed on two shortcomings in particular.

Firstly, it was found that research widely considers the incorporation of label correlation into a multi-label learning method as a fundamental element for good classification performance. However, as illustrated, label correlations are not necessarily shared globally by all data cases. Despite this, it was observed that limited research is based on local label correlation exploitation for accurate multi-label classification. Secondly, some multi-label classification applications require interpretable results. Therefore, input variable ranking for multi-label data is an important area of research as it provides insight regarding the importance of the respective input variables. Such input variable rankings can also be used to perform variable selection. However, it was observed that few variable importance measures for multi-label data consider a distinction between global and local relevancy. Moreover, the few available measures mostly neglect to exploit label dependencies.

To allow for a better understanding of the current resources available, and to provide background knowledge so that the first shortcoming could be addressed, Chapter 2 gave an in-depth overview of multi-label learning. This included the definitions of multi-label evaluation measures, summaries of popular multi-label benchmark datasets and references to popular software for multi-label learning. Furthermore, different viewpoints of label correlation in the multi-label literature were outlined, namely the explicit and implicit exploitation viewpoint, the global and local label correlation viewpoint, as well as differences between first-, second- and higher-order methods. Hereafter, extensive research was undertaken on how existing multi-label learning methods exploit label correlations according to the different viewpoints. Within each of the three categories of multi-label learning (problem transformation methods, algorithm adaptation methods and ensemble methods) different multi-label learning methods were discussed in detail.

In Chapter 3 a new tree-based ensemble method for multi-label classification was proposed, namely LDsplit. LDsplit was developed with the aim to implicitly exploit local higher-order label correlations within multi-label data to achieve good multi-label classification. Different to other previously proposed multi-label tree-based ensembles, LDsplit implicitly exploits higher-order label correlations in a simple manner by making use of label hierarchies. Observations that make up a node at a level of an LDsplit tree-structure do not only share information regarding the label used for that level, they also share information regarding the labels of parent levels. In this way higher-order label correlations were implicitly incorporated in the proposed model. Moreover, the proposed node-splitting procedure allows for the exploitation of local label correlations since the data are divided into subgroups that share label information. It was described how Random LDsplit and Conditional LDsplit can be fit depending on whether random or predetermined label hierarchies are used. Furthermore, the chapter included a detailed discussion of the distinctive and favourable properties of LDsplit compared to other well-known and related multi-label learning methods. Scaling techniques for larger values of K were also given. All the required R-functions for LDsplit are given in Appendix C.

Chapter 4 presented an extensive empirical evaluation of LDsplit on six diverse benchmark datasets. In the first stage of the study, where the interactions between the tuning parameters were investigated, a general rule of thumb was provided for setting the values of m and M. It was specified that m should be set small ($3 \leq m \leq 4$) while M is large with respect to K. However, since results are dependent on the multi-label dataset at hand, an adequate way to determine m and M in practice is by means of cross-validation, if computation time allows for this.

When comparing Random and Conditional LDsplit in the second stage of the empirical study of Chapter 4, it was found that a small Conditional LDsplit ensemble can achieve similar or better performance than a large Random LDsplit ensemble.

Furthermore, results of Chapter 4 indicated a considerable dependence on the chosen binary classifier for LDsplit. In fact, a much smaller difference in performance is observed between the random and predetermined label ordering strategies in comparison. Since this is the case, care should be taken to ensure that an appropriate base classifier is selected for LDsplit. This can be done by using cross-validation as illustrated in Chapter 4. However, this is a computationally expensive strategy. Therefore, it may be beneficial to conduct an extensive study with regards to binary classifiers within LDsplit in future work, so that a general recommendation can be made.

Finally, in the last stage of the empirical study of Chapter 4, the predictive performances of Random and Conditional LDsplit models with both an SVM or decision tree as base classifier were compared to 17 other well-known and related multi-label learning methods considering 12 evaluation measures across the six benchmark datasets. The corrected Friedman test and the post-hoc Nemenyi test were used to assess the overall differences in performance of LDsplit models compared to the other previously proposed models. In general, the LDsplit models produced very satisfactory predictive performance. Results showed that in terms of mean rank, Conditional LDsplit and Random LDsplit models with an SVM base classifier were the first- and second-best performing models for six of the evaluation measures and were found within the top three models for ten of the evaluation measures. Furthermore, these LDsplit models achieved statistically significant improvements in performance compared to the tree-based methods ML-C4.5, MODT and RFML-C4.5 as well as the algorithm adaptation method, ML-kNN, for many evaluation measures. Based on these results, these two LDsplit models were found to be the top two performing models on average in the empirical study. This emphasises the contribution of this dissertation in terms of local label correlation exploitation for better classification performance.

In Chapter 5 attention shifted to the second identified shortcoming of the multi-label literature on which this dissertation was based. Firstly, traditional variable importance measures for single-label trees, as well as previous proposals to measure variable importance and perform variable selection for multi-label data, were discussed. The discussion highlighted that many of the current methods for input variable ranking within multi-label data are based on the problem transformation paradigm and therefore mostly neglect exploitation of label dependencies. Moreover, few proposals consider both global and local importance of variables. In fact, no local importance measures could be found in the literature that incorporate label correlations. Consequently, to address these shortcomings, different ways were outlined to show how an LDsplit ensemble can produce global and local input variable rankings. Advantageously, the proposed global and local input variable importance measures are influenced by label correlations since they are derived from a fitted LDsplit ensemble which implicitly exploits local higher-order label correlations. LDsplit MDA can provide a global and local variable importance ranking for any fitted LDsplit model used for classification. LDsplit MDA was therefore established as the preferred strategy of this work as it can be used for better interpretability of LDsplit models in general. Furthermore, the derived input variable rankings can also be used to apply a filter approach for variable selection.

Chapter 6 considered empirical properties and applications of the proposed LDsplit MDA method. Firstly, an extensive synthetic study that consisted of 96 configurations was performed to evaluate whether the proposed LDsplit MDA measures can detect truly globally and locally relevant variables. When considering the influence that label densities, label correlation, input variable correlation, data signal and the LDsplit tuning parameters, m and M, have on performance, it was found that LDsplit MDA is able to best detect truly globally and locally important variables when label correlations in the data are larger. However, this may have been anticipated since LDsplit was developed with the aim to implicitly exploit label correlations. It was furthermore concluded that LDsplit MDA can allow for better interpretation of any fitted LDsplit model, since it adequately detected truly globally and locally important input variables for most cases in the synthetic study and did not show large dependence on the values of m and M. If the values of m and M are to be specified for LDsplit MDA, the same general rule of thumb as provided for LDsplit classification was found to hold. Lastly, the clear distinction shown between relevant and irrelevant variables in the synthetic study confirms the potential of LDsplit MDA as a filter approach for variable selection.

The second application of LDsplit MDA in Chapter 6 consisted of a short benchmark dataset application on the well-known *Emotions* dataset. It was illustrated how the proposed global and local LDsplit MDA measures can be used to detect interesting relationships between input variables and labels. Some of the observed findings may have been lost if only previously proposed global importance measures were used to interpret the data. For example, the study demonstrated that some of the globally important variables only show local importance for a selection of labels, and that some local importance rankings are very similar to or very different from that of other labels. This highlights the contribution of this dissertation in terms of providing both global and local measures for variable importance that are dependent on label correlations.

7.2 Opportunities for future research

This section provides several possible avenues for future research.

In the first stage of the empirical study presented in Chapter 4, the standard training datasets of six multi-label benchmark datasets were used to examine the properties of the LDsplit tuning parameters, m and M. In a similar manner the influence of the minimum node-size, n, can be investigated in future work, since a fixed value of $n = 5$ was used throughout this dissertation. It would also be interesting to investigate the relationship between n and the binary classifier used for LDsplit. For example, in practice, some nodes may operate close to the minimum node-size. In this case, a binary classifier that does not require a large set of observations to produce good performance, may be a better choice.

Two binary base classifiers within LDsplit were considered in the empirical study of Chapter 4, namely a decision tree and an SVM. In general, LDsplit models with an SVM base classifier gave better classification performance. In fact, the Conditional LDsplit and Random LDsplit models with an SVM base classifier were identified as the top two performing models on average in the empirical study. It would be interesting to investigate whether classification performance of these two models can be improved by performing cross-validation to determine the values of the SVM tuning parameters, γ and C. Note that, since only the default hyper-parameters of the SVM algorithm are used, LDsplit performs at least as well as reflected in the results in the dissertation. Such possibilities confirm the opportunities for further research into and improvements of LDsplit, but pursuing all of these was deemed to be too much for a single dissertation.

It may also be interesting to investigate how performance is affected when threshold strategies, such as those given in Al-Otaibi *et al.* (2014a), are used to determine the values of the threshold applied per label, $t_{Y_1}, t_{Y_2}, \ldots, t_{Y_K}$. One can for example consider selecting random threshold values in the range $[0.5, 1]$ for each label per tree-structure. This would result in higher diversity of ensemble members and possibly better generalisation ability. In general, it should be noted that by increasing a threshold, stronger evidence is required to declare a label present. This can naturally lead to a label being wrongly declared absent. Therefore, depending on the specific application, adjustments may be made to the thresholds to reduce the probability of an error that could have serious consequences.

Since the results of this dissertation demonstrated that LDsplit is very dependent on the chosen binary classifier, it would be interesting to investigate how classification performance is influenced when other binary base classifiers are used. For example, one might consider using a random forest of decision trees, a hard margin SVM, logistic regression, or AdaBoost as base classifier. One can also consider allocating different base classifiers per label, as the flexibility of the LDsplit framework makes this possible. In this case a label that defines the binary classification problem at a node has an associated base classifier which is thereupon used to split the node. Wever *et al.* (2020) propose a label-wise base learner selection method, LiBRe, that optimises label-wise macro averaged performance measures. Therefore, in future work, LiBRe can for example be used to determine the base classifier used per label. Rivolli *et al.* (2020) state that the influence of base classifiers on the predictive performance of multi-label classifiers has not been considered in depth by many empirical studies. This therefore establishes an avenue for future research.

In the third stage of the empirical study presented in Chapter 4, the classification performance of four LDsplit models were compared to 17 other well-known and related multi-label learning methods across six benchmark datasets. The study can be improved by including additional LDsplit models, for example those that make use of other base classifiers or other label ordering strategies for Conditional LDsplit, such as BN strategies (Section 3.3.1). One can also consider adding a comprehensive complexity analysis to the study. Furthermore, it may be valuable to consider the performance of the models on additional multi-label benchmark datasets as well.

The empirical work presented in Chapter 6 included a synthetic study to evaluate the performance of LDsplit MDA as a multi-label input variable importance method that produces global and local input variable rankings. In future the global input variable ranking strategies of Kocev *et al*. (2013) and Petković *et al.* (2020) can be applied to the synthetic data of Chapter 6 to evaluate how these methods compare to LDsplit MDA. Local input variable rankings may also be obtained by applying a similar strategy as outlined in Section 5.5.1 to the tree-based ensembles of Kocev *et al*. (2013) and Petković *et al.* (2020). LDsplit MDI (as discussed in Section 5.5.2) can also be applied to the synthetic data to evaluate how this strategy compares to the other ranking strategies. One can also consider the global and local MDA strategy (as outlined in Section 5.5.1) as a general approach to be applied with other well-known multi-label classifiers as well. For example, for EBR each BR model in the ensemble has a corresponding OOB set. Therefore, the BR model can be used to classify the OOB sample before and after permuting an input variable. The global importance of an input variable can be defined as the average difference in the value of a multi-label evaluation measure before and after the input variable was permuted across the ensemble. However, for the local importance of an input variable the difference in performance is considered per label. Since the LDsplit MDA and LDsplit MDI global and local input variable rankings are influenced by label correlations, it would be interesting to see how these rankings compare to that of other models.

The synthetic study of Chapter 6 predominantly evaluated the performance of LDsplit MDA as a multi-label input variable importance method that produces global and local input variable rankings. In future work it would be advantageous to evaluate the performance of LDsplit MDA more extensively as a filter approach for variable selection. An independent probe strategy, such as that given by Sandrock and Steel (2016), can for example be applied to the importance values of LDsplit MDA to perform variable selection. In this case the performance of LDsplit MDA can be compared to that of other previously proposed multi-label variable selection methods. Synthetic data as well as multi-label benchmark datasets could be used for this purpose.

All empirical work presented in Chapter 4 and Chapter 6 were executed in R. The code written to fit the Random and Conditional LDsplit models and to classify observations based on the models are available in Appendix C. Furthermore, the code of LDsplit MDA is given in Appendix D. However, to allow for more convenient access to this code, future research can focus on the development of a GitHub page and a complete R-package for LDsplit. Ideally, the R package would include functions to fit Random LDsplit and Conditional LDsplit, classify data observations based on the fitted models, and allow for easy calculations of global and local LDsplit MDA variable importance measures. The functions would also ideally be flexible and allow for quick and simple specification of the number of tree-levels (m), the size of the ensemble (M), the label threshold values ($t_{Y_1}, t_{Y_2}, \ldots, t_{Y_K}$), the minimum node-size (n), the label ordering strategy for Conditional LDsplit, and per label specification of a binary base classifier selected from a range of options such as SVM, decision tree, random forest and AdaBoost. It may also be valuable to extend the developed LDsplit R-package to other programming languages such as Python.

Within the general framework of LDsplit as proposed in Chapter 3, some adaptations can be considered in future work. For example, instead of applying a minimum node-size stopping criterion when fitting an LDsplit tree, post- or pre-pruning strategies can be considered in future work. Another possible adaptation to the LDsplit algorithm is the allocation of contributing weights to the respective LDsplit trees in the ensemble. Currently, after an observation obtains a set of posterior probabilities based on the LDsplit ensemble, a final posterior probability is found per label by averaging the probabilities obtained for each of the respective labels across the ensemble. However, it might be interesting to investigate if results can improve if a weighted average is instead taken by allowing certain LDsplit trees to contribute more to the final multi-label classification. The motivation is that some LDsplit trees in the ensemble may consist of a label subset or label ordering that allows for better predictive performance. Consequently, by allowing such a tree a larger contributing weight, classification performance of unseen cases may improve. It may also be possible to allow for a label weight per LDsplit tree. Such contributing weights can be derived in different ways. For example, Xia *et al.* (2021) give a general approach that can be applied to any multi-label ensemble method to provide a label weight for each classifier in the ensemble. Xia *et al.* (2021) develop an optimisation algorithm to find the optimal solution of a regularised objective function that requires minimisation.

For LDsplit, diversity of ensemble members is achieved by a different label order and label subset per tree-structure. However, it would be interesting to investigate the effect that ensemble learning techniques such as bagging and random forest have on the classification performance of LDsplit in future studies. To fit a bagged ensemble of LDsplit trees, each tree-structure in the ensemble would be fit as before but would be based on a bootstrap sample of the training observations instead of the full set of training data. The same holds true when a random forest of LDsplit trees is fit; however, the binary classification problems formed at nodes would only be based on subsets of the available input variables as well. Note that this relates to the Triple-Random Ensemble Multi-Label Classification (TREMLC) framework (Nasierding *et al.*, 2010). TREMLC is an algorithm independent ensemble method for multi-label classification that bases the fit of each classifier in the ensemble on a randomly selected input variable subset, label subset and instance subset. In this way TREMLC uses three ways to obtain diversity between ensemble members. However, if a random forest of LDsplit trees is fit, the label order used for each tree gives a fourth way of obtaining diversity between ensemble members. Furthermore, since bootstrap sampling is used to obtain bagged and random forest ensembles, an OOB set would be formed per LDsplit tree when these ensemble techniques are applied. It would therefore be possible to use these OOB sets to implement LDsplit MDA. In this dissertation LDsplit MDA was implemented by basing the fit of each LDsplit tree on a random subsample of the training observations. Therefore, in future studies the results of this dissertation can be compared to that obtained when a bagged or random forest ensemble is instead used. This can help to establish which strategy is generally best for LDsplit MDA.

In a different line of work, LDsplit may be extended to address the multi-label imbalance problem (discussed in Section 1.4.3). Moyano *et al.* (2020) state that when an ensemble of multi-label classifiers is fit that considers only a subset of labels per classifier, the number of possible combinations of labels per subset is lower, so that the imbalance of the output space is consequently reduced in each of the ensemble members. Therefore, it is noteworthy that by currently basing the fit of each tree in an LDsplit ensemble on a subset of labels, each classifier arguably has lower imbalance in the output space compared to the full label space. This property of LDsplit and how it influences the class imbalance problem can be more extensively investigated in future studies.

Many modern multi-label image and text datasets are characterised by extremely large label collections. Therefore, research of XMLC (as discussed in Section 1.4.3) has become increasingly important. According to You *et al*. (2019) methods for XMLC can be categorised into the following four types: one-against-all, embedding-based, deep-learning based, and instance or label tree-based methods. An example of a tree-based method for XMLC is given in Prajapati and Thakkar (2022). Another tree-based method that fairs well in settings with many labels is HOMER (as described in Section 2.7.3). In future it would be interesting to investigate how LDsplit can be adapted to handle XMLC, since it appears that tree-based methods can perform well in these settings.

Lastly, since LDsplit fits an ensemble of hierarchical trees, it would be interesting to investigate in future if the LDsplit framework can be extended to handle hierarchical multi-label classification (as defined in Section 1.1).

References

Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L. and Herrera, F. 2011. KEEL Data-Mining Software Tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255-287.

Al-Otaibi, R., Flach, P. and Kull, M. 2014a. Multi-label classification: a comparative study on threshold selection methods. *Proceedings of the First International Workshop on Learning over Multiple Contexts (LMCE) at the 7th European Machine Learning and Data Mining Conference (ECML-PKDD 2014)*, 29-36.

Al-Otaibi, R., Kull, M. and Flach, P. 2014b. Lacova: a tree-based multi-label classifier using label covariance as splitting criterion. *Proceedings of the 13th International Conference on Machine Learning and Applications*, 74-79.

Atkinson, A.B. 1970. On the measurement of inequality. *Journal of Economic Theory*, 2(3):244-263.

Bi, J., Bennett, K., Embrechts, M., Breneman, C. and Song, M. 2003. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229-1243.

Blockeel, H., De Raedt, L. and Ramon, J. 1998. Top-down induction of clustering trees. *Proceedings of the 15th International Conference on Machine Learning*, 55-63.

Blockeel, H. and Struyf, J. 2002. Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research*, 3:621-650.

Bogatinovski, J., Todorovski, L., Džeroski, S. and Kocev, D. 2022. Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications*, 203:117215. Available: https://www.sciencedirect.com/science/article/pii/S0957417422005991.

Boutell, M.R., Luo, J., Shen, X. and Brown, C.M. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37:1757-1771.

Breiman, L. 2001. Random forests. *Machine Learning*, 45(1):5-32.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. 1984. *Classification and Regression Trees*. Wadsworth, New York.

Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., Hadley, S.F., Hadley, A., Betts, M., Fern, X.Z. and Irvine, J. 2013. The 9th annual MLSP competition: new methods for acoustic classification of multiple simultaneous bird species in a noisy environment. *Proceedings of the 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 1-8.

Burkov, A. 2019. *The Hundred-page Machine Learning Book*. Andriy Burkov.

Charte, D. and Charte, F. 2019. mldr: exploratory data analysis and manipulation of multi-label data sets [Online]. Available: https://CRAN.R-project.org/package=mldr. [2022, October].

Charte, F. and Charte, D. 2015. Working with multilabel datasets in R: the mldr package. *R Journal*, 7(2):149-164.

Charte, F., Rivera, A.J., Charte, D., del Jesus, M.J. and Herrera, F. 2018. Tips, guidelines and tools for managing multi-label datasets: The mldr.datasets R package and the Cometa data repository. *Neurocomputing*, 289:68-85.

Charte, F., Rivera, A.J., del Jesus, M.J. and Herrera, F. 2013. A first approach to deal with imbalance in multi-label datasets. *Proceedings of the 8th International Conference on Hybrid Artificial Intelligent Systems (HAIS),* 150–160.

Charte, F., Rivera, A.J., del Jesus, M.J. and Herrera, F. 2014. Concurrence among imbalanced labels and its influence on multilabel resampling algorithms. *Proceedings of the 9th International Conference on Hybrid Artificial Intelligence Systems*, 110-121.

Charte, F., Rivera, A.J., del Jesus, M.J. and Herrera, F. 2015. Addressing imbalance in multilabel classification: measures and random resampling algorithms. *Neurocomputing*, 163:3-16.

Chasalow, S. 2012. combinat: combinatorics utilities [Online]. Available: https://cran.r-project.org/package=combinat. [2022, October].

Chen, H., Li, T., Fan, X. and Luo, C. 2019. Feature selection for imbalanced data based on neighborhood rough sets. *Information Sciences*, 483:1-20.

Chen, Y.T. and Chen, M.C. 2011. Using chi-square statistics to measure similarities for text categorization. *Expert Systems with Applications*, 38(4):3085-3090.

Cheng, B., Liu, M., Zhang, D. and Shen, D. 2019. Robust multi-label transfer feature learning for early diagnosis of Alzheimer's disease. *Brain Imaging and Behavior*, 13(1):138-153.

Chu, W.T. and Guo, H.J. 2017. Movie genre classification based on poster images with deep neural networks. *Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes*, 39-45.

Clare, A. and King, R.D. 2001. Knowledge discovery in multi-label phenotype data. *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, 42–53.

Daniel, F., Weston, S. and Tenenbaum, D. 2022. doParallel: foreach parallel adaptor for the 'parallel' package [Online]. Available: https://cran.r-project.org/package=doParallel. [2022, October].

Daniels, Z.A. and Metaxas, D.N. 2017. Addressing imbalance in multi-label classification using structured Hellinger forests. *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 1826–1832.

Da Silva, P.N., Gonçalves, E.C., Plastino, A. and Freitas, A.A. 2014. Distinct chains for different instances: an effective strategy for multi-label classifier chains. *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 453-468.

Dembczynski, K., Cheng, W. and Hüllermeier, E. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 279-286.

Dfuf, I.A., Pérez-Minayo, J.F., Mcwilliams, J.M.M. and Fernández, C.G. 2020. Variable importance analysis in imbalanced datasets: A new approach. *IEEE Access*, 8:127404-127430.

Dietterich, T.G. 2000. Ensemble methods in machine learning. *Multiple Classifier Systems*. Springer, Berlin, Heidelberg, 1-15.

Dollár, P. and Zitnick, C.L. 2013. Structured forests for fast edge detection. *Proceedings of the IEEE International Conference on Computer Vision*, 1841-1848.

Doquire, G. and Verleysen, M. 2013. Mutual information-based feature selection for multilabel classification. *Neurocomputing*, 122:148-155.

Dreyfus, G. and Guyon, I. 2006. Assessment methods. *Feature Extraction: Foundations and Application*. Springer, Berlin, Heidelberg, 65-88.

Duygulu, P., Barnard, K., de Freitas, J. and Forsyth, D. 2002. Object recognition as machine translation: learning a lexicon for a fixed image vocabulary. *Proceedings of the 7th European Conference on Computer Vision*, 349–354.

Elisseeff, A. and Weston, J. 2001. A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems*, 14:681-687.

Enrique Sucar, L., Bielza, C., Morales, E.F., Hernandez-Leal, P., Zaragoza, J.H. and Larrañaga, P. 2014. Multi-label classification with Bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41:14-22.

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E. and Brinker, K. 2008. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133-153.

Gjorgjevikj, D., Madjarov, G. and Dzeroski, S. 2013. Hybrid decision tree architecture utilizing local svms for efficient multi-label learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(7).

Gonçalves, E.C., Plastino, A. and Freitas, A.A. 2013. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. *Proceedings of the IEEE 25th International Conference on Tools with Artificial Intelligence*, 469-476.

Gonçalves, E.C., Plastino, A. and Freitas, A.A. 2015. Simpler is better: a novel genetic algorithm to induce compact multi-label chain classifiers. *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, 559-566.

Hasan, M.R., Jamil, M. and Rahman, M.G.R.M.S. 2004. Speaker identification using mel frequency cepstral coefficients. *Proceedings of the 3rd International Conference on Electrical and Computer Engineering*, 1(4):565-568.

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference and Prediction (Second edition)*. Springer.

Hollander, M., Wolfe, D.A. and Chicken, E. 2014. *Nonparametric Statistical Methods (Third edition)*. John Wiley and Sons.

Huang, S. J. and Zhou, Z. H. 2012. Multi-label learning by exploiting label correlations locally. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):949-955.

Hüllermeier, E., Fürnkranz, J., Cheng, W. and Brinker, K. 2008. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172:1897-1916.

Iman, R.L. and Davenport, J.M. 1980. Approximations of the critical region of the Friedman statistic. *Communications in Statistics-Theory and Methods*, 9(6):571-595.

Jun, X., Lu, Y., Lei, Z. and Guolun, D. 2019. Conditional entropy based classifier chains for multi-label classification. *Neurocomputing*, 335:185-194.

Kaggle: NLP on research articles [Online]. 2020. Available: https://www.kaggle.com/datasets/vetrirah/janatahack-independence-day-2020-mlhackathon. [2022, October].

Kaggle: Rainforest Connection Species Audio Detection [Online]. 2020. Available: https://www.kaggle.com/competitions/rfcx-species-audio-detection/overview/ description. [2022, October].

Katakis, I., Tsoumakas, G. and Vlahavas, I. 2008. Multilabel text classification for automated tag suggestion. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) Discovery Challenge*, 75-83.

Kira, K. and Rendell, L.A. 1992. A practical approach to feature selection. *Proceedings of the 9th International Conference on Machine Learning*, 249-256.

Klimt, B. and Yang, Y. 2004. The enron corpus: a new dataset for email classification research. *Proceedings of the 15th European Conference on Machine Learning*, 217-226.

Kocev, D., Slavkov, I. and Dzeroski, S. 2013. Feature ranking for multi-label classification using predictive clustering trees. *Proceedings of the International Workshop on Solving Complex Machine Learning Problems with Ensemble Methods in conjunction with ECML/PKDD*, 56-68.

Kocev, D., Vens, C., Struyf, J. and Džeroski, S. 2007. Ensembles of multi-objective decision trees. *Proceedings of the 18th European Conference on Machine Learning*, 624-631.

Kohavi, R. and John, G.H. 1997. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273-324.

Koller, D. and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Kong, D., Ding, C., Huang, H. and Zhao, H. 2012. Multi-label ReliefF and F-statistic feature selections for image annotation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2352-2359.

Kononenko, I. 1994. Estimating attributes: analysis and extensions of Relief. *Proceedings of the European Conference on Machine Learning*, 171-182.

Kostovska, A., Bogatinovski, J., Džeroski, S., Kocev, D. and Panov, P. 2022. A catalogue with semantic annotations makes multilabel datasets FAIR. *Scientific Reports*, 12(1):1-11.

Kourentzes, N., Svetunkov, I. and Schaer, O. 2022. tsutils: time series exploration, modelling and forecasting [Online]. Available: https://CRAN.R-project.org/package=tsutils. [2022, October].

Levatić, J., Kocev, D. and Džeroski, S. 2015. The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, 45(2): 247-271.

Lin, W. and Xu, D. 2016. Imbalanced multi-label learning for identifying antimicrobial peptides and their functional types. *Bioinformatics*, 32(24):3745-3752.

Liu, H. and Motoda, H. 2007. *Computational Methods of Feature Selection*. CRC Press.

Liu, J., Chang, W.C., Wu, Y. and Yang, Y. 2017. Deep learning for extreme multi-label text classification. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 115-124.

Liu, J., Lin, Y., Li, Y., Weng, W. and Wu, S. 2018. Online multi-label streaming feature selection based on neighborhood rough set. *Pattern Recognition*, 84:273-287.

Liu, W. and Tsang, I. 2015. On the optimality of classifier chain for multi-label classification. *Advances in Neural Information Processing Systems*, 28:712-720.

Liu, W., Wang, H., Shen, X. and Tsang, I. 2021. The emerging trends of multi-label learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7955-7974.

Loza Mencía, E. and Fürnkranz, J. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 50-65.

Luaces, O., Dıez, J., del Coz, J.J., Barranquero, J. and Bahamonde, A. 2012. Synthetic datasets for sound experimental evaluation of multilabel classifiers. Technical Report, Universidad de Oviedo, Oviedo, Spain.

Madjarov, G., Kocev, D., Gjorgjevikj, D. and Džeroski, S. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084-3104.

McCallum, A.K. 1999. Multi-label text classification with a mixture model trained by EM. *Working Notes of the AAAI'99 Workshop on Text Learning*. Orlando, Florida.

Mencía, E.L., Park, S.H. and Fürnkranz, J, 2010. Efficient voting prediction for pairwise multilabel classification. *Neurocomputing*, 73(7-9):1164-1176.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C. and Lin, C. 2022. e1071: Misc functions of the Department of Statistics, Probability Theory Group (Formerly: E1071) [Online]. Available: https://CRAN.R-project.org/package=e1071. [2022, October].

Meyer, P.E. 2022. infotheo: information-theoretic measures [Online]. Available: https://cran.r-project.org/package=infotheo. [2022, October].

Ming, O. 2012. FRACTION: numeric number into fraction [Online]. Available: https://cran.r-project.org/package=FRACTION. [2022, October].

Moyano, J.M., Gibaja, E.L., Cios, K.J. and Ventura, S. 2018. Review of ensembles of multilabel classifiers: models, experimental study and prospects. *Information Fusion*, 44:33-45.

Moyano, J.M., Gibaja, E.L., Cios, K.J. and Ventura, S. 2020. Combining multi-label classifiers based on projections of the output space using evolutionary algorithms. *Knowledge-Based Systems*, 196(10):57-70.

Nasierding, G., Kouzani, A.Z. and Tsoumakas, G. 2010. A triple-random ensemble classification method for mining multi-label data. *Proceedings of the IEEE International Conference on Data Mining Workshops*, 49-56.

Nasierding, G., Tsoumakas, G. and Kouzani, A.Z. 2009. Clustering based multi-label classification for image annotation and retrieval. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 4514-4519.

Nemenyi, P.B. 1963. *Distribution-free Multiple Comparisons.* Published doctoral dissertation. Princeton University.

Olson, M. 2017. JOUSBoost: implements under/oversampling for probability estimation [Online]. Available: https://CRAN.R-project.org/package=JOUSBoost. [2022, October].

Oman, S.D. 2009. Easily simulated multivariate binary distributions with given positive and negative correlations. *Computational Statistics and Data Analysis*, 53(4):999-1005.

Papanikolaou, Y., Tsoumakas, G. and Katakis, I. 2018. Hierarchical partitioning of the output space in multi-label data. *Data and Knowledge Engineering*, 116:42-60.

Park, S.H. and Fürnkranz, J. 2007. Efficient pairwise classification. *Proceedings of the European Conference on Machine Learning*, 658-665.

Parkhi, O.M., Vedaldi, A., Zisserman, A. and Jawahar, C.V. 2012. Cats and dogs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3498-3505.

Pereira, R.B., Plastino, A., Zadrozny, B. and Merschmann, L.H. 2018. Categorizing feature selection methods for multi-label classification. *Artificial Intelligence Review*, 49(1):57-78.

Pestian, J., Brew, C., Matykiewicz, P., Hovermale, D.J., Johnson, N., Cohen, K.B. and Duch, W. 2007. A shared task involving multi-label classification of clinical free text. *Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing*, 97-104.

Petković, M., Džeroski, S. and Kocev, D. 2020. Multi-label feature ranking with ensemble methods. *Machine Learning*, 109(11):2141-2159.

Petterson, J. and Caetano, T. 2011. Submodular multi-label learning. *Advances in Neural Information Processing Systems*, 24:1512-1520.

Prajapati, P. and Thakkar, A. 2022. Performance improvement of extreme multi-label classification using K-way tree construction with parallel clustering algorithm. *Journal of King Saud University-Computer and Information Sciences*, 34(8): 6354-6364.

Qi, G.J., Hua, X.S., Rui, Y., Tang, J. and Zhang, H.J. 2008. Two-dimensional multilabel active learning with an efficient online adaptation model for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1880-1897.

Quinlan, J. R. 1995. C4.5: programs for machine learning, Morgan Kaufmann, San Mateo, California (1992). *Artificial Intelligence*, 10(3):475-476.

Rabiner, L. and Juang, B.H. 1993. *Fundamentals of Speech Recognition*. Prentice-Hall.

Ramírez-Corona, M., Sucar, L.E. and Morales, E.F. 2014. Chained path evaluation for hierarchical multi-label classification. *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference*, 502-507.

Ratnarajah, N. and Qiu, A. 2014. Multi-label segmentation of white matter structures: application to neonatal brains. *NeuroImage*, 102:913-922.

Read, J. 2008. A pruned problem transformation method for multi-label classification. *Proceedings of the 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, 143-150.

Read, J., Bifet, A., Holmes, G. and Pfahringer, B. 2012. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88(1):243-272.

Read, J., Martino, L. and Luengo, D. 2013. Efficient Monte Carlo optimization for multi-label classifier chains. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 3457-3461.

Read, J., Martino, L. and Luengo, D. 2014. Efficient Monte Carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535-1546.

Read, J., Pfahringer, B., Holmes, G. and Frank, E. 2009. Classifier chains for multi-label classification. *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 254-269.

Read, J., Pfahringer, B., Holmes, G. and Frank, E. 2011. Classifier chains for multi-label classification. *Machine Learning*, 85:333-359.

Read, J., Pfahringer, B., Holmes, G. and Frank, E. 2021. Classifier chains: a review and perspectives. *Journal of Artificial Intelligence Research*, 70:683-718.

Read, J., Reutemann, P., Pfahringer, B. and Holmes, G. 2016. MEKA: a multi-label/multi-target extension to WEKA. *Journal of Machine Learning Research*, 17:1-5.

Ren, K. 2021. rlist: a toolbox for non-tabular data manipulation [Online]. Available: https://cran.r-project.org/package=rlist. [2022, October].

Ren, Z., Peetz, M.H., Liang, S., Van Dolen, W. and De Rijke, M. 2014. Hierarchical multi-label classification of social text streams. *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 213-222.

Reyes, O., Morell, C. and Ventura, S. 2015. Scalable extensions of the ReliefF algorithm for weighting and selecting features on the multi-label learning context. *Neurocomputing*, 161:168-182.

Rivolli, A. 2021. utiml: utilities for multi-label learning [Online]. Available: https://CRAN.R-project.org/package=utiml. [2022, October].

Rivolli, A. and de Carvalho, A.C. 2018. The utiml package: multi-label classification in R. *R Journal*, 10(2):24-38.

Rivolli, A., Parker, L.C. and de Carvalho, A.C. 2017. Food truck recommendation using multi-label classification. *Proceedings of EPIA Conference 2017: Progress in Artificial Intelligence*, 585-596.

Rivolli, A., Read, J., Soares, C., Pfahringer, B. and de Carvalho, A.C. 2020. An empirical analysis of binary transformation strategies and base algorithms for multi-label learning. *Machine Learning*, 109(8):1509-1563.

Robnik-Šikonja, M. and Kononenko, I. 2003. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1):23-69.

Ruiz, R., Riquelme, J.C. and Aguilar-Ruiz, J.S. 2005. Heuristic search over a ranking for feature selection. *Proceedings of the International Work-Conference on Artificial Neural Networks*, 742-749.

Sajnani, H., Saini, V., Kumar, K., Gabrielova, E., Choudary, P. and Lopes, C. 2013. Classifying Yelp reviews into relevant categories. Available: http://www.ics.uci.edu/~vpsaini/. [2022, October].

Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

Sanden, C. and Zhang, J.Z. 2011. Enhancing multi-label music genre classification through ensemble techniques. *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 705-714.

Sandrock, T. and Steel, S.J. 2016. Probe variables for multi-label variable selection. Technical Report, University of Stellenbosch, Stellenbosch, South Africa.

Sandrock, T. and Steel, S.J. 2017. An algorithm for generating multi-label classification data. Technical Report, University of Stellenbosch, Stellenbosch, South Africa.

Schapire, R.E. and Singer, Y. 2000. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2):135-168.

Scornet, E. 2020. Trees, forests, and impurity-based variable importance. *arXiv preprint arXiv:2001.04295*.

Senge, R., Coz, J.J.D. and Hüllermeier, E. 2014. On the problem of error propagation in classifier chains for multi-label classification. *Data Analysis, Machine Learning and Knowledge Discovery*. Springer, 163-170.

Shannon, C.E. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379-423.

Shao, H., Li, G., Liu, G. and Wang, Y. 2013. Symptom selection for multi-label data of inquiry diagnosis in traditional Chinese medicine. *Science China Information Sciences*, 56(5):1-13.

Snoek, C.G., Worring, M., Van Gemert, J.C., Geusebroek, J.M. and Smeulders, A.W. 2006. The challenge problem for automated detection of 101 semantic concepts in multimedia. *Proceedings of the 14th ACM International Conference on Multimedia*, 421-430.

Spolaôr, N., Cherman, E.A., Monard, M.C. and Lee, H.D. 2013. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135-151.

Spolaôr, N. and Tsoumakas, G. 2013. Evaluating feature selection methods for multi-label text classification. *Proceedings of the First Workshop on Bio-Medical Semantic Indexing and Question Answering (BioASQ)*. Valencia, Spain.

Spyromitros-Xioufis, E., Tsoumakas, G. and Vlahavas, I. 2011. Multi-label learning approaches for music instrument recognition. *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, 734-743.

Srivastava, A.N. and Zane-Ulman, B. 2005. Discovering recurring anomalies in text reports regarding complex space systems. *Proceedings of the 2005 IEEE Aerospace Conference*, 3853-3862.

Stoppiglia, H., Dreyfus, G., Dubois, R. and Oussar, Y. 2003. Ranking a random feature for variable and feature selection. *The Journal of Machine Learning Research*, 3:1399-1414.

Strobl, C., Boulesteix, A.L., Kneib, T., Augustin, T. and Zeileis, A. 2008. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):1-11.

Sun, L., Ji, S. and Ye, J. 2013. *Multi-label Dimensionality Reduction*. CRC Press.

Sun, L., Wang, T., Ding, W., Xu, J. and Lin, Y. 2021. Feature selection using Fisher score and multilabel neighborhood rough sets for multilabel classification. *Information Sciences*, 578:887-912.

Szymanski, P. and Kajdanowicz, T. 2018. scikit-multilearn [Online]. Available: https://github.com/scikit-multilearn/scikit-multilearn. [2022, October].

Szymanski, P. and Kajdanowicz, T. 2019. scikit-multilearn: a scikit-based Python environment for performing multi-label classification. *Journal of Machine Learning Research*, 20(1):209-230.

Tenenboim-Chekina, L., Rokach, L. and Shapira, B. 2010. Identification of label dependencies for multi-label classification. *Proceedings of the 2nd International Workshop on Learning from Multi-Label Data*, 53-60.

Therneau, T., Atkinson, B. and Ripley, B. 2022. rpart: recursive partitioning and regression trees [Online]. Available: https://CRAN.R-project.org/package=rpart. [2022, October].

Tomás, J.T., Spolaôr, N., Cherman, E.A. and Monard, M.C. 2014. A framework to generate synthetic multi-label datasets. *Electronic Notes in Theoretical Computer Science*, 302:155-176.

Trohidis, K., Tsoumakas, G., Kalliris, G. and Vlahavas, I.P. 2008. Multi-label classification of music into emotions. *Proceedings of the 9th International Conference on Music Information Retrieval*, 8:320-330.

Tsoumakas, G., Dimou, A., Spyromitros, E., Mezaris, V., Kompatsiaris, I. and Vlahavas, I. 2009. Correlation-based pruning of stacked binary relevance models for multi-label learning. *Proceedings of the 1st International Workshop on Learning from Multi-label Data*, 101-116.

Tsoumakas, G. and Katakis, I. 2007. Multi-label classification: an overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1-13.

Tsoumakas, G., Katakis, I. and Vlahavas, I., 2008. Effective and efficient multilabel classification in domains with large number of labels. *Proceedings of ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 21:53-59.

Tsoumakas, G., Katakis, I. and Vlahavas, I. 2010. Mining multi-label data. *Data Mining and Knowledge Discovery Handbook*. Springer, Berlin, Heidelberg.

Tsoumakas, G., Katakis, I. and Vlahavas, I. 2011a. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079-1089.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J. and Vlahavas, I. 2011b. Mulan: a Java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411-2414.

Tuv, E., Borisov, A. and Torkkola, K. 2008. Ensemble-based variable selection using independent probes. *Computational Methods of Feature Selection*. Chapman and Hall/CRC, 131-145.

Tzanetakis, G. and Cook, P. 2002. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293-302.

Ueda, N. and Saito, K. 2002. Parametric mixture models for multi-labeled text. *Advances in Neural Information Processing Systems*, 15:721–728.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C. and Xu, W. 2016. Cnn-rnn: a unified framework for multi-label image classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2285-2294.

Wang, S., Wang, J., Wang, Z. and Ji, Q. 2014. Enhancing multi-label classification by modelling dependencies among labels. *Pattern Recognition*, 47(10):3405-3413.

Wehrmann, J., Cerri, R. and Barros, R. 2018. Hierarchical multi-label classification networks. *Proceedings of the International Conference on Machine Learning*, 5225-5234.

Wever, M., Tornede, A., Mohr, F. and Hüllermeier, E. 2020. LiBRe: label-wise selection of base learners in binary relevance for multi-label classification. *Proceedings of the International Symposium on Intelligent Data Analysis*, 561-573.

Wolberg, W.H., Street, W.N. and Mangasarian, O.L. 1992. Breast cancer Wisconsin (diagnostic) data set. *UCI Machine Learning Repository*. Available: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29 [2022, October].

Wu, Q., Tan, M., Song, H., Chen, J. and Ng, M.K. 2016. ML-FOREST: a multi-label tree ensemble method for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2665-2680.

Xia, Y., Chen, K. and Yang, Y. 2021. Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*, 557:421-442.

Xie, Y., Li, D., Zhang, D. and Shuang, H. 2017. An improved multi-label relief feature selection algorithm for unbalanced datasets. *Proceedings of the International Conference on Intelligent and Interactive Systems and Applications*, 141-151.

Yin, L., Ge, Y., Xiao, K., Wang, X. and Quan, X. 2013. Feature selection for high-dimensional imbalanced data. *Neurocomputing*, 105:3-11.

You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H. and Zhu, S. 2019. AttentionXML: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 32:5812-5822.

Zaragoza, J.H., Sucar, L.E., Morales, E.F., Bielza, C. and Larrañaga, P. 2011. Bayesian chain classifiers for multidimensional classification. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence,* 2192–2195.

Zdravevski, E., Lameski, P., Kulakov, A., Jakimovski, B., Filiposka, S. and Trajanov, D. 2015. Feature ranking based on information gain for large classification problems with MapReduce. *Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2:186-191.

Zhang, L. and Duan, Q. 2019. A feature selection method for multi-label text based on feature importance. *Applied Sciences*, 9(4):665-689.

Zhang, M.L. and Zhang, K. 2010. Multi-label learning by exploiting label dependency. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 999-1008.

Zhang, M.L. and Zhou, Z.H. 2006. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Zhang, M.L. and Zhou, Z.H. 2007. ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038-2048.

Zhang, M.L. and Zhou, Z.H. 2013. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819-1837.

Zhou, H., Yang, Y. and Shen, H.B. 2017. Hum-mPLoc 3.0: prediction enhancement of human protein subcellular localization through modeling the hidden correlations of gene ontology and functional domain features. *Bioinformatics*, 33(6):843-853.

Zhu, Y., Kwok, J.T. and Zhou, Z.H. 2017. Multi-label learning with global and local label correlation. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1081-1094.

Appendix A: Detailed results for empirical study of Chapter 4

This appendix contains the detailed results of the Random and Conditional LDsplit empirical study described in Chapter 4.

Appendix A.1 gives the detailed Random LDsplit cross-validation results as described in Section 4.2. For all the datasets the average performance of each m and M combination over the five cross-validation folds are provided. The standard deviation of performance over the five folds is given in brackets and the model that produces the best performance for each evaluation measure is given in bold for each dataset.

Appendix A.2 illustrates the average model performance and standard deviation after refitting each of the Random LDsplit models five times to classify a single validation fold of the *Emotions* data.

Appendix A.3 gives the detailed Conditional LDsplit cross-validation results as described in Section 4.2. For all the datasets the average performance of each m and M combination over the five cross-validation folds are provided. The standard deviation of performance over the five folds is given in brackets and the model that produces the best performance for each evaluation measure is given in bold for each dataset.

Appendix A.4 gives a comparison of the cross-validation results of Random and Conditional LDsplit as described in Section 4.3. For each value of m, the model that produces the best performance for each evaluation measure is given in bold and the percentage decrease in performance is reported in brackets for the other two models. Furthermore, for each evaluation measure, the model that gives the best performance across all values of m is highlighted.

Appendix A.5 gives the mean rank diagrams as described in Section 4.4. The diagrams are based on the Friedman and Nemenyi tests conducted using the performance of $A = 21$ models across $B = 5$ datasets, namely *Emotions*, *Scene*, *Yeast*, *Medical* and *Corel5k*.

A.1 Results for different choices of m and M for Random LDsplit

EMOTIONS (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	6	0.190 (0.009)	0.508 (0.028)	0.652 (0.021)	0.279 (0.047)	0.619 (0.016)	0.649 (0.017)
	18	0.194 (0.014)	0.493 (0.037)	0.639 (0.033)	0.271 (0.035)	0.605 (0.037)	0.636 (0.039)
	30	0.196 (0.015)	0.488 (0.033)	0.633 (0.032)	0.268 (0.035)	0.603 (0.035)	0.632 (0.038)
m=3	6	0.189 (0.009)	0.513 (0.015)	0.648 (0.021)	0.289 (0.039)	0.622 (0.015)	0.648 (0.024)
	18	0.194 (0.013)	0.505 (0.030)	0.647 (0.029)	0.279 (0.029)	0.608 (0.031)	0.640 (0.034)
	30	0.192 (0.015)	0.507 (0.030)	0.648 (0.032)	0.284 (0.029)	0.613 (0.035)	0.643 (0.038)
	60	0.192 (0.017)	0.506 (0.034)	0.647 (0.033)	0.279 (0.026)	0.610 (0.035)	0.643 (0.041)
	100	0.191 (0.017)	0.503 (0.037)	0.646 (0.035)	0.281 (0.041)	0.613 (0.036)	0.642 (0.041)
m=4	6	0.199 (0.008)	0.509 (0.029)	0.642 (0.024)	0.274 (0.036)	0.598 (0.037)	0.639 (0.027)
	18	0.195 (0.013)	0.501 (0.033)	0.640 (0.028)	0.279 (0.027)	0.609 (0.032)	0.639 (0.035)
	30	0.191 (0.016)	0.524 (0.031)	0.656 (0.035)	0.294 (0.023)	0.616 (0.037)	0.650 (0.039)
	60	0.191 (0.013)	0.517 (0.034)	0.652 (0.028)	0.289 (0.036)	0.620 (0.034)	0.650 (0.036)
	100	0.192 (0.016)	0.517 (0.029)	0.654 (0.029)	0.279 (0.023)	0.616 (0.035)	0.648 (0.035)
	240	0.189 (0.017)	0.514 (0.034)	0.653 (0.034)	0.286 (0.040)	0.620 (0.038)	0.650 (0.040)
	300	0.190 (0.015)	0.515 (0.030)	0.651 (0.030)	0.286 (0.032)	0.617 (0.034)	0.649 (0.037)
m=5	6	0.203 (0.010)	0.499 (0.019)	0.624 (0.018)	0.274 (0.026)	0.576 (0.040)	0.617 (0.021)
	18	0.196 (0.015)	0.517 (0.023)	0.648 (0.027)	0.281 (0.031)	0.609 (0.035)	0.642 (0.037)
	30	0.190 (0.018)	0.521 (0.040)	0.654 (0.033)	0.291 (0.046)	**0.624 (0.040)**	0.652 (0.042)
	60	0.192 (0.018)	0.520 (0.029)	0.655 (0.028)	0.279 (0.042)	0.614 (0.036)	0.648 (0.039)
	100	**0.186 (0.018)**	**0.532 (0.038)**	**0.667 (0.035)**	0.294 (0.042)	**0.624 (0.041)**	**0.659 (0.043)**
	240	0.190 (0.018)	0.519 (0.031)	0.654 (0.033)	0.284 (0.027)	0.614 (0.039)	0.649 (0.042)
	300	0.190 (0.016)	0.521 (0.028)	0.654 (0.029)	0.286 (0.028)	0.617 (0.037)	0.651 (0.038)
	720	0.191 (0.016)	0.521 (0.029)	0.653 (0.030)	0.286 (0.035)	0.617 (0.037)	0.649 (0.038)
m=6	6	0.209 (0.021)	0.496 (0.044)	0.620 (0.046)	0.263 (0.024)	0.576 (0.049)	0.611 (0.063)
	18	0.196 (0.016)	0.523 (0.041)	0.647 (0.036)	0.289 (0.054)	0.611 (0.042)	0.644 (0.041)
	30	0.192 (0.013)	0.523 (0.023)	0.649 (0.024)	0.279 (0.025)	0.611 (0.027)	0.648 (0.033)
	60	0.190 (0.018)	0.524 (0.030)	0.657 (0.026)	0.286 (0.034)	0.618 (0.038)	0.652 (0.040)
	100	0.189 (0.015)	0.531 (0.023)	0.660 (0.024)	0.291 (0.034)	0.620 (0.035)	0.656 (0.035)
	240	0.189 (0.016)	0.531 (0.023)	0.661 (0.025)	**0.297 (0.027)**	0.619 (0.035)	0.655 (0.036)
	300	0.190 (0.015)	0.526 (0.025)	0.655 (0.027)	0.291 (0.033)	0.616 (0.033)	0.651 (0.034)
	720	0.188 (0.016)	**0.532 (0.026)**	0.661 (0.029)	**0.297 (0.029)**	0.620 (0.036)	0.655 (0.036)

EMOTIONS (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	6	0.234 (0.019)	0.438 (0.062)	0.576 (0.047)	0.202 (0.084)	0.558 (0.046)	0.577 (0.044)
	18	0.228 (0.020)	0.451 (0.051)	0.601 (0.042)	0.187 (0.050)	0.579 (0.038)	0.598 (0.042)
	30	0.230 (0.017)	0.455 (0.051)	0.602 (0.046)	0.194 (0.048)	0.579 (0.039)	0.598 (0.040)
m=3	6	0.236 (0.014)	0.447 (0.047)	0.585 (0.051)	0.192 (0.036)	0.568 (0.046)	0.579 (0.042)
	18	0.217 (0.018)	0.468 (0.042)	0.601 (0.037)	0.233 (0.042)	0.577 (0.047)	0.605 (0.047)
	30	0.215 (0.014)	0.463 (0.032)	0.600 (0.023)	0.228 (0.042)	0.593 (0.035)	0.603 (0.035)
	60	0.217 (0.019)	0.467 (0.052)	0.600 (0.039)	0.230 (0.072)	0.592 (0.052)	0.603 (0.039)
	100	0.213 (0.013)	0.475 (0.041)	0.610 (0.029)	0.238 (0.056)	0.600 (0.038)	0.610 (0.031)
m=4	6	0.235 (0.012)	0.426 (0.029)	0.567 (0.032)	0.184 (0.024)	0.539 (0.041)	0.565 (0.031)
	18	0.207 (0.018)	0.476 (0.032)	0.608 (0.030)	**0.261 (0.049)**	0.576 (0.033)	0.613 (0.031)
	30	0.207 (0.016)	**0.482 (0.038)**	**0.616 (0.037)**	0.251 (0.047)	0.602 (0.038)	**0.617 (0.032)**
	60	0.209 (0.011)	0.465 (0.035)	0.598 (0.023)	0.235 (0.054)	0.590 (0.039)	0.605 (0.029)
	100	0.206 (0.012)	0.476 (0.035)	0.607 (0.024)	0.241 (0.054)	0.591 (0.041)	0.609 (0.030)
	240	**0.204 (0.015)**	0.479 (0.039)	0.614 (0.031)	0.243 (0.066)	0.603 (0.045)	**0.617 (0.027)**
	300	0.207 (0.016)	0.475 (0.044)	0.609 (0.034)	0.243 (0.058)	0.595 (0.051)	0.612 (0.031)
m=5	6	0.206 (0.020)	0.470 (0.045)	0.611 (0.039)	0.233 (0.069)	0.576 (0.051)	0.610 (0.050)
	18	0.207 (0.007)	0.469 (0.027)	0.603 (0.018)	0.238 (0.044)	0.565 (0.027)	0.607 (0.019)
	30	0.209 (0.014)	0.470 (0.038)	0.605 (0.027)	0.233 (0.059)	0.591 (0.045)	0.607 (0.029)
	60	0.212 (0.020)	0.473 (0.048)	0.605 (0.036)	0.246 (0.069)	0.584 (0.053)	0.596 (0.048)
	100	0.213 (0.025)	0.475 (0.053)	0.604 (0.045)	0.256 (0.059)	0.575 (0.064)	0.596 (0.046)
	240	0.211 (0.017)	0.474 (0.041)	0.603 (0.042)	0.246 (0.044)	**0.611 (0.053)**	0.603 (0.035)
	300	0.217 (0.014)	0.458 (0.025)	0.596 (0.022)	0.228 (0.046)	0.598 (0.048)	0.593 (0.017)
	720	0.215 (0.013)	0.461 (0.027)	0.596 (0.025)	0.235 (0.045)	0.586 (0.033)	0.600 (0.019)
m=6	6	0.224 (0.027)	0.427 (0.051)	0.555 (0.057)	0.215 (0.048)	0.506 (0.061)	0.558 (0.054)
	18	0.218 (0.019)	0.435 (0.038)	0.578 (0.044)	0.207 (0.048)	0.527 (0.035)	0.561 (0.040)
	30	0.210 (0.006)	0.450 (0.028)	0.586 (0.018)	0.235 (0.046)	0.562 (0.034)	0.585 (0.020)
	60	0.206 (0.014)	0.458 (0.031)	0.590 (0.039)	0.240 (0.034)	0.562 (0.025)	0.593 (0.038)
	100	**0.204 (0.018)**	0.461 (0.052)	0.597 (0.039)	0.243 (0.069)	0.563 (0.045)	0.595 (0.042)
	240	0.210 (0.014)	0.443 (0.043)	0.580 (0.027)	0.225 (0.050)	0.545 (0.044)	0.578 (0.035)
	300	0.208 (0.013)	0.450 (0.036)	0.588 (0.032)	0.233 (0.045)	0.555 (0.036)	0.584 (0.031)
	720	0.206 (0.014)	0.450 (0.039)	0.590 (0.030)	0.230 (0.049)	0.553 (0.044)	0.585 (0.031)

SCENE (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	6	0.083 (0.007)	0.677 (0.026)	0.761 (0.019)	0.637 (0.031)	0.754 (0.021)	0.745 (0.021)
	18	0.082 (0.007)	0.677 (0.026)	0.764 (0.019)	0.635 (0.032)	0.758 (0.021)	0.748 (0.021)
	30	0.082 (0.007)	0.677 (0.028)	0.763 (0.020)	0.634 (0.034)	0.758 (0.021)	0.748 (0.022)
m=3	6	0.080 (0.006)	0.691 (0.022)	0.767 (0.014)	0.656 (0.030)	0.764 (0.016)	0.754 (0.016)
	18	0.081 (0.006)	0.689 (0.026)	0.767 (0.017)	0.650 (0.034)	0.763 (0.019)	0.752 (0.019)
	30	0.081 (0.006)	0.683 (0.026)	0.766 (0.018)	0.642 (0.032)	0.759 (0.020)	0.750 (0.019)
	60	0.081 (0.006)	0.686 (0.027)	0.768 (0.016)	0.645 (0.034)	0.763 (0.018)	0.752 (0.019)
	100	0.081 (0.007)	0.684 (0.031)	0.768 (0.019)	0.643 (0.038)	0.762 (0.021)	0.752 (0.022)
m=4	6	0.081 (0.008)	0.701 (0.026)	0.767 (0.022)	0.668 (0.033)	0.764 (0.023)	0.754 (0.024)
	18	0.082 (0.006)	0.699 (0.023)	0.766 (0.016)	0.663 (0.030)	0.761 (0.018)	0.751 (0.018)
	30	0.081 (0.006)	0.693 (0.024)	0.766 (0.017)	0.655 (0.030)	0.761 (0.019)	0.752 (0.019)
	60	0.080 (0.006)	0.700 (0.027)	0.769 (0.017)	0.663 (0.033)	0.766 (0.019)	0.755 (0.020)
	100	0.079 (0.006)	0.701 (0.023)	0.773 (0.016)	0.665 (0.030)	0.768 (0.018)	0.759 (0.018)
	240	0.080 (0.006)	0.697 (0.027)	0.771 (0.017)	0.660 (0.034)	0.767 (0.019)	0.757 (0.019)
	300	0.080 (0.006)	0.698 (0.026)	0.771 (0.017)	0.660 (0.032)	0.766 (0.020)	0.756 (0.020)
m=5	6	0.080 (0.007)	0.712 (0.025)	0.766 (0.023)	0.684 (0.029)	0.765 (0.022)	0.755 (0.022)
	18	0.080 (0.006)	0.712 (0.021)	0.769 (0.018)	0.680 (0.028)	0.768 (0.019)	0.757 (0.019)
	30	0.079 (0.006)	0.713 (0.024)	0.771 (0.018)	0.684 (0.030)	0.771 (0.020)	0.760 (0.020)
	60	0.077 (0.008)	0.722 (0.026)	0.778 (0.021)	0.694 (0.032)	0.776 (0.024)	0.766 (0.023)
	100	0.078 (0.007)	0.716 (0.028)	0.774 (0.021)	0.686 (0.034)	0.773 (0.022)	0.763 (0.023)
	240	0.077 (0.007)	0.717 (0.026)	0.777 (0.020)	0.685 (0.032)	0.774 (0.021)	0.764 (0.021)
	300	0.077 (0.007)	0.718 (0.028)	0.775 (0.021)	0.689 (0.033)	0.773 (0.022)	0.764 (0.022)
	720	0.077 (0.007)	0.720 (0.029)	0.777 (0.022)	0.690 (0.034)	0.776 (0.023)	0.766 (0.023)
m=6	6	0.079 (0.005)	0.715 (0.013)	0.768 (0.013)	0.690 (0.022)	0.766 (0.015)	0.757 (0.014)
	18	0.080 (0.006)	0.723 (0.020)	0.769 (0.012)	0.697 (0.030)	0.768 (0.015)	0.759 (0.016)
	30	0.077 (0.005)	0.735 (0.021)	0.777 (0.016)	0.709 (0.025)	0.778 (0.016)	0.768 (0.015)
	60	0.077 (0.008)	0.731 (0.032)	0.776 (0.025)	0.705 (0.038)	0.776 (0.025)	0.766 (0.025)
	100	**0.076 (0.006)**	0.735 (0.024)	**0.782 (0.017)**	0.709 (0.030)	**0.782 (0.019)**	**0.771 (0.018)**
	240	**0.076 (0.007)**	**0.736 (0.025)**	0.781 (0.021)	**0.712 (0.028)**	0.780 (0.023)	0.769 (0.022)
	300	**0.076 (0.006)**	0.731 (0.025)	0.778 (0.019)	0.706 (0.030)	0.778 (0.020)	0.768 (0.020)
	720	**0.076 (0.007)**	0.735 (0.025)	0.781 (0.021)	0.710 (0.030)	0.781 (0.022)	0.770 (0.022)

SCENE (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	6	0.135 (0.004)	0.443 (0.023)	0.576 (0.018)	0.385 (0.024)	0.569 (0.020)	0.555 (0.017)
	18	0.131 (0.009)	0.465 (0.014)	0.602 (0.020)	0.396 (0.024)	0.587 (0.019)	0.579 (0.023)
	30	0.130 (0.005)	0.473 (0.011)	0.611 (0.016)	0.400 (0.015)	0.598 (0.017)	0.587 (0.017)
m=3	6	0.120 (0.009)	0.496 (0.030)	0.621 (0.029)	0.446 (0.027)	0.610 (0.029)	0.605 (0.029)
	18	0.119 (0.011)	0.498 (0.027)	0.628 (0.027)	0.448 (0.035)	0.617 (0.030)	0.609 (0.033)
	30	0.117 (0.008)	0.502 (0.026)	0.627 (0.027)	0.457 (0.028)	0.616 (0.029)	0.609 (0.028)
	60	0.115 (0.010)	0.503 (0.019)	0.632 (0.026)	0.461 (0.023)	0.624 (0.028)	0.615 (0.027)
	100	0.117 (0.010)	0.498 (0.023)	0.627 (0.027)	0.453 (0.025)	0.618 (0.029)	0.609 (0.030)
m=4	6	0.118 (0.009)	0.503 (0.016)	0.616 (0.016)	0.471 (0.028)	0.611 (0.015)	0.603 (0.020)
	18	0.110 (0.007)	0.522 (0.026)	0.641 (0.025)	0.493 (0.026)	0.639 (0.029)	0.627 (0.026)
	30	0.108 (0.008)	0.515 (0.026)	0.644 (0.023)	0.485 (0.030)	0.638 (0.027)	0.629 (0.027)
	60	0.107 (0.008)	0.523 (0.030)	0.647 (0.023)	0.494 (0.032)	0.642 (0.028)	0.634 (0.027)
	100	0.108 (0.009)	0.519 (0.022)	0.644 (0.028)	0.488 (0.024)	0.640 (0.031)	0.631 (0.029)
	240	0.109 (0.009)	0.513 (0.028)	0.638 (0.029)	0.484 (0.031)	0.633 (0.032)	0.624 (0.031)
	300	0.108 (0.008)	0.517 (0.026)	0.642 (0.028)	0.487 (0.027)	0.637 (0.031)	0.628 (0.029)
m=5	6	0.111 (0.010)	0.513 (0.028)	0.633 (0.028)	0.485 (0.035)	0.626 (0.030)	0.619 (0.031)
	18	0.110 (0.009)	0.514 (0.026)	0.638 (0.026)	0.482 (0.027)	0.632 (0.030)	0.624 (0.029)
	30	0.106 (0.006)	0.524 (0.025)	0.649 (0.023)	0.495 (0.026)	0.645 (0.025)	0.636 (0.024)
	60	0.110 (0.011)	0.520 (0.026)	0.642 (0.030)	0.487 (0.033)	0.637 (0.034)	0.628 (0.032)
	100	0.112 (0.005)	0.515 (0.013)	0.639 (0.021)	0.480 (0.012)	0.635 (0.024)	0.622 (0.019)
	240	0.111 (0.005)	0.508 (0.016	0.633 (0.020	0.476 (0.019)	0.628 (0.022)	0.620 (0.020
	300	0.111 (0.010)	0.514 (0.027)	0.639 (0.028)	0.481 (0.027)	0.637 (0.031)	0.625 (0.032)
	720	0.108 (0.011)	0.524 (0.029)	0.645 (0.029)	0.493 (0.035)	0.642 (0.033)	0.632 (0.029)
m=6	6	0.109 (0.007)	0.495 (0.035)	0.622 (0.034)	0.476 (0.028)	0.621 (0.033)	0.611 (0.034)
	18	0.103 (0.005)	0.530 (0.017)	0.650 (0.017)	0.513 (0.014)	0.647 (0.020)	0.640 (0.020)
	30	0.104 (0.011)	0.525 (0.036)	0.648 (0.036)	0.509 (0.033)	0.643 (0.039)	0.635 (0.040)
	60	0.101 (0.007)	0.528 (0.022)	0.654 (0.021)	0.513 (0.022)	0.651 (0.024)	0.643 (0.024)
	100	0.101 (0.006)	0.527 (0.024)	0.654 (0.026)	0.511 (0.021)	0.652 (0.027)	0.642 (0.025)
	240	0.101 (0.006)	0.530 (0.023)	0.655 (0.025)	0.514 (0.018)	0.653 (0.024)	0.644 (0.024)
	300	**0.099 (0.005**)	**0.534 (0.021)**	**0.662 (0.021)**	**0.517 (0.018)**	**0.659 (0.021**)	**0.651 (0.021)**
	720	0.100 (0.005)	0.532 (0.019)	0.659 (0.021)	0.516 (0.014)	0.656 (0.020)	0.647 (0.020)

YEAST (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	20	0.190 (0.005)	0.527 (0.016)	0.658 (0.015)	0.202 (0.026)	0.498 (0.034)	0.656 (0.014)
	40	0.188 (0.007)	0.529 (0.015)	0.659 (0.014)	0.205 (0.022)	0.495 (0.024)	0.658 (0.014)
	60	0.188 (0.008)	0.530 (0.018)	0.660 (0.017)	0.204 (0.027)	0.488 (0.035)	0.659 (0.016)
	80	0.188 (0.007)	0.527 (0.016)	0.658 (0.016)	0.200 (0.024)	0.494 (0.026)	0.658 (0.016)
	100	0.188 (0.008)	0.527 (0.017)	0.659 (0.016)	0.199 (0.024)	0.488 (0.035)	0.658 (0.016)
m=3	20	0.191 (0.006)	0.528 (0.015)	0.658 (0.014)	0.203 (0.025)	0.483 (0.025)	0.655 (0.013)
	40	0.187 (0.007)	0.531 (0.019)	0.662 (0.017)	0.207 (0.026)	0.498 (0.034)	0.660 (0.017)
	60	0.185 (0.009)	0.535 (0.020)	0.665 (0.016)	0.218 (0.031)	0.509 (0.040)	0.664 (0.018)
	80	0.185 (0.010)	0.538 (0.022)	0.666 (0.017)	0.215 (0.038)	0.516 (0.014)	**0.667 (0.020)**
	100	0.186 (0.009)	0.534 (0.019)	0.662 (0.016)	0.213 (0.033)	0.503 (0.027)	0.663 (0.018)
	200	**0.184 (0.009)**	0.538 (0.016)	**0.667 (0.013)**	0.213 (0.037)	0.486 (0.031)	**0.667 (0.015)**
	300	**0.184 (0.009)**	0.538 (0.017)	**0.667 (0.014)**	0.216 (0.037)	0.490 (0.027)	**0.667 (0.016)**
	500	0.188 (0.008)	0.529 (0.017)	0.660 (0.016)	0.204 (0.026)	0.488 (0.037)	0.658 (0.016)
	1000	0.188 (0.008)	0.529 (0.017)	0.659 (0.017)	0.204 (0.026)	0.487 (0.035)	0.658 (0.016)
m=4	20	0.188 (0.008)	0.532 (0.019)	0.661 (0.017)	0.210 (0.034)	0.489 (0.033)	0.660 (0.017)
	40	0.189 (0.006)	0.530 (0.015)	0.660 (0.015)	0.207 (0.025)	0.495 (0.041)	0.659 (0.013)
	60	0.185 (0.007)	0.535 (0.016)	0.665 (0.013)	0.221 (0.030)	0.508 (0.040)	0.664 (0.015)
	80	0.186 (0.009)	0.534 (0.019)	0.662 (0.017)	0.218 (0.030)	0.516 (0.022)	0.662 (0.017)
	100	0.187 (0.009)	0.531 (0.019)	0.660 (0.017)	0.209 (0.029)	0.513 (0.028)	0.661 (0.018)
	200	0.186 (0.009)	0.537 (0.015)	0.666 (0.013)	0.216 (0.035)	0.491 (0.045)	0.665 (0.014)
	300	**0.184 (0.009)**	**0.539 (0.016)**	**0.667 (0.014)**	0.219 (0.034)	0.482 (0.028)	**0.667 (0.014)**
	500	0.188 (0.007)	0.529 (0.016)	0.660 (0.016)	0.204 (0.029)	0.502 (0.042)	0.658 (0.015)
	1000	0.188 (0.008)	0.529 (0.017)	0.660 (0.016)	0.207 (0.029)	0.495 (0.047)	0.659 (0.016)
m=5	20	0.189 (0.008)	0.529 (0.021)	0.659 (0.017)	0.205 (0.030)	0.508 (0.042)	0.658 (0.018)
	40	0.189 (0.007)	0.526 (0.019)	0.657 (0.017)	0.206 (0.025)	0.495 (0.037)	0.655 (0.017)
	60	0.185 (0.008)	0.535 (0.016)	0.665 (0.012)	0.220 (0.030)	0.518 (0.049)	0.663 (0.015)
	80	0.187 (0.009)	0.533 (0.019)	0.661 (0.017)	0.217 (0.031)	0.513 (0.022)	0.661 (0.018)
	100	0.186 (0.010)	0.533 (0.021)	0.661 (0.017)	0.220 (0.038)	0.536 (0.036)	0.661 (0.019)
	200	0.192 (0.019)	0.509 (0.067)	0.643 (0.052)	0.179 (0.102)	0.502 (0.047)	0.640 (0.061)
	300	**0.184 (0.008)**	0.537 (0.016)	0.666 (0.014)	0.218 (0.037)	0.486 (0.042)	0.665 (0.014)
	500	0.188 (0.007)	0.529 (0.017)	0.659 (0.017)	0.209 (0.030)	0.499 (0.039)	0.658 (0.015)
	1000	0.188 (0.008)	0.529 (0.017)	0.660 (0.016)	0.208 (0.029)	0.501 (0.042)	0.658 (0.016)
m=6	20	0.190 (0.004)	0.520 (0.019)	0.653 (0.021)	0.180 (0.026)	0.510 (0.039)	0.650 (0.016)
	40	0.189 (0.007)	0.527 (0.017)	0.657 (0.016)	0.211 (0.028)	0.502 (0.062)	0.656 (0.015)
	60	0.185 (0.010)	0.535 (0.021)	0.665 (0.015)	0.222 (0.037)	0.519 (0.042)	0.663 (0.019)
	80	0.186 (0.009)	0.533 (0.019)	0.661 (0.014)	0.213 (0.040)	**0.542 (0.044)**	0.661 (0.017)
	100	0.187 (0.009)	0.528 (0.019)	0.656 (0.017)	0.209 (0.030)	0.508 (0.030)	0.657 (0.017)
	200	0.185 (0.011)	0.532 (0.026)	0.661 (0.022)	0.216 (0.044)	0.514 (0.045)	0.661 (0.022)
	300	**0.184 (0.008)**	0.538 (0.014)	0.538 (0.011)	**0.223 (0.035)**	0.518 (0.054)	0.666 (0.012)
	500	0.189 (0.008)	0.527 (0.017)	0.657 (0.016)	0.209 (0.029)	0.519 (0.054)	0.656 (0.015)
	1000	0.188 (0.007)	0.528 (0.017)	0.658 (0.015)	0.209 (0.030)	0.503 (0.044)	0.656 (0.015)

YEAST (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	20	0.230 (0.010)	0.446 (0.025)	0.594 (0.023)	0.081 (0.015)	0.436 (0.020)	0.590 (0.020)
	40	0.222 (0.005)	0.453 (0.014)	0.602 (0.013)	0.079 (0.018)	0.453 (0.031)	0.599 (0.011)
	60	0.224 (0.007)	0.448 (0.016)	0.599 (0.014)	0.077 (0.022)	0.459 (0.017)	0.595 (0.013)
	80	0.225 (0.006)	0.448 (0.016)	0.598 (0.016)	0.073 (0.018)	0.447 (0.020)	0.595 (0.011)
	100	0.227 (0.005)	0.444 (0.014)	0.594 (0.012)	0.071 (0.016)	0.454 (0.031)	0.591 (0.011)
m=3	20	0.215 (0.005)	0.457 (0.020)	0.604 (0.017)	0.096 (0.022)	0.440 (0.016)	0.603 (0.016)
	40	0.214 (0.006)	0.463 (0.018)	0.608 (0.014)	0.098 (0.020)	0.462 (0.017)	0.607 (0.015)
	60	0.212 (0.007)	0.469 (0.019)	0.615 (0.018)	0.101 (0.023)	0.484 (0.040)	0.613 (0.016)
	80	0.214 (0.006)	0.458 (0.012)	0.605 (0.012)	0.089 (0.016)	0.467 (0.030)	0.604 (0.011)
	100	0.212 (0.006)	0.466 (0.016)	0.612 (0.014)	0.097 (0.017)	0.475 (0.014)	0.611 (0.014)
	200	0.212 (0.008)	0.465 (0.018)	0.612 (0.016)	0.095 (0.026)	0.470 (0.025)	0.610 (0.016)
	300	0.212 (0.006)	0.467 (0.014)	0.611 (0.011)	0.097 (0.028)	**0.490 (0.032)**	0.611 (0.011)
	500	0.212 (0.006)	0.465 (0.016)	0.611 (0.012)	0.093 (0.021)	0.485 (0.025)	0.609 (0.014)
	1000	0.211 (0.007)	0.466 (0.015)	0.612 (0.013)	0.098 (0.021)	0.488 (0.027)	0.610 (0.013)
m=4	20	0.212 (0.011)	0.470 (0.023)	0.616 (0.021)	0.101 (0.017)	0.468 (0.038)	0.611 (0.020)
	40	0.211 (0.005)	0.463 (0.018)	0.603 (0.016)	0.110 (0.019)	0.471 (0.038)	0.606 (0.014)
	60	0.208 (0.008)	0.470 (0.018)	0.613 (0.016)	0.106 (0.023)	0.476 (0.029)	0.611 (0.017)
	80	0.205 (0.008)	0.473 (0.023)	0.614 (0.018)	**0.121 (0.029)**	0.483 (0.035)	0.615 (0.019)
	100	0.208 (0.006)	0.465 (0.017)	0.607 (0.016)	0.106 (0.020)	0.473 (0.036)	0.606 (0.015)
	200	0.206 (0.006)	0.475 (0.017)	0.617 (0.015)	0.113 (0.024)	0.485 (0.032)	**0.617 (0.013)**
	300	0.205 (0.009)	0.474 (0.020)	0.616 (0.014)	0.114 (0.037)	0.482 (0.012)	0.615 (0.017)
	500	0.205 (0.008)	**0.476 (0.015)**	**0.618 (0.011)**	0.119 (0.026)	0.476 (0.026)	**0.617 (0.013)**
	1000	0.206 (0.009)	0.472 (0.018)	0.613 (0.014)	0.116 (0.028)	0.488 (0.027)	0.614 (0.016)
m=5	20	0.212 (0.011)	0.470 (0.023)	0.616 (0.021)	0.101 (0.017)	0.468 (0.038)	0.611 (0.020)
	40	0.211 (0.005)	0.463 (0.018)	0.603 (0.016)	0.110 (0.019)	0.471 (0.038)	0.606 (0.014)
	60	0.208 (0.008)	0.470 (0.018)	0.613 (0.016)	0.106 (0.023)	0.476 (0.029)	0.611 (0.017)
	80	0.205 (0.008)	0.473 (0.023)	0.614 (0.018)	**0.121 (0.029)**	0.483 (0.035)	0.615 (0.019)
	100	0.208 (0.006)	0.465 (0.017)	0.607 (0.016)	0.106 (0.020)	0.473 (0.036)	0.606 (0.015)
	200	0.206 (0.006)	0.475 (0.017)	0.617 (0.015)	0.113 (0.024)	0.485 (0.032)	**0.617 (0.013)**
	300	0.205 (0.009)	0.474 (0.020)	0.616 (0.014)	0.114 (0.037)	0.482 (0.012)	0.615 (0.017)
	500	0.205 (0.008)	**0.476 (0.015)**	**0.618 (0.011)**	0.119 (0.026)	0.476 (0.026)	**0.617 (0.013)**
	1000	0.206 (0.009)	0.472 (0.018)	0.613 (0.014)	0.116 (0.028)	0.488 (0.027)	0.614 (0.016)
m=6	20	0.210 (0.010)	0.451 (0.021)	0.597 (0.020)	0.096 (0.026)	0.468 (0.032)	0.597 (0.021)
	40	0.207 (0.009)	0.457 (0.020)	0.601 (0.016)	0.100 (0.017)	0.476 (0.066)	0.600 (0.018)
	60	0.204 (0.008)	0.466 (0.019)	0.607 (0.018)	0.114 (0.026)	0.459 (0.030)	0.609 (0.016)
	80	0.203 (0.008)	0.460 (0.017)	0.600 (0.014)	0.111 (0.023)	0.459 (0.059)	0.604 (0.016)
	100	0.201 (0.007)	0.466 (0.018)	0.607 (0.016)	0.113 (0.017)	0.461 (0.026)	0.610 (0.016)
	200	0.202 (0.008)	0.466 (0.020)	0.607 (0.014)	0.113 (0.024)	0.473 (0.046)	0.608 (0.016)
	300	**0.200 (0.009)**	0.470 (0.016)	0.609 (0.011)	0.118 (0.025)	0.471 (0.048)	0.612 (0.014)
	500	0.202 (0.009)	0.465 (0.020)	0.605 (0.014)	0.115 (0.026)	0.463 (0.051)	0.608 (0.017)
	1000	0.201 (0.008)	0.469 (0.016)	0.608 (0.013)	0.119 (0.021)	0.467 (0.047)	0.611 (0.015)

MEDICAL (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	60	0.012 (0.002)	0.712 (0.038)	0.802 (0.036)	0.614 (0.043)	0.828 (0.042)	0.783 (0.037)
	80	0.012 (0.002)	0.712 (0.050)	0.802 (0.041)	0.617 (0.059)	0.827 (0.029)	0.785 (0.044)
	100	0.012 (0.002)	0.714 (0.044)	0.803 (0.037)	0.623 (0.051)	0.832 (0.031)	**0.787 (0.039)**
	200	0.012 (0.002)	0.710 (0.046)	0.801 (0.038)	0.614 (0.053)	0.831 (0.032)	0.783 (0.040)
	300	0.012 (0.002)	0.713 (0.047)	0.803 (0.038)	0.616 (0.055)	0.832 (0.033)	0.786 (0.040)
	500	0.012 (0.002)	0.711 (0.044)	0.802 (0.038)	0.616 (0.050)	0.829 (0.034)	0.786 (0.040)
	1000	0.012 (0.002)	0.713 (0.047)	0.803 (0.039)	0.617 (0.054)	0.829 (0.035)	0.786 (0.038)
m=3	60	0.012 (0.003)	0.695 (0.066)	0.794 (0.052)	0.589 (0.088)	0.828 (0.032)	0.774 (0.058)
	80	0.012 (0.003)	0.698 (0.072)	0.793 (0.057)	0.595 (0.092)	0.834 (0.031)	0.774 (0.062)
	100	0.012 (0.002)	0.714 (0.044)	0.803 (0.037)	0.619 (0.053)	0.837 (0.028)	**0.787 (0.039)**
	200	0.012 (0.002)	**0.715 (0.047)**	**0.804 (0.039)**	0.620 (0.051)	0.835 (0.029)	**0.787 (0.040)**
	300	0.012 (0.002)	0.712 (0.044)	0.802 (0.037)	0.617 (0.049)	0.826 (0.037)	0.786 (0.040)
	500	0.012 (0.002)	0.709 (0.046)	0.801 (0.038)	0.612 (0.053)	0.831 (0.033)	0.784 (0.041)
	1000	0.012 (0.002)	0.713 (0.049)	0.803 (0.040)	0.617 (0.058)	0.827 (0.036)	0.786 (0.041)
m=4	60	0.012 (0.002)	0.697 (0.042)	0.795 (0.032)	0.603 (0.053)	0.831 (0.035)	0.780 (0.033)
	80	0.012 (0.002)	0.694 (0.030)	0.787 (0.030)	0.606 (0.031)	0.841 (0.024)	0.773 (0.032)
	100	0.012 (0.002)	0.711 (0.041)	0.802 (0.036)	0.616 (0.048)	0.822 (0.031)	0.785 (0.036)
	200	0.012 (0.002)	0.712 (0.046)	0.802 (0.040)	0.617 (0.052)	**0.845 (0.026)**	0.784 (0.041)
	300	0.012 (0.002)	0.714 (0.043)	0.803 (0.037)	0.622 (0.048)	0.831 (0.032)	**0.787 (0.039)**
	500	0.012 (0.002)	0.714 (0.043)	**0.804 (0.036)**	0.622 (0.051)	0.837 (0.026)	**0.787 (0.039)**
	1000	0.012 (0.002)	0.711 (0.044)	0.801 (0.038)	0.616 (0.048)	0.830 (0.031)	0.784 (0.039)
m=5	60	0.012 (0.002)	0.697 (0.042)	0.795 (0.032)	0.603 (0.053)	0.831 (0.035)	0.780 (0.033)
	80	0.012 (0.002)	0.694 (0.030)	0.787 (0.030)	0.606 (0.031)	0.841 (0.024)	0.773 (0.032)
	100	0.012 (0.002)	0.711 (0.041)	0.802 (0.036)	0.616 (0.048)	0.822 (0.031)	0.785 (0.036)
	200	0.012 (0.002)	0.712 (0.046)	0.802 (0.040)	0.617 (0.052)	**0.845 (0.026)**	0.784 (0.041)
	300	0.012 (0.002)	0.714 (0.043)	0.803 (0.037)	0.622 (0.048)	0.831 (0.032)	**0.787 (0.039)**
	500	0.012 (0.002)	0.714 (0.043)	**0.804 (0.036)**	0.622 (0.051)	0.837 (0.026)	**0.787 (0.039)**
	1000	0.012 (0.002)	0.711 (0.044)	0.801 (0.038)	0.616 (0.048)	0.830 (0.031)	0.784 (0.039)
m=6	60	0.012 (0.003)	0.698 (0.059)	0.792 (0.048)	0.612 (0.074)	0.840 (0.046)	0.779 (0.050)
	80	0.012 (0.002)	0.706 (0.048)	0.800 (0.039)	0.614 (0.064)	0.825 (0.039)	0.783 (0.043)
	100	0.012 (0.002)	0.694 (0.038)	0.788 (0.036)	0.609 (0.044)	0.817 (0.022)	0.773 (0.038)
	200	0.012 (0.002)	0.705 (0.041)	0.797 (0.036)	0.612 (0.052)	0.829 (0.025)	0.780 (0.039)
	300	0.012 (0.002)	0.698 (0.049)	0.794 (0.036)	0.609 (0.065)	0.836 (0.026)	0.781 (0.041)
	500	**0.011 (0.002)**	**0.715 (0.047)**	0.802 (0.039)	**0.627 (0.055)**	0.838 (0.026)	**0.787 (0.042)**
	1000	0.012 (0.002)	0.711 (0.044)	0.800 (0.038)	0.622 (0.053)	0.839 (0.033)	0.785 (0.042)

MEDICAL (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	60	0.011 (0.002)	0.733 (0.068)	0.815 (0.050)	0.650 (0.061)	0.859 (0.030)	0.796 (0.045)
	80	0.011 (0.002)	0.732 (0.067)	**0.817 (0.045)**	0.650 (0.057)	0.860 (0.029)	0.798 (0.041)
	100	0.011 (0.002)	0.732 (0.069)	0.815 (0.051)	0.648 (0.063)	0.859 (0.030)	0.797 (0.045)
	200	0.011 (0.002)	0.730 (0.066)	0.816 (0.047)	0.647 (0.062)	0.863 (0.027)	0.797 (0.044)
	300	0.011 (0.002)	0.731 (0.068)	0.814 (0.050)	0.647 (0.062)	0.859 (0.030)	0.795 (0.045)
	500	0.011 (0.002)	0.731 (0.068)	0.814 (0.050)	0.647 (0.062)	0.859 (0.030)	0.795 (0.045)
	1000	0.011 (0.002)	0.731 (0.068)	0.814 (0.050)	0.647 (0.062)	0.859 (0.030)	0.795 (0.045)
m=3	60	0.011 (0.002)	0.707 (0.058)	0.799 (0.041)	0.628 (0.057)	0.856 (0.030)	0.783 (0.041)
	80	**0.010 (0.002)**	0.722 (0.071)	0.814 (0.050)	0.644 (0.073)	0.872 (0.032)	0.797 (0.046)
	100	0.011 (0.002)	0.730 (0.068)	0.814 (0.051)	0.650 (0.068)	0.869 (0.028)	0.796 (0.048)
	200	0.011 (0.002)	**0.734 (0.068)**	**0.817 (0.049)**	0.652 (0.062)	0.859 (0.030)	0.798 (0.044)
	300	0.011 (0.002)	**0.734 (0.068)**	**0.817 (0.048)**	0.652 (0.062)	0.859 (0.030)	**0.799 (0.043)**
	500	0.011 (0.002)	0.731 (0.068)	0.814 (0.050)	0.647 (0.062)	0.859 (0.030)	0.795 (0.045)
	1000	0.011 (0.002)	0.731 (0.068)	0.814 (0.050)	0.647 (0.062)	0.859 (0.030)	0.795 (0.045)
m=4	60	0.012 (0.002)	0.707 (0.078)	0.795 (0.059)	0.625 (0.076)	0.857 (0.030)	0.775 (0.056)
	80	0.011 (0.002)	0.722 (0.061)	0.814 (0.046)	0.641 (0.056)	0.864 (0.032)	0.795 (0.042)
	100	0.011 (0.002)	0.729 (0.070)	0.812 (0.051)	0.647 (0.066)	0.854 (0.030)	0.794 (0.047)
	200	0.011 (0.002)	0.733 (0.068)	**0.817 (0.048)**	0.652 (0.062)	0.862 (0.033)	0.798 (0.043)
	300	0.011 (0.002)	0.733 (0.070)	**0.817 (0.049)**	0.652 (0.068)	0.867 (0.029)	0.798 (0.045)
	500	**0.010 (0.002)**	0.730 (0.064)	**0.817 (0.046)**	0.650 (0.059)	0.862 (0.030)	**0.799 (0.042)**
	1000	0.011 (0.002)	0.733 (0.069)	**0.817 (0.048)**	0.650 (0.064)	0.859 (0.031)	0.798 (0.043)
m=5	60	0.012 (0.002)	0.707 (0.078)	0.795 (0.059)	0.625 (0.076)	0.857 (0.030)	0.775 (0.056)
	80	0.011 (0.002)	0.722 (0.061)	0.814 (0.046)	0.641 (0.056)	0.864 (0.032)	0.795 (0.042)
	100	0.011 (0.002)	0.729 (0.070)	0.812 (0.051)	0.647 (0.066)	0.854 (0.030)	0.794 (0.047)
	200	0.011 (0.002)	0.733 (0.068)	**0.817 (0.048)**	0.652 (0.062)	0.862 (0.033)	0.798 (0.043)
	300	0.011 (0.002)	0.733 (0.070)	**0.817 (0.049)**	0.652 (0.068)	0.867 (0.029)	0.798 (0.045)
	500	**0.010 (0.002)**	0.730 (0.064)	**0.817 (0.046)**	0.650 (0.059)	0.862 (0.030)	**0.799 (0.042)**
	1000	0.011 (0.002)	0.733 (0.069)	**0.817 (0.048)**	0.650 (0.064)	0.859 (0.031)	0.798 (0.043)
m=6	60	0.012 (0.002)	0.684 (0.096)	0.782 (0.064)	0.606 (0.103)	0.858 (0.035)	0.765 (0.065)
	80	0.012 (0.002)	0.667 (0.087)	0.774 (0.068)	0.594 (0.078)	0.870 (0.044)	0.760 (0.064)
	100	0.012 (0.003)	0.630 (0.096)	0.750 (0.074)	0.558 (0.096)	0.859 (0.049)	0.745 (0.071)
	200	**0.010 (0.002)**	0.724 (0.063)	0.812 (0.046)	**0.653 (0.061)**	**0.878 (0.030)**	0.797 (0.046)
	300	0.011 (0.002)	0.716 (0.057)	0.808 (0.041)	0.641 (0.053)	0.874 (0.030)	0.794 (0.037)
	500	0.011 (0.002)	0.704 (0.067)	0.801 (0.048)	0.628 (0.070)	0.876 (0.029)	0.787 (0.049)
	1000	0.011 (0.002)	0.703 (0.048)	0.800 (0.034)	0.633 (0.048)	0.873 (0.035)	0.788 (0.036)

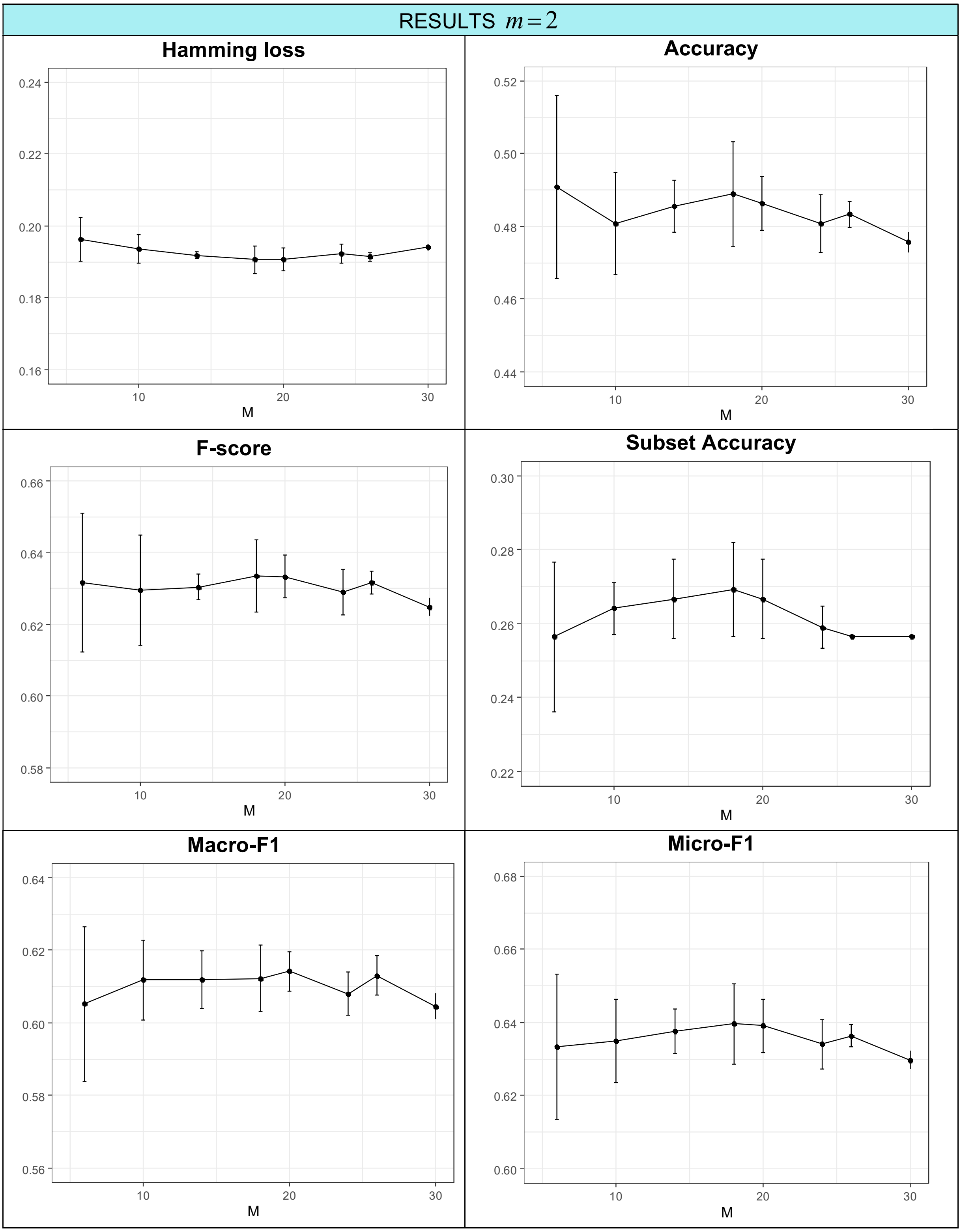
ENRON (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	60	**0.047 (0.001)**	0.418 (0.012)	0.578 (0.012)	0.120 (0.026)	0.471 (0.052)	0.551 (0.011)
	80	**0.047 (0.001)**	0.415 (0.011)	0.576 (0.011)	0.115 (0.027)	0.457 (0.049)	0.550 (0.011)
	100	**0.047 (0.001)**	0.418 (0.010)	0.578 (0.010)	0.116 (0.024)	0.456 (0.044)	0.552 (0.011)
	200	**0.047 (0.001)**	0.419 (0.011)	0.578 (0.010)	0.118 (0.027)	0.463 (0.051)	0.552 (0.009)
	300	**0.047 (0.001)**	0.417 (0.010)	0.577 (0.010)	0.115 (0.027)	0.450 (0.049)	0.551 (0.010)
	500	**0.047 (0.001)**	0.417 (0.010)	0.577 (0.011)	0.115 (0.025)	0.460 (0.041)	0.551 (0.010)
	1000	**0.047 (0.001)**	0.417 (0.010)	0.577 (0.010)	0.115 (0.027)	0.462 (0.054)	0.551 (0.009)
m=3	60	**0.047 (0.001)**	0.407 (0.018)	0.566 (0.023)	0.112 (0.022)	0.462 (0.048)	0.538 (0.025)
	80	**0.047 (0.001)**	0.416 (0.011)	0.576 (0.011)	0.115 (0.026)	0.451 (0.053)	0.550 (0.010)
	100	**0.047 (0.001)**	0.419 (0.009)	0.579 (0.010)	**0.121 (0.025)**	0.442 (0.041)	**0.553 (0.008)**
	200	**0.047 (0.001)**	0.417 (0.011)	0.577 (0.011)	0.115 (0.025)	0.465 (0.053)	0.551 (0.010)
	300	**0.047 (0.001)**	0.418 (0.013)	0.578 (0.012)	0.116 (0.025)	0.459 (0.038)	0.552 (0.011)
	500	**0.047 (0.001)**	0.417 (0.010)	0.577 (0.010)	0.115 (0.026)	0.465 (0.052)	0.551 (0.010)
	1000	**0.047 (0.001)**	0.417 (0.010)	0.577 (0.010)	0.115 (0.025)	0.466 (0.051)	0.551 (0.009)
m=4	60	**0.047 (0.001)**	**0.420 (0.012)**	**0.580 (0.011)**	0.117 (0.028)	0.466 (0.056)	0.552 (0.009)
	80	**0.047 (0.001)**	0.417 (0.015)	0.577 (0.014)	0.118 (0.030)	0.456 (0.045)	0.551 (0.010)
	100	**0.047 (0.001)**	0.417 (0.008)	0.577 (0.010)	0.115 (0.026)	0.467 (0.052)	0.551 (0.009)
	200	**0.047 (0.001)**	0.417 (0.012)	0.577 (0.012)	0.116 (0.028)	0.466 (0.047)	0.551 (0.011)
	300	**0.047 (0.001)**	0.419 (0.014)	0.579 (0.013)	0.118 (0.029)	0.455 (0.037)	0.551 (0.011)
	500	**0.047 (0.001)**	0.416 (0.011)	0.577 (0.010)	0.115 (0.028)	0.464 (0.052)	0.550 (0.009)
	1000	**0.047 (0.001)**	0.418 (0.010)	0.578 (0.010)	0.115 (0.025)	0.467 (0.052)	0.552 (0.010)
m=5	60	**0.047 (0.001)**	**0.420 (0.012)**	**0.580 (0.011)**	0.117 (0.028)	0.466 (0.056)	0.552 (0.009)
	80	**0.047 (0.001)**	0.417 (0.015)	0.577 (0.014)	0.118 (0.030)	0.456 (0.045)	0.551 (0.010)
	100	**0.047 (0.001)**	0.417 (0.008)	0.577 (0.010)	0.115 (0.026)	0.467 (0.052)	0.551 (0.009)
	200	**0.047 (0.001)**	0.417 (0.012)	0.577 (0.012)	0.116 (0.028)	0.466 (0.047)	0.551 (0.011)
	300	**0.047 (0.001)**	0.419 (0.014)	0.579 (0.013)	0.118 (0.029)	0.455 (0.037)	0.551 (0.011)
	500	**0.047 (0.001)**	0.416 (0.011)	0.577 (0.010)	0.115 (0.028)	0.464 (0.052)	0.550 (0.009)
	1000	**0.047 (0.001)**	0.418 (0.010)	0.578 (0.010)	0.115 (0.025)	0.467 (0.052)	0.552 (0.010)
m=6	60	**0.047 (0.001)**	0.415 (0.010)	0.573 (0.007)	0.116 (0.032)	0.470 (0.044)	0.548 (0.008)
	80	**0.047 (0.001)**	0.415 (0.011)	0.576 (0.012)	0.115 (0.026)	0.469 (0.041)	0.550 (0.011)
	100	**0.047 (0.001)**	0.416 (0.008)	0.577 (0.009)	0.114 (0.024)	0.463 (0.054)	0.551 (0.008)
	200	**0.047 (0.001)**	0.417 (0.012)	0.577 (0.013)	0.116 (0.026)	0.473 (0.047)	0.551 (0.012)
	300	**0.047 (0.001)**	0.418 (0.012)	0.577 (0.013)	0.116 (0.026)	**0.480 (0.043)**	0.551 (0.011)
	500	**0.047 (0.001)**	0.417 (0.012)	0.577 (0.011)	0.115 (0.027)	0.464 (0.052)	0.551 (0.010)
	1000	**0.047 (0.001)**	0.417 (0.009)	0.577 (0.010)	0.115 (0.025)	0.465 (0.051)	0.551 (0.009)

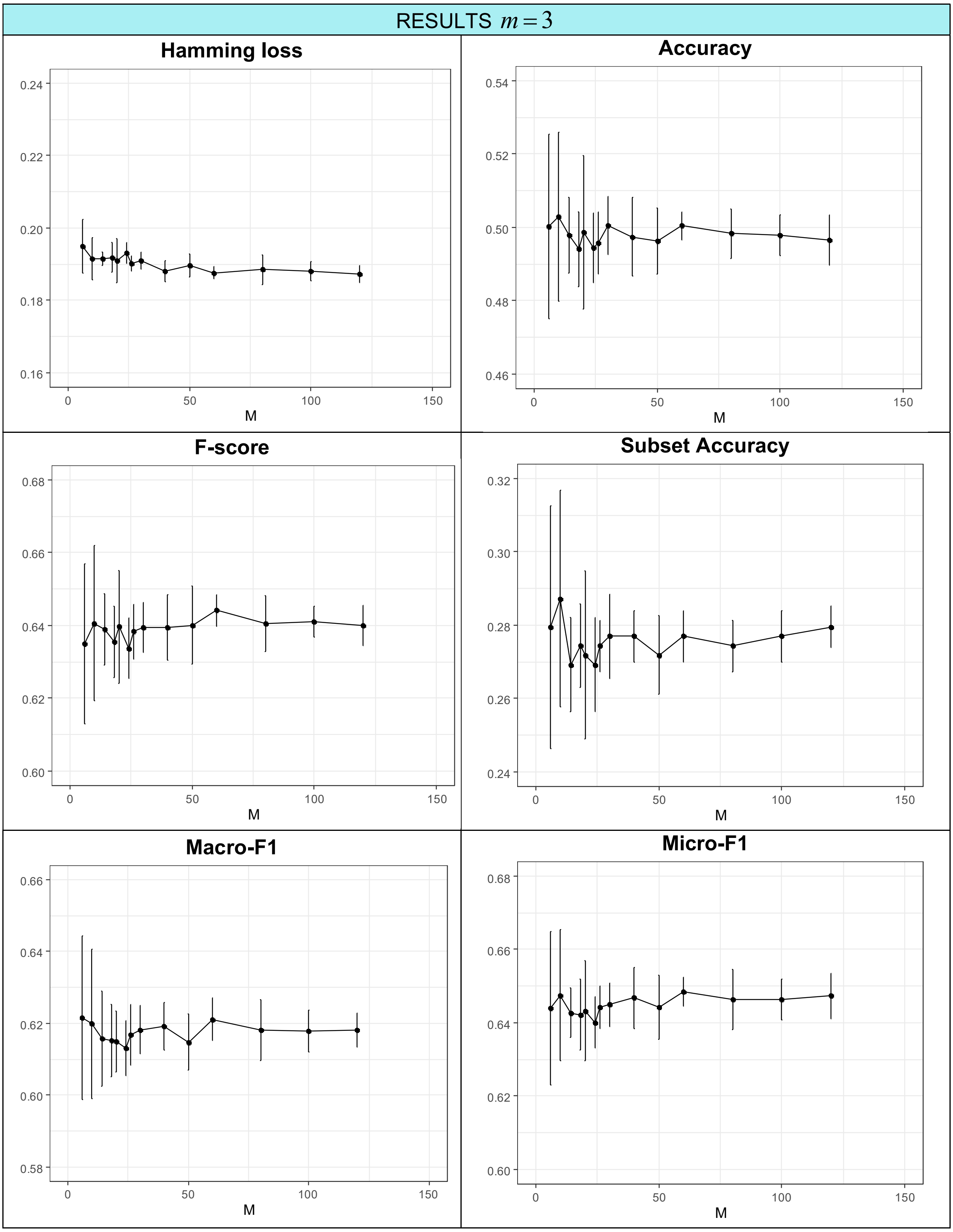
ENRON (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	60	0.051 (0.002)	**0.402 (0.021)**	**0.561 (0.022)**	0.086 (0.013)	0.432 (0.034)	0.546 (0.015)
	80	0.051 (0.002)	**0.402 (0.023)**	0.558 (0.025)	0.089 (0.012)	0.438 (0.026)	0.546 (0.016)
	100	0.051 (0.002)	0.399 (0.024)	0.556 (0.025)	0.085 (0.013)	0.442 (0.027)	0.544 (0.016)
	200	0.051 (0.002)	0.398 (0.026)	0.556 (0.027)	0.085 (0.013)	0.436 (0.023)	0.544 (0.019)
	300	0.051 (0.002)	0.401 (0.024)	0.559 (0.026)	0.085 (0.013)	0.438 (0.028)	0.547 (0.017)
	500	0.051 (0.002)	0.400 (0.024)	0.558 (0.026)	0.085 (0.013)	0.439 (0.024)	0.546 (0.017)
	1000	0.051 (0.002)	0.401 (0.024)	0.559 (0.026)	0.085 (0.013)	0.440 (0.026)	0.546 (0.017)
m=3	60	0.050 (0.002)	0.399 (0.022)	0.556 (0.023)	0.089 (0.014)	0.441 (0.034)	0.542 (0.018)
	80	0.051 (0.003)	0.387 (0.031)	0.544 (0.032)	0.082 (0.015)	0.438 (0.022)	0.530 (0.029)
	100	0.051 (0.003)	0.401 (0.026)	0.558 (0.028)	0.085 (0.015)	0.440 (0.026)	0.546 (0.018)
	200	0.051 (0.002)	0.400 (0.021)	0.558 (0.023)	0.084 (0.012)	0.440 (0.028)	0.547 (0.015)
	300	0.051 (0.002)	0.401 (0.024)	0.559 (0.026)	0.087 (0.014)	0.443 (0.028)	0.547 (0.017)
	500	0.051 (0.002)	0.400 (0.024)	0.558 (0.027)	0.085 (0.013)	0.432 (0.029)	0.546 (0.017)
	1000	0.051 (0.002)	0.401 (0.024)	0.559 (0.026)	0.085 (0.013)	**0.452 (0.022)**	0.547 (0.016)
m=4	60	0.050 (0.002)	0.396 (0.018)	0.554 (0.017)	0.084 (0.013)	0.441 (0.016)	0.541 (0.008)
	80	0.050 (0.002)	0.395 (0.026)	0.552 (0.028)	0.085 (0.017)	0.434 (0.024)	0.540 (0.021)
	100	0.050 (0.001)	0.395 (0.024)	0.551 (0.027)	0.084 (0.013)	0.436 (0.028)	0.541 (0.017)
	200	0.050 (0.002)	0.400 (0.026)	0.558 (0.026)	0.088 (0.016)	0.439 (0.024)	0.546 (0.016)
	300	0.050 (0.002)	0.399 (0.022)	0.557 (0.022)	0.084 (0.012)	0.447 (0.021)	0.544 (0.014)
	500	0.050 (0.002)	0.401 (0.023)	0.559 (0.026)	0.084 (0.012)	0.447 (0.020)	**0.548 (0.017)**
	1000	0.050 (0.002)	**0.402 (0.024)**	0.560 (0.027)	0.086 (0.014)	0.444 (0.025)	**0.548 (0.017)**
m=5	60	0.050 (0.002)	0.396 (0.018)	0.554 (0.017)	0.084 (0.013)	0.441 (0.016)	0.541 (0.008)
	80	0.050 (0.002)	0.395 (0.026)	0.552 (0.028)	0.085 (0.017)	0.434 (0.024)	0.540 (0.021)
	100	0.050 (0.001)	0.395 (0.024)	0.551 (0.027)	0.084 (0.013)	0.436 (0.028)	0.541 (0.017)
	200	0.050 (0.002)	0.400 (0.026)	0.558 (0.026)	0.088 (0.016)	0.439 (0.024)	0.546 (0.016)
	300	0.050 (0.002)	0.399 (0.022)	0.557 (0.022)	0.084 (0.012)	0.447 (0.021)	0.544 (0.014)
	500	0.050 (0.002)	0.401 (0.023)	0.559 (0.026)	0.084 (0.012)	0.447 (0.020)	**0.548 (0.017)**
	1000	0.050 (0.002)	**0.402 (0.024)**	0.560 (0.027)	0.086 (0.014)	0.444 (0.025)	**0.548 (0.017)**
m=6	60	0.050 (0.002)	0.395 (0.024)	0.552 (0.023)	0.087 (0.014)	0.431 (0.011)	0.538 (0.019)
	80	**0.049 (0.001)**	0.396 (0.023)	0.554 (0.023)	0.086 (0.016)	0.442 (0.012)	0.541 (0.015)
	100	**0.049 (0.002)**	0.396 (0.027)	0.551 (0.029)	0.084 (0.019)	0.443 (0.023)	0.540 (0.021)
	200	0.050 (0.002)	0.394 (0.033)	0.552 (0.032)	0.084 (0.017)	0.430 (0.015)	0.541 (0.024)
	300	**0.049 (0.002)**	0.400 (0.021)	0.556 (0.022)	0.085 (0.009)	0.444 (0.017)	0.545 (0.014)
	500	**0.049 (0.002)**	0.399 (0.025)	0.555 (0.025)	**0.090 (0.014)**	0.434 (0.025)	0.545 (0.015)
	1000	0.050 (0.001)	0.396 (0.023)	0.553 (0.023)	0.085 (0.013)	0.438 (0.011)	0.543 (0.015)

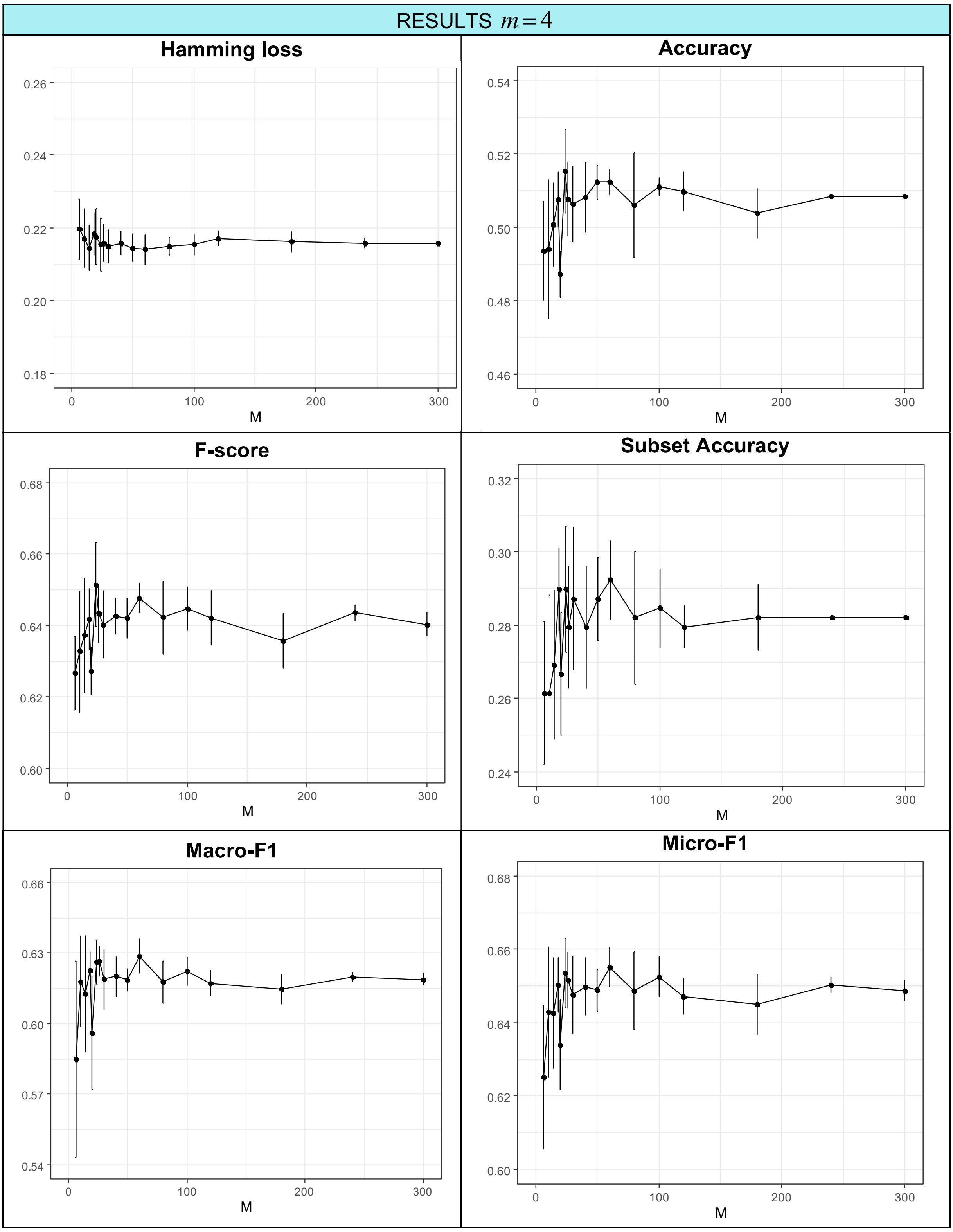
COREL5K (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	500	**0.009 (<0.001)**	**0.054 (0.005)**	**0.104 (0.011)**	0.008 (0.002)	0.256 (0.016)	**0.105 (0.011)**
	1000	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.011)	0.008 (0.002)	0.249 (0.025)	0.103 (0.011)
m=3	500	**0.009 (<0.001)**	0.053 (0.006)	0.102 (0.011)	0.008 (0.002)	0.250 (0.024)	0.103 (0.012)
	1000	**0.009 (<0.001)**	0.053 (0.006)	0.102 (0.011)	**0.009 (0.002)**	0.252 (0.023)	0.103 (0.012)
m=4	500	**0.009 (<0.001)**	0.053 (0.004)	0.102 (0.008)	**0.009 (0.002)**	0.253 (0.018)	0.103 (0.009)
	1000	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.010)	0.008 (0.002)	**0.257 (0.017)**	0.103 (0.011)
m=5	500	**0.009 (<0.001)**	0.053 (0.007)	0.102 (0.014)	**0.009 (0.003)**	0.256 (0.019)	0.104 (0.014)
	1000	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.010)	0.008 (0.002)	0.254 (0.021)	0.104 (0.011)
m=6	500	**0.009 (<0.001)**	0.052 (0.006)	0.101 (0.012)	0.008 (0.002)	0.255 (0.021)	0.102 (0.013)
	1000	**0.009 (<0.001)**	0.052 (0.005)	0.101 (0.010)	0.008 (0.002)	0.252 (0.018)	0.102 (0.011)

COREL5K (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset Accuracy	Macro-F1	Micro-F1
m=2	500	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.184 (0.026)	**0.037 (0.006)**
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.184 (0.026)	**0.037 (0.006)**
m=3	500	**0.009 (0.000)**	0.018 (0.003)	0.036 (0.006)	**0.001 (0.001)**	0.179 (0.028)	0.036 (0.006)
	1000	**0.009 (0.000)**	0.018 (0.004)	0.036 (0.008)	**0.001 (0.001)**	0.176 (0.029)	0.036 (0.008)
m=4	500	**0.009 (0.000)**	0.018 (0.003)	0.036 (0.006)	**0.001 (0.001)**	0.184 (0.026)	0.036 (0.006)
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	**0.186 (0.027)**	**0.037 (0.006)**
m=5	500	**0.009 (0.000)**	0.018 (0.003)	0.036 (0.006)	**0.001 (0.001)**	0.184 (0.026)	0.036 (0.006)
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	**0.186 (0.027)**	**0.037 (0.006)**
m=6	500	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.184 (0.026)	**0.037 (0.006)**
	1000	**0.009 (0.000)**	0.018 (0.004)	0.036 (0.007)	**0.001 (0.001)**	0.181 (0.026)	0.036 (0.007)

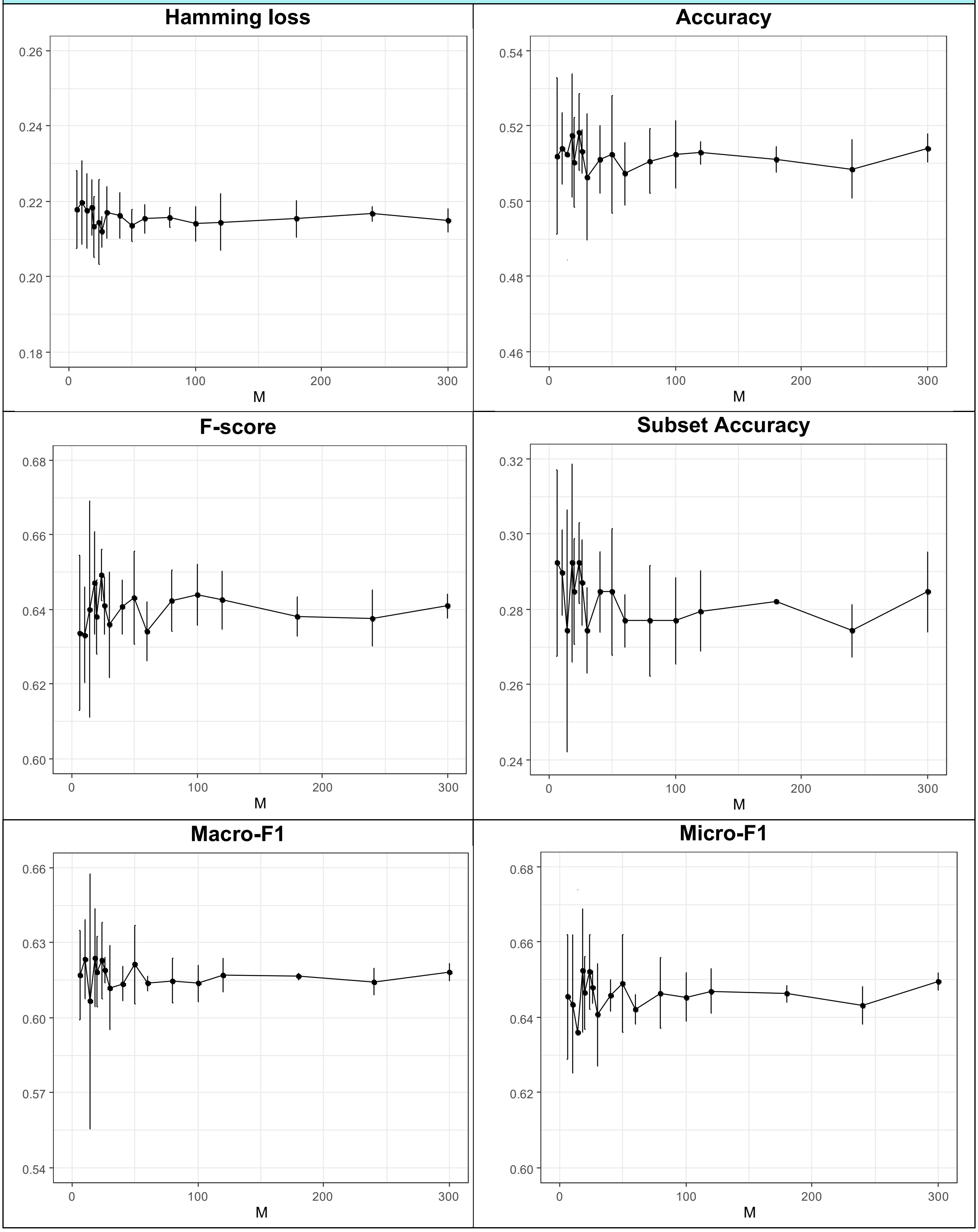
A.2 Classifying a single validation fold of the *Emotions* data multiple times

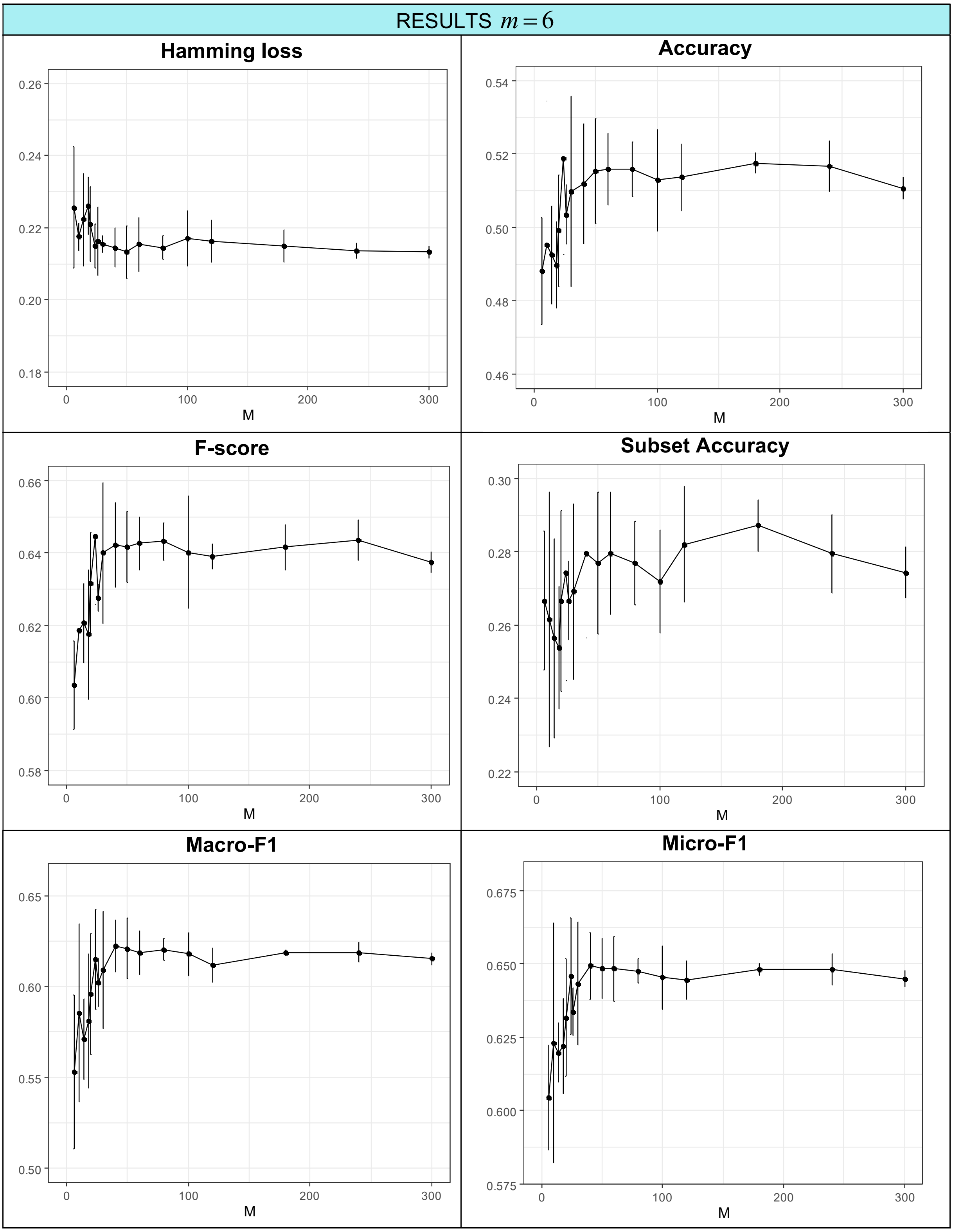



RESULTS $m = 3$
Hamming loss
Accuracy
F-score
Subset Accuracy
Macro-F1
Micro-F1
M


RESULTS $m = 4$
Hamming loss
0.26
0.24
0.22
0.20
0.18
0
100
200
300
M
Accuracy
0.54
0.52
0.50
0.48
0.46
0
100
200
300
M
F-score
0.68
0.66
0.64
0.62
0.60
0
100
200
300
M
Subset Accuracy
0.32
0.30
0.28
0.26
0.24
0
100
200
300
M
Macro-F1
0.66
0.63
0.60
0.57
0.54
0
100
200
300
M
Micro-F1
0.68
0.66
0.64
0.62
0.60
0
100
200
300
M

RESULTS $m = 5$


Hamming loss
0.26
0.24
0.22
0.20
0.18
0
100
200
300
M
Accuracy
0.54
0.52
0.50
0.48
0.46
F-score
0.68
0.66
0.64
0.62
0.60
Subset Accuracy
0.32
0.30
0.28
0.26
0.24
Macro-F1
0.66
0.63
0.60
0.57
0.54
Micro-F1
0.68
0.66
0.64
0.62
0.60


RESULTS $m = 6$
Hamming loss
0.26
0.24
0.22
0.20
0.18
0
100
200
300
M
Accuracy
0.54
0.52
0.50
0.48
0.46
0
100
200
300
M
F-score
0.66
0.64
0.62
0.60
0.58
0
100
200
300
M
Subset Accuracy
0.30
0.28
0.26
0.24
0.22
0
100
200
300
M
Macro-F1
0.65
0.60
0.55
0.50
0
100
200
300
M
Micro-F1
0.675
0.650
0.625
0.600
0.575
0
100
200
300
M

A.3 Results for different choices of m and M for Conditional LDsplit

EMOTIONS (SVM base classifier)						
	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2 M=15	0.195 (0.011)	0.499 (0.026)	0.640 (0.023)	0.274 (0.040)	0.603 (0.027)	0.636 (0.030)
m=3 M=20	**0.194 (0.011)**	**0.514 (0.029)**	**0.649 (0.023)**	**0.281 (0.027)**	**0.613 (0.028)**	**0.644 (0.031)**
m=4 M=15	0.196 (0.016)	0.511 (0.031)	0.643 (0.033)	0.279 (0.028)	0.606 (0.032)	0.633 (0.039)
m=5 M=6	0.212 (0.019)	0.488 (0.036)	0.609 (0.037)	0.258 (0.024)	0.569 (0.051)	0.595 (0.049)
m=6 M=1	0.224 (0.025)	0.490 (0.052)	0.603 (0.049)	0.262 (0.050)	0.572 (0.049)	0.586 (0.061)

EMOTIONS (Decision Tree base classifier)						
	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2 M=15	0.227 (0.015)	**0.474 (0.053)**	**0.611 (0.039)**	0.220 (0.069)	**0.588 (0.036)**	**0.610 (0.038)**
m=3 M=20	**0.217 (0.015)**	0.465 (0.017)	0.605 (0.025)	**0.225 (0.023)**	0.579 (0.019)	0.608 (0.025)
m=4 M=15	0.225 (0.020)	0.446 (0.027)	0.581 (0.031)	0.212 (0.032)	0.567 (0.025)	0.590 (0.032)
m=5 M=6	0.225 (0.020)	0.446 (0.027)	0.581 (0.031)	0.212 (0.032)	0.567 (0.025)	0.590 (0.032)
m=6 M=1	0.292 (0.024)	0.376 (0.052)	0.497 (0.059)	0.149 (0.053)	0.478 (0.042)	0.480 (0.038)

SCENE (SVM base classifier)						
	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2 M=15	0.081 (0.007)	0.680 (0.031)	0.765 (0.021)	0.638 (0.038)	0.760 (0.023)	0.749 (0.024)
m=3 M=20	0.080 (0.006)	0.691 (0.029)	**0.768 (0.019)**	0.655 (0.034)	0.764 (0.020)	0.754 (0.021)
m=4 M=15	**0.079 (0.008)**	0.716 (0.037)	**0.768 (0.025)**	0.691 (0.044)	**0.769 (0.025)**	**0.758 (0.027)**
m=5 M=6	0.082 (0.009)	0.739 (0.041)	0.767 (0.029)	0.714 (0.047)	**0.769 (0.028)**	**0.758 (0.029)**
m=6 M=1	0.087 (0.014)	**0.744 (0.041)**	0.755 (0.039)	**0.719 (0.049)**	0.761 (0.037)	0.748 (0.039)

SCENE (Decision Tree base classifier)						
	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2 M=15	0.123 (0.008)	0.499 (0.023)	0.628 (0.024)	0.430 (0.029)	0.616 (0.027)	0.608 (0.025)
m=3 M=20	0.112 (0.010)	0.525 (0.045)	0.643 (0.044)	0.486 (0.041)	0.632 (0.045)	0.629 (0.044)
m=4 M=15	**0.110 (0.008)**	0.546 (0.049)	**0.649 (0.042)**	0.514 (0.045)	**0.640 (0.039)**	**0.636 (0.038)**
m=5 M=6	**0.110 (0.008)**	0.546 (0.049)	**0.649 (0.042)**	0.514 (0.045)	**0.640 (0.039)**	**0.636 (0.038)**
m=6 M=1	0.129 (0.012)	**0.571 (0.042)**	0.624 (0.038)	**0.537 (0.040)**	0.626 (0.037)	0.611 (0.038)

YEAST (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	40	**0.188 (0.007)**	0.531 (0.014)	0.661 (0.014)	0.205 (0.024)	0.498 (0.027)	0.660 (0.014)
	91	**0.188 (0.008)**	0.530 (0.018)	0.661 (0.017)	0.207 (0.028)	0.490 (0.036)	0.659 (0.017)
m=3	40	**0.188 (0.006)**	0.535 (0.016)	0.664 (0.015)	0.210 (0.023)	0.494 (0.032)	0.661 (0.014)
	100	**0.188 (0.007)**	0.534 (0.015)	0.662 (0.014)	0.214 (0.026)	0.498 (0.024)	0.661 (0.014)
	300	**0.188 (0.007)**	0.536 (0.016)	0.664 (0.014)	0.217 (0.030)	0.493 (0.034)	0.662 (0.014)
m=4	40	0.190 (0.005)	0.534 (0.018)	0.662 (0.014)	0.212 (0.031)	**0.506 (0.033)**	0.660 (0.015)
	100	**0.188 (0.006)**	**0.538 (0.016)**	**0.666 (0.014)**	**0.223 (0.028)**	0.501 (0.026)	**0.663 (0.014)**
	300	0.189 (0.005)	0.537 (0.013)	0.665 (0.012)	0.219 (0.027)	0.493 (0.032)	0.662 (0.011)
	500	0.189 (0.006)	0.537 (0.015)	0.665 (0.013)	0.219 (0.030)	0.494 (0.033)	0.662 (0.012)
	1000	**0.188 (0.006)**	**0.538 (0.014)**	**0.666 (0.012)**	0.221 (0.030)	0.494 (0.033)	**0.663 (0.012)**
m=5	40	0.195 (0.017)	0.518 (0.056)	0.650 (0.042)	0.184 (0.091)	0.483 (0.051)	0.644 (0.048)
	100	0.189 (0.006)	**0.538 (0.017)**	0.665 (0.014)	0.220 (0.032)	0.500 (0.023)	**0.663 (0.015)**
	300	**0.188 (0.006)**	**0.538 (0.015)**	**0.666 (0.012)**	0.219 (0.035)	0.501 (0.025)	**0.663 (0.013)**
	500	0.189 (0.006)	0.537 (0.015)	0.665 (0.013)	0.219 (0.030)	0.494 (0.033)	0.662 (0.012)
	1000	**0.188 (0.006)**	**0.538 (0.014)**	**0.666 (0.012)**	0.221 (0.030)	0.494 (0.033)	**0.663 (0.012)**
m=6	40	0.205 (0.021)	0.464 (0.077)	0.606 (0.067)	0.119 (0.103)	0.459 (0.049)	0.602 (0.067)
	100	0.197 (0.016)	0.498 (0.051)	0.634 (0.039)	0.155 (0.091)	0.472 (0.052)	0.631 (0.045)
	300	0.204 (0.021)	0.477 (0.068)	0.617 (0.058)	0.132 (0.109)	0.466 (0.047)	0.612 (0.060)
	500	0.197 (0.019)	0.499 (0.074)	0.636 (0.062)	0.163 (0.097)	0.475 (0.028)	0.631 (0.067)
	1000	0.195 (0.007)	0.507 (0.043)	0.642 (0.038)	0.164 (0.081)	0.485 (0.029)	0.639 (0.035)

YEAST (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	40	0.223 (0.006)	0.450 (0.019)	0.601 (0.017)	0.074 (0.016)	0.449 (0.027)	0.597 (0.015)
	91	0.222 (0.007)	0.448 (0.018)	0.598 (0.019)	0.077 (0.022)	0.451 (0.026)	0.596 (0.013)
m=3	40	0.215 (0.007)	0.461 (0.015)	0.609 (0.016)	0.097 (0.016)	0.454 (0.024)	0.607 (0.013)
	100	0.212 (0.007)	0.472 (0.015)	0.618 (0.016)	0.101 (0.011)	**0.459 (0.029)**	0.616 (0.012)
	300	0.211 (0.006)	0.472 (0.016)	0.616 (0.018)	0.107 (0.012)	0.457 (0.027)	0.615 (0.012)
m=4	40	0.210 (0.006)	0.472 (0.021)	0.614 (0.023)	0.109 (0.022)	**0.459 (0.023)**	0.614 (0.017)
	100	0.205 (0.005)	0.482 (0.010)	**0.622 (0.012)**	0.131 (0.011)	0.445 (0.018)	0.621 (0.009)
	300	0.205 (0.007)	0.479 (0.016)	0.621 (0.018)	0.125 (0.012)	0.458 (0.029)	0.621 (0.014)
	500	0.206 (0.006)	0.479 (0.014)	0.618 (0.016)	0.131 (0.014)	0.444 (0.022)	0.619 (0.013)
	1000	**0.204 (0.007)**	**0.483 (0.016)**	**0.622 (0.017)**	**0.135 (0.011)**	0.453 (0.035)	**0.623 (0.014)**
m=5	40	0.210 (0.006)	0.472 (0.021)	0.614 (0.023)	0.109 (0.022)	**0.459 (0.023)**	0.614 (0.017)
	100	0.205 (0.005)	0.482 (0.010)	**0.622 (0.012)**	0.131 (0.011)	0.445 (0.018)	0.621 (0.009)
	300	0.205 (0.007)	0.479 (0.016)	0.621 (0.018)	0.125 (0.012)	0.458 (0.029)	0.621 (0.014)
	500	0.206 (0.006)	0.479 (0.014)	0.618 (0.016)	0.131 (0.014)	0.444 (0.022)	0.619 (0.013)
	1000	**0.204 (0.007)**	**0.483 (0.016)**	**0.622 (0.017)**	**0.135 (0.011)**	0.453 (0.035)	**0.623 (0.014)**
m=6	40	0.218 (0.014)	0.429 (0.046)	0.573 (0.042)	0.091 (0.037)	0.420 (0.041)	0.574 (0.043)
	100	0.210 (0.007)	0.449 (0.020)	0.593 (0.019)	0.111 (0.021)	0.419 (0.007)	0.594 (0.015)
	300	0.208 (0.008)	0.452 (0.015)	0.595 (0.015)	0.119 (0.014)	0.419 (0.025)	0.597 (0.015)
	500	0.207 (0.007)	0.458 (0.014)	0.601 (0.014)	0.121 (0.016)	0.430 (0.015)	0.602 (0.013)
	1000	0.205 (0.007)	0.459 (0.015)	0.602 (0.016)	0.123 (0.020)	0.427 (0.018)	0.603 (0.014)

MEDICAL (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	100	**0.012 (0.002)**	0.713 (0.043)	0.803 (0.037)	0.619 (0.049)	0.834 (0.030)	**0.787 (0.039)**
	300	**0.012 (0.002)**	0.712 (0.043)	0.803 (0.037)	0.617 (0.049)	0.829 (0.034)	**0.787 (0.038)**
	500	**0.012 (0.002)**	**0.715 (0.050)**	**0.804 (0.041)**	**0.620 (0.056)**	0.833 (0.032)	**0.787 (0.042)**
	990	**0.012 (0.002)**	0.713 (0.047)	0.803 (0.039)	0.619 (0.053)	0.829 (0.034)	0.786 (0.040)
m=3	100	0.014 (0.002)	0.623 (0.046)	0.738 (0.045)	0.536 (0.044)	0.826 (0.040)	0.729 (0.049)
	300	**0.012 (0.003)**	0.698 (0.064)	0.792 (0.052)	0.611 (0.066)	0.831 (0.030)	0.780 (0.052)
	500	**0.012 (0.003)**	0.675 (0.069)	0.778 (0.058)	0.583 (0.072)	0.833 (0.036)	0.767 (0.060)
	1000	**0.012 (0.002)**	0.712 (0.047)	0.802 (0.039)	0.619 (0.055)	0.831 (0.033)	0.786 (0.042)
m=4	100	0.017 (0.006)	0.464 (0.234)	0.579 (0.257)	0.389 (0.210)	0.828 (0.040)	0.564 (0.251)
	300	0.014 (0.004)	0.593 (0.141)	0.714 (0.116)	0.511 (0.148)	**0.844 (0.031)**	0.706 (0.118)
	500	0.013 (0.004)	0.612 (0.129)	0.729 (0.111)	0.528 (0.135)	0.841 (0.049)	0.723 (0.112)
	1000	0.013 (0.004)	0.613 (0.110)	0.728 (0.091)	0.520 (0.118)	0.837 (0.030)	0.719 (0.090)
m=5	100	0.017 (0.006)	0.464 (0.234)	0.579 (0.257)	0.389 (0.210)	0.828 (0.040)	0.564 (0.251)
	300	0.014 (0.004)	0.593 (0.141)	0.714 (0.116)	0.511 (0.148)	**0.844 (0.031)**	0.706 (0.118)
	500	0.013 (0.004)	0.612 (0.129)	0.729 (0.111)	0.528 (0.135)	0.841 (0.049)	0.723 (0.112)
	1000	0.013 (0.004)	0.613 (0.110)	0.728 (0.091)	0.520 (0.118)	0.837 (0.030)	0.719 (0.090)
m=6	100	0.021 (0.002)	0.320 (0.075)	0.465 (0.087)	0.261 (0.058)	0.815 (0.031)	0.464 (0.091)
	300	0.020 (0.007)	0.340 (0.261)	0.445 (0.287)	0.292 (0.241)	0.715 (0.137)	0.432 (0.284)
	500	0.021 (0.005)	0.291 (0.163)	0.420 (0.190)	0.247 (0.142)	0.830 (0.054)	0.409 (0.199)
	1000	0.021 (0.005)	0.281 (0.204)	0.403 (0.207)	0.228 (0.182)	0.765 (0.072)	0.405 (0.202)

MEDICAL (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	100	**0.011 (0.002)**	**0.731 (0.068)**	**0.814 (0.050)**	**0.647 (0.062)**	0.859 (0.030)	**0.795 (0.045)**
	300	**0.011 (0.002)**	**0.731 (0.068)**	**0.814 (0.050)**	**0.647 (0.062)**	0.859 (0.030)	**0.795 (0.045)**
	500	**0.011 (0.002)**	**0.731 (0.068)**	**0.814 (0.050)**	**0.647 (0.062)**	0.859 (0.030)	**0.795 (0.045)**
	990	**0.011 (0.002)**	**0.731 (0.068)**	**0.814 (0.050)**	**0.647 (0.062)**	0.859 (0.030)	**0.795 (0.045)**
m=3	100	0.013 (0.002)	0.595 (0.091)	0.725 (0.067)	0.520 (0.090)	0.878 (0.034)	0.720 (0.065)
	300	0.012 (0.002)	0.677 (0.072)	0.784 (0.049)	0.597 (0.082)	0.867 (0.039)	0.769 (0.052)
	500	0.012 (0.002)	0.670 (0.063)	0.777 (0.048)	0.602 (0.064)	0.866 (0.035)	0.768 (0.053)
	1000	**0.011 (0.001)**	0.704 (0.044)	0.799 (0.034)	0.627 (0.039)	0.860 (0.025)	0.785 (0.034)
m=4	100	0.017 (0.003)	0.468 (0.116)	0.612 (0.100)	0.394 (0.112)	0.874 (0.033)	0.606 (0.090)
	300	0.014 (0.003)	0.567 (0.100)	0.697 (0.080)	0.489 (0.102)	0.869 (0.033)	0.687 (0.081)
	500	0.014 (0.002)	0.563 (0.081)	0.701 (0.065)	0.486 (0.087)	0.870 (0.042)	0.692 (0.072)
	1000	0.013 (0.004)	0.591 (0.122)	0.719 (0.096)	0.530 (0.127)	**0.888 (0.025)**	0.717 (0.103)
m=5	100	0.017 (0.003)	0.468 (0.116)	0.612 (0.100)	0.394 (0.112)	0.874 (0.033)	0.606 (0.090)
	300	0.014 (0.003)	0.567 (0.100)	0.697 (0.080)	0.489 (0.102)	0.869 (0.033)	0.687 (0.081)
	500	0.014 (0.002)	0.563 (0.081)	0.701 (0.065)	0.486 (0.087)	0.870 (0.042)	0.692 (0.072)
	1000	0.013 (0.004)	0.591 (0.122)	0.719 (0.096)	0.530 (0.127)	**0.888 (0.025)**	0.717 (0.103)
m=6	100	0.020 (0.005)	0.323 (0.158)	0.463 (0.166)	0.267 (0.151)	0.824 (0.057)	0.472 (0.165)
	300	0.020 (0.003)	0.297 (0.107)	0.442 (0.128)	0.245 (0.101)	0.863 (0.069)	0.441 (0.120)
	500	0.022 (0.004)	0.235 (0.152)	0.354 (0.189)	0.194 (0.132)	0.848 (0.054)	0.345 (0.190)
	1000	0.021 (0.004)	0.266 (0.146)	0.399 (0.174)	0.211 (0.131)	0.874 (0.040)	0.408 (0.177)

ENRON (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	100	**0.047 (0.001)**	0.414 (0.013)	0.574 (0.011)	0.116 (0.026)	0.454 (0.039)	0.548 (0.013)
	300	**0.047 (0.001)**	0.420 (0.014)	0.580 (0.012)	0.118 (0.025)	0.459 (0.049)	0.553 (0.011)
	500	**0.047 (0.001)**	0.417 (0.011)	0.577 (0.010)	0.116 (0.026)	0.462 (0.054)	0.551 (0.010)
	1000	**0.047 (0.001)**	0.418 (0.012)	0.578 (0.011)	0.116 (0.026)	**0.467 (0.050)**	0.551 (0.010)
m=3	100	**0.047 (0.001)**	0.421 (0.012)	**0.581 (0.012)**	0.117 (0.026)	0.458 (0.043)	**0.554 (0.010)**
	300	**0.047 (0.001)**	0.417 (0.012)	0.578 (0.012)	0.115 (0.026)	0.460 (0.037)	0.551 (0.011)
	500	**0.047 (0.001)**	0.418 (0.013)	0.578 (0.012)	0.115 (0.029)	0.461 (0.056)	0.552 (0.012)
	1000	**0.047 (0.001)**	0.417 (0.012)	0.577 (0.011)	0.117 (0.029)	0.460 (0.054)	0.551 (0.010)
m=4	100	**0.047 (0.001)**	0.419 (0.014)	0.578 (0.013)	0.119 (0.029)	0.465 (0.051)	0.549 (0.009)
	300	**0.047 (0.001)**	0.419 (0.015)	0.579 (0.013)	0.115 (0.029)	0.465 (0.048)	0.551 (0.012)
	500	**0.047 (0.001)**	0.418 (0.012)	0.579 (0.012)	0.115 (0.028)	**0.467 (0.052)**	0.552 (0.011)
	1000	**0.047 (0.001)**	0.418 (0.012)	0.578 (0.011)	0.115 (0.028)	0.466 (0.051)	0.552 (0.011)
m=5	100	**0.047 (0.001)**	0.419 (0.014)	0.578 (0.013)	0.119 (0.029)	0.465 (0.051)	0.549 (0.009)
	300	**0.047 (0.001)**	0.419 (0.015)	0.579 (0.013)	0.115 (0.029)	0.465 (0.048)	0.551 (0.012)
	500	**0.047 (0.001)**	0.418 (0.012)	0.579 (0.012)	0.115 (0.028)	**0.467 (0.052)**	0.552 (0.011)
	1000	**0.047 (0.001)**	0.418 (0.012)	0.578 (0.011)	0.115 (0.028)	0.466 (0.051)	0.552 (0.011)
m=6	100	**0.047 (0.001)**	**0.423 (0.012)**	**0.581 (0.013)**	**0.120 (0.023)**	0.464 (0.053)	0.552 (0.009)
	300	**0.047 (0.001)**	0.421 (0.011)	**0.581 (0.012)**	**0.120 (0.025)**	0.466 (0.053)	0.553 (0.011)
	500	**0.047 (0.001)**	0.421 (0.015)	**0.581 (0.014)**	**0.120 (0.029)**	0.463 (0.055)	0.552 (0.013)
	1000	**0.047 (0.001)**	0.420 (0.010)	0.579 (0.011)	0.119 (0.025)	0.464 (0.054)	0.551 (0.010)

ENRON (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	100	0.051 (0.002)	0.392 (0.019)	0.552 (0.021)	0.084 (0.014)	0.431 (0.032)	0.539 (0.015)
	300	0.051 (0.002)	**0.401 (0.024)**	**0.559 (0.025)**	0.084 (0.012)	0.435 (0.028)	**0.546 (0.017)**
	500	0.051 (0.002)	0.400 (0.024)	**0.559 (0.026)**	0.085 (0.013)	0.435 (0.028)	**0.546 (0.017)**
	1000	0.051 (0.002)	**0.401 (0.024)**	0.401 (0.026)	0.085 (0.013)	0.436 (0.028)	**0.546 (0.017)**
m=3	100	0.051 (0.001)	0.391 (0.022)	0.548 (0.020)	0.088 (0.013)	0.424 (0.023)	0.536 (0.013)
	300	0.051 (0.002)	0.398 (0.021)	0.556 (0.023)	0.082 (0.011)	0.429 (0.030)	0.543 (0.015)
	500	0.051 (0.002)	0.398 (0.023)	0.555 (0.025)	0.085 (0.013)	0.435 (0.028)	0.543 (0.016)
	1000	0.051 (0.002)	0.399 (0.023)	0.557 (0.025)	0.084 (0.012)	0.434 (0.027)	0.545 (0.016)
m=4	100	0.050 (0.002)	0.394 (0.017)	0.551 (0.019)	0.080 (0.015)	0.433 (0.023)	0.537 (0.012)
	300	0.050 (0.002)	0.396 (0.024)	0.553 (0.027)	0.084 (0.011)	**0.439 (0.021)**	0.542 (0.017)
	500	0.050 (0.002)	0.400 (0.020)	**0.559 (0.021)**	0.085 (0.005)	0.433 (0.030)	**0.546 (0.015)**
	1000	0.050 (0.002)	0.400 (0.025)	0.558 (0.027)	0.086 (0.013)	0.435 (0.027)	0.545 (0.019)
m=5	100	0.050 (0.002)	0.394 (0.017)	0.551 (0.019)	0.080 (0.015)	0.433 (0.023)	0.537 (0.012)
	300	0.050 (0.002)	0.396 (0.024)	0.553 (0.027)	0.084 (0.011)	**0.439 (0.021)**	0.542 (0.017)
	500	0.050 (0.002)	0.400 (0.020)	**0.559 (0.021)**	0.085 (0.005)	0.433 (0.030)	**0.546 (0.015)**
	1000	0.050 (0.002)	0.400 (0.025)	0.558 (0.027)	0.086 (0.013)	0.435 (0.027)	0.545 (0.019)
m=6	100	**0.049 (0.002)**	0.397 (0.023)	0.552 (0.021)	0.087 (0.012)	**0.439 (0.027)**	0.543 (0.017)
	300	**0.049 (0.001)**	0.400 (0.021)	0.556 (0.021)	0.089 (0.014)	0.435 (0.025)	0.545 (0.017)
	500	**0.049 (0.001)**	0.396 (0.022)	0.553 (0.023)	0.081 (0.006)	0.431 (0.020)	0.541 (0.018)
	1000	**0.049 (0.001)**	0.400 (0.020)	0.557 (0.022)	**0.091 (0.010)**	0.430 (0.010)	0.544 (0.015)

COREL5K (SVM base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	500	**0.009 (<0.001)**	**0.054 (0.005)**	**0.103 (0.010)**	**0.009 (0.002)**	0.257 (0.019)	**0.105 (0.010)**
	1000	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.011)	**0.009 (0.002)**	0.259 (0.019)	0.104 (0.011)
m=3	500	**0.009 (<0.001)**	0.050 (0.001)	0.096 (0.003)	0.008 (0.002)	0.247 (0.014)	0.097 (0.003)
	1000	**0.009 (<0.001)**	0.053 (0.006)	0.102 (0.011)	0.008 (0.002)	0.257 (0.019)	0.104 (0.012)
m=4	500	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.010)	0.008 (0.004)	0.258 (0.012)	0.103 (0.011)
	1000	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.011)	**0.009 (0.002)**	0.258 (0.019)	0.104 (0.012)
m=5	500	**0.009 (<0.001)**	0.052 (0.005)	0.100 (0.001)	0.008 (0.002)	0.256 (0.018)	0.102 (0.010)
	1000	**0.009 (<0.001)**	0.053 (0.005)	0.102 (0.001)	**0.009 (0.002)**	0.257 (0.015)	0.103 (0.011)
m=6	500	**0.009 (<0.001)**	0.052 (0.006)	0.100 (0.012)	0.008 (0.002)	**0.261 (0.017)**	0.101 (0.012)
	1000	**0.009 (<0.001)**	0.052 (0.005)	0.101 (0.010)	0.008 (0.002)	0.252 (0.020)	0.102 (0.011)

COREL5K (Decision Tree base classifier)							
	M	Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	500	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.184 (0.026)	**0.037 (0.006)**
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.184 (0.026)	**0.037 (0.006)**
m=3	500	**0.009 (0.000)**	0.017 (0.004)	0.033 (0.008)	**0.001 (0.001)**	0.172 (0.023)	0.033 (0.008)
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	**0.186 (0.028)**	**0.037 (0.006)**
m=4	500	**0.009 (0.000)**	0.015 (0.002)	0.030 (0.003)	**0.001 (0.001)**	0.182 (0.035)	0.030 (0.003)
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.185 (0.025)	**0.037 (0.006)**
m=5	500	**0.009 (0.000)**	0.015 (0.002)	0.030 (0.003)	**0.001 (0.001)**	0.182 (0.035)	0.030 (0.003)
	1000	**0.009 (0.000)**	**0.019 (0.003)**	**0.037 (0.006)**	**0.001 (0.001)**	0.185 (0.025)	**0.037 (0.006)**
m=6	500	**0.009 (0.000)**	0.018 (0.003)	0.035 (0.006)	**0.001 (0.001)**	0.184 (0.028)	0.035 (0.005)
	1000	**0.009 (0.000)**	0.018 (0.003)	0.035 (0.006)	**0.001 (0.001)**	0.184 (0.029)	0.035 (0.006)

A.4 Comparison of Random and Conditional LDsplit cross-validation results

EMOTIONS (SVM base classifier)							
		Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	**Random M=18**	**0.194**	0.493 (1.2%)	0.639 (0.2%)	0.271 (1.1%)	**0.605**	**0.636**
	Random M=30	0.196 (1.0%)	0.488 (2.2%)	0.633 (1.1%)	0.268 (2.2%)	0.603 (0.3%)	0.632 (0.6%)
	Conditional M=15	0.195 (0.5%)	**0.499**	**0.640**	**0.274**	0.603 (0.3%)	**0.636**
m=3	**Random M=18**	0.194 (1.6%)	0.505 (1.8%)	0.647 (0.3%)	0.279 (0.7%)	0.608 (0.8%)	0.640 (0.6%)
	Random M=100	**0.191**	0.503 (2.1%)	0.646 (0.5%)	**0.281**	**0.613**	0.642 (0.3%)
	Conditional M=20	0.194 (1.6%)	**0.514**	**0.649**	**0.281**	**0.613**	**0.644**
m=4	**Random M=18**	0.195 (1.6%)	0.501 (3.1%)	0.640 (2.1%)	**0.279**	0.609 (1.1%)	0.639 (1.4%)
	Random M=100	**0.192**	**0.517**	**0.654**	**0.279**	**0.616**	**0.648**
	Conditional M=15	0.196 (2.1%)	0.511 (1.2%)	0.643 (1.7%)	**0.279**	0.606 (1.6%)	0.633 (2.3%)
m=5	**Random M=6**	0.203 (9.1%)	0.499 (6.2%)	0.624 (6.4%)	0.274 (6.8%)	0.576 (7.7%)	0.617 (6.4%)
	Random M=100	**0.186**	**0.532**	**0.667**	**0.294**	**0.624**	**0.659**
	Conditional M=6	0.212 (14.0%)	0.488 (8.3%)	0.609 (8.7%)	0.258 (12.2%)	0.569 (8.8%)	0.595 (9.7%)
m=6	**Random M=6**	0.209 (10.6%)	0.496 (6.6%)	0.620 (6.1%)	0.263 (9.6%)	0.576 (9.6%)	0.611 (6.9%)
	Random M=100	**0.189**	**0.531**	**0.660**	**0.291**	**0.620**	**0.656**
	Conditional M=1	0.224 (18.5%)	0.490 (7.7%)	0.603 (8.6%)	0.262 (10%)	0.572 (10.0%)	0.586 (10.7%)

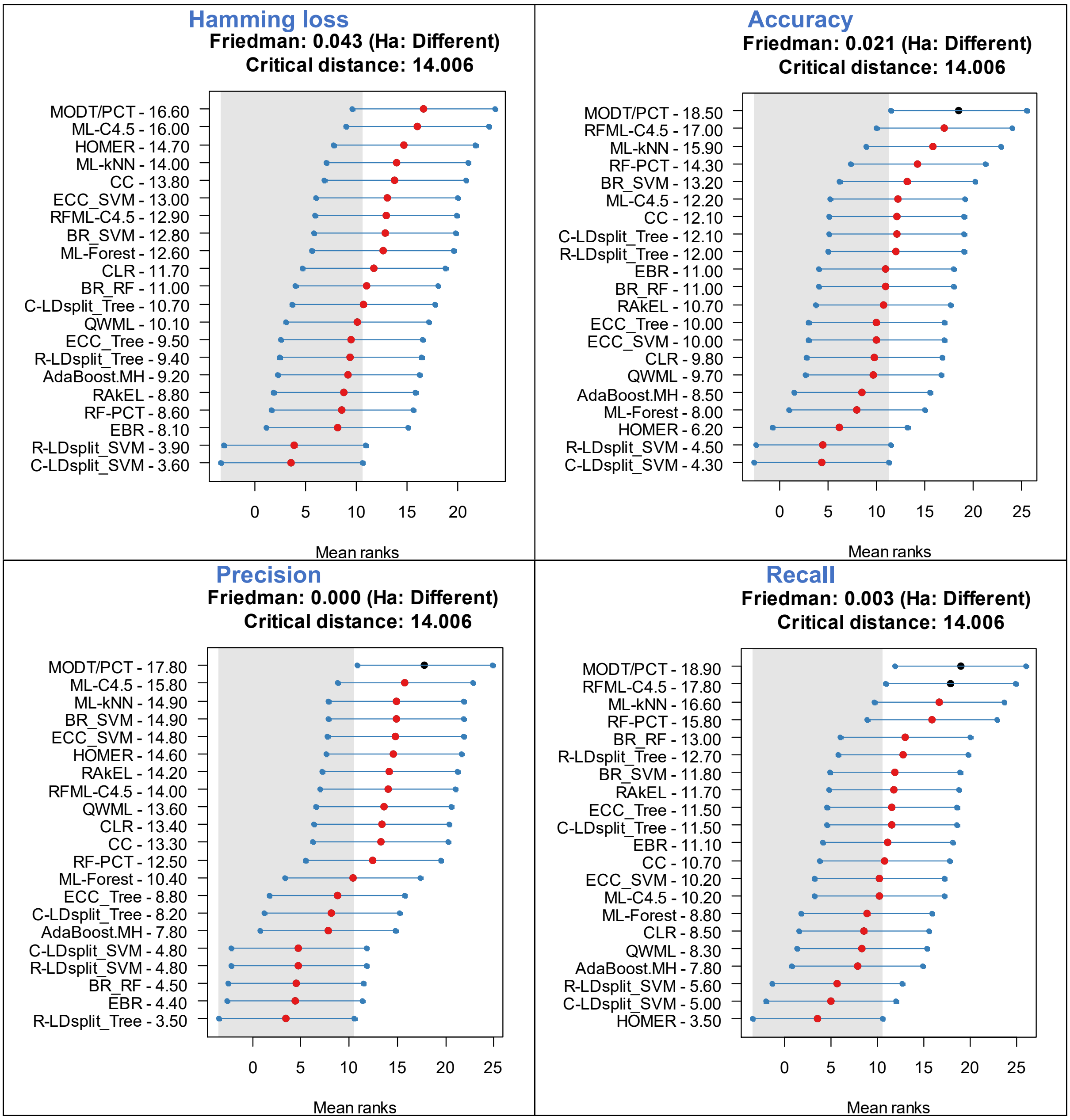
SCENE (SVM base classifier)							
		Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	**Random M=18**	0.082 (1.2%)	0.677 (0.4%)	0.764 (0.1%)	0.635 (0.5%)	0.758 (0.3%)	0.748 (0.1%)
	Random M=30	0.082 (1.2%)	0.677 (0.4%)	0.763 (0.3%)	0.634 (0.6%)	0.758 (0.3%)	0.748 (0.1%)
	Conditional M=15	**0.081**	**0.680**	**0.765**	**0.638**	**0.760**	**0.749**
m=3	**Random M=18**	0.081 (1.3%)	0.689 (0.3%)	0.767 (0.1%)	0.650 (0.8%)	0.763 (0.1%)	0.752 (0.3%)
	Random M=100	0.081 (1.3%)	0.684 (1.0%)	**0.768**	0.643 (1.8%)	0.762 (0.3%)	0.752 (0.3%)
	Conditional M=20	**0.080**	**0.691**	**0.768**	**0.655**	**0.764**	**0.754**
m=4	**Random M=18**	0.082 (3.8%)	0.699 (2.4%)	0.766 (0.9%)	0.663 (4.1%)	0.761 (1.0%)	0.751 (1.1%)
	Random M=100	**0.079**	0.701 (2.1%)	**0.773**	0.665 (3.8%)	0.768 (0.1%)	**0.759**
	Conditional M=15	**0.079**	**0.716**	0.768 (0.6%)	**0.691**	**0.769**	0.758 (0.1%)
m=5	**Random M=6**	0.080 (2.6%)	0.712 (3.7%)	0.766 (1.0%)	0.684 (4.2%)	0.765 (1.0%)	0.755 (1.0%)
	Random M=100	**0.078**	0.716 (3.1%)	**0.774**	0.686 (3.9%)	**0.773**	**0.763**
	Conditional M=6	0.082 (5.1%)	**0.739**	0.767 (0.9%)	**0.714**	0.769 (0.5%)	0.758 (0.7%)
m=6	**Random M=6**	0.079 (3.9%)	0.715 (3.9%)	0.768 (1.8%)	0.690 (4.0%)	0.766 (2.0%)	0.757 (1.8%)
	Random M=100	**0.076**	0.735 (1.2%)	**0.782**	0.709 (1.4%)	**0.782**	**0.771**
	Conditional M=1	0.087 (14.5%)	**0.744**	0.755 (3.5%)	**0.719**	0.761 (2.7%)	0.748 (3.0%)

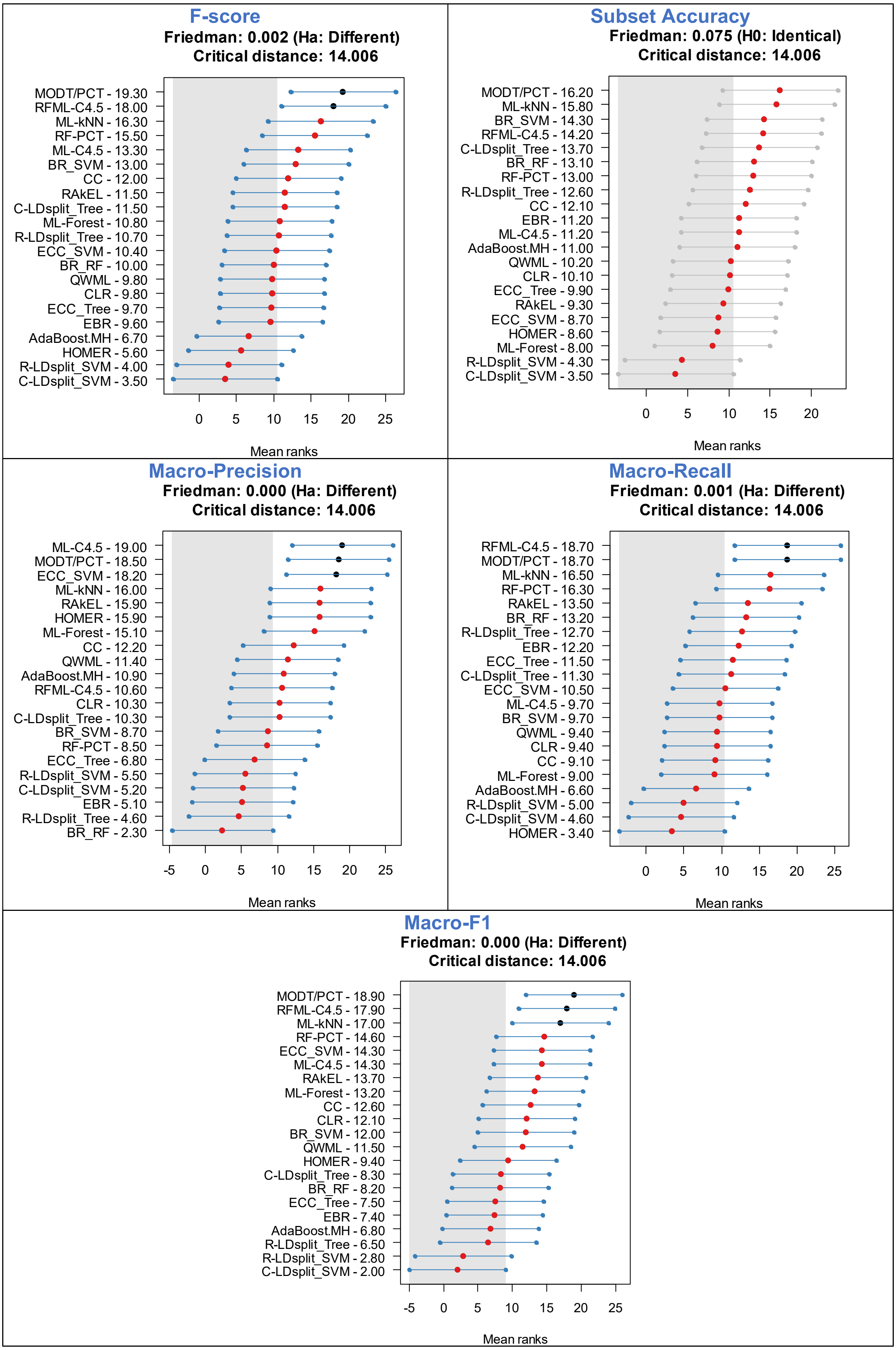
YEAST (SVM base classifier)							
		Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	**Random M=40**	**0.188**	0.529 (0.4%)	0.659 (0.3%)	**0.205**	0.495 (0.6%)	0.658 (0.3%)
	Random M=100	**0.188**	0.527 (0.8%)	0.659 (0.3%)	0.199 (2.9%)	0.488 (2.0%)	0.658 (0.3%)
	Conditional M=40	**0.188**	**0.531**	**0.661**	**0.205**	**0.498**	**0.660**
m=3	**Random M=100**	0.186 (1.1%)	0.534 (0.7%)	0.662 (0.7%)	0.213 (1.4%)	**0.503**	0.663 (0.6%)
	Random M=300	**0.184**	**0.538**	**0.667**	**0.216**	0.490 (2.65)	**0.667**
	Conditional M=100	0.188 (2.2%)	0.534 (0.7%)	0.662 (0.7%)	0.214 (0.9%)	0.498 (1.0%)	0.661 (0.9%)
m=4	**Random M=100**	0.187 (1.6%)	0.531 (1.5%)	0.660 (1.0%)	0.209 (6.3%)	**0.513**	0.661 (0.9%)
	Random M=300	**0.184**	**0.539**	**0.667**	0.219 (1.8%)	0.482 (6.0%)	**0.667**
	Conditional M=100	0.188 (2.2%)	0.538 (0.2%)	0.666 (0.1%)	**0.223**	0.501 (2.3%)	0.663 (0.6%)
m=5	**Random M=100**	0.186 (1.1%)	0.533 (0.9%)	0.661 (0.8%)	**0.220**	**0.536**	0.661 (0.6%)
	Random M=300	**0.184**	0.537 (0.2%)	**0.666**	0.218 (0.9%)	0.486 (9.3%)	**0.665**
	Conditional M=100	0.189 (2.7%)	**0.538**	0.665 (0.2%)	**0.220**	0.500 (6.7%)	0.663 (0.3%)
m=6	**Random M=100**	0.187 (1.6%)	0.528 (1.9%)	**0.656**	0.209 (6.3%)	0.508 (1.9%)	0.657 (1.4%)
	Random M=300	**0.184**	**0.538**	0.538 (18.0%)	**0.223**	**0.518**	**0.666**
	Conditional M=100	0.197 (7.1%)	0.498 (7.4%)	0.634 (3.4%)	0.155 (30.5%)	0.472 (8.9%)	0.631 (5.3%)

MEDICAL (Decision Tree base classifier)							
		Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	**Random M=300**	**0.011**	**0.731**	**0.814**	**0.647**	**0.859**	**0.795**
	Random M=500	**0.011**	**0.731**	**0.814**	**0.647**	**0.859**	**0.795**
	Conditional M=300	**0.011**	**0.731**	**0.814**	**0.647**	**0.859**	**0.795**
m=3	**Random M=300**	**0.011**	**0.734**	**0.817**	**0.652**	0.859 (0.9%)	**0.799**
	Random M=500	**0.011**	0.731 (0.4%)	0.814 (0.4%)	0.647 (0.8%)	0.859 (0.9%)	0.795 (0.5%)
	Conditional M=300	0.012 (9.1%)	0.677 (7.8%)	0.784 (4.0%)	0.597 (4.5%)	**0.867**	0.769 (3.8%)
m=4	**Random M=300**	0.011 (10.0%)	**0.733**	**0.817**	**0.652**	0.867 (0.2%)	0.798 (0.1%)
	Random M=500	**0.010**	0.730 (0.4%)	**0.817**	0.650 (0.3%)	0.862 (0.8%)	**0.799**
	Conditional M=300	0.014 (40.0%)	0.567 (22.6%)	0.697 (14.7%)	0.489 (25.0%)	**0.869**	0.687 (14.0%)
m=5	**Random M=300**	0.011 (10.0%)	**0.733**	**0.817**	**0.652**	0.867 (0.2%)	0.798 (0.1%)
	Random M=500	**0.010**	0.730 (0.4%)	**0.817**	0.650 (0.3%)	0.862 (0.8%)	**0.799**
	Conditional M=300	0.014 (40.0%)	0.567 (22.6%)	0.697 (14.7%)	0.489 (25.0%)	**0.869**	0.687 (14.0%)
m=6	**Random M=300**	**0.011**	**0.716**	**0.808**	**0.641**	0.874 (0.2%)	**0.794**
	Random M=500	**0.011**	0.704 (1.7%)	0.801 (0.9%)	0.628 (2.0%)	**0.876**	0.787 (0.9%)
	Conditional M=300	0.020 (81.8%)	0.297 (58.5%)	0.442 (45.3%)	0.245 (61.8%)	0.863 (1.5%)	0.441 (44.5%)

ENRON (SVM base classifier)							
		Hamming loss	Accuracy	F-score	Subset-accuracy	Macro-F1	Micro-F1
m=2	**Random M=300**	**0.011**	**0.731**	**0.814**	**0.647**	**0.859**	**0.795**
	Random M=500	**0.011**	**0.731**	**0.814**	**0.647**	**0.859**	**0.795**
	Conditional M=300	**0.011**	**0.731**	**0.814**	**0.647**	**0.859**	**0.795**
m=3	**Random M=300**	**0.011**	**0.734**	**0.817**	**0.652**	0.859 (0.9%)	**0.799**
	Random M=500	**0.011**	0.731 (0.4%)	0.814 (0.4%)	0.647 (0.8%)	0.859 (0.9%)	0.795 (0.5%)
	Conditional M=300	0.012 (9.1%)	0.677 (7.8%)	0.784 (4.0%)	0.597 (4.5%)	**0.867**	0.769 (3.8%)
m=4	**Random M=300**	0.011 (10.0%)	**0.733**	**0.817**	**0.652**	0.867 (0.2%)	0.798 (0.1%)
	Random M=500	**0.010**	0.730 (0.4%)	**0.817**	0.650 (0.3%)	0.862 (0.8%)	**0.799**
	Conditional M=300	0.014 (40.0%)	0.567 (22.6%)	0.697 (14.7%)	0.489 (25.0%)	**0.869**	0.687 (14.0%)
m=5	**Random M=300**	0.011 (10.0%)	**0.733**	**0.817**	**0.652**	0.867 (0.2%)	0.798 (0.1%)
	Random M=500	**0.010**	0.730 (0.4%)	**0.817**	0.650 (0.3%)	0.862 (0.8%)	**0.799**
	Conditional M=300	0.014 (40.0%)	0.567 (22.6%)	0.697 (14.7%)	0.489 (25.0%)	**0.869**	0.687 (14.0%)
m=6	**Random M=300**	**0.011**	**0.716**	**0.808**	**0.641**	0.874 (0.2%)	**0.794**
	Random M=500	**0.011**	0.704 (1.7%)	0.801 (0.9%)	0.628 (2.0%)	**0.876**	0.787 (0.9%)
	Conditional M=300	0.020 (81.8%)	0.297 (58.5%)	0.442 (45.3%)	0.245 (61.8%)	0.863 (1.5%)	0.441 (44.5%)

A.5 Mean rank diagrams based on the Friedman and Nemenyi tests




F-score
Friedman: 0.002 (Ha: Different)
Critical distance: 14.006
MODT/PCT - 19.30
RFML-C4.5 - 18.00
ML-kNN - 16.30
RF-PCT - 15.50
ML-C4.5 - 13.30
BR_SVM - 13.00
CC - 12.00
RAkEL - 11.50
C-LDsplit_Tree - 11.50
ML-Forest - 10.80
R-LDsplit_Tree - 10.70
ECC_SVM - 10.40
BR_RF - 10.00
QWML - 9.80
CLR - 9.80
ECC_Tree - 9.70
EBR - 9.60
AdaBoost.MH - 6.70
HOMER - 5.60
R-LDsplit_SVM - 4.00
C-LDsplit_SVM - 3.50
0 5 10 15 20 25
Mean ranks
Subset Accuracy
Friedman: 0.075 (H0: Identical)
Critical distance: 14.006
MODT/PCT - 16.20
ML-kNN - 15.80
BR_SVM - 14.30
RFML-C4.5 - 14.20
C-LDsplit_Tree - 13.70
BR_RF - 13.10
RF-PCT - 13.00
R-LDsplit_Tree - 12.60
CC - 12.10
EBR - 11.20
ML-C4.5 - 11.20
AdaBoost.MH - 11.00
QWML - 10.20
CLR - 10.10
ECC_Tree - 9.90
RAkEL - 9.30
ECC_SVM - 8.70
HOMER - 8.60
ML-Forest - 8.00
R-LDsplit_SVM - 4.30
C-LDsplit_SVM - 3.50
0 5 10 15 20
Mean ranks
Macro-Precision
Friedman: 0.000 (Ha: Different)
Critical distance: 14.006
ML-C4.5 - 19.00
MODT/PCT - 18.50
ECC_SVM - 18.20
ML-kNN - 16.00
RAkEL - 15.90
HOMER - 15.90
ML-Forest - 15.10
CC - 12.20
QWML - 11.40
AdaBoost.MH - 10.90
RFML-C4.5 - 10.60
CLR - 10.30
C-LDsplit_Tree - 10.30
BR_SVM - 8.70
RF-PCT - 8.50
ECC_Tree - 6.80
R-LDsplit_SVM - 5.50
C-LDsplit_SVM - 5.20
EBR - 5.10
R-LDsplit_Tree - 4.60
BR_RF - 2.30
-5 0 5 10 15 20 25
Mean ranks
Macro-Recall
Friedman: 0.001 (Ha: Different)
Critical distance: 14.006
RFML-C4.5 - 18.70
MODT/PCT - 18.70
ML-kNN - 16.50
RF-PCT - 16.30
RAkEL - 13.50
BR_RF - 13.20
R-LDsplit_Tree - 12.70
EBR - 12.20
ECC_Tree - 11.50
C-LDsplit_Tree - 11.30
ECC_SVM - 10.50
ML-C4.5 - 9.70
BR_SVM - 9.70
QWML - 9.40
CLR - 9.40
CC - 9.10
ML-Forest - 9.00
AdaBoost.MH - 6.60
R-LDsplit_SVM - 5.00
C-LDsplit_SVM - 4.60
HOMER - 3.40
0 5 10 15 20 25
Mean ranks
Macro-F1
Friedman: 0.000 (Ha: Different)
Critical distance: 14.006
MODT/PCT - 18.90
RFML-C4.5 - 17.90
ML-kNN - 17.00
RF-PCT - 14.60
ECC_SVM - 14.30
ML-C4.5 - 14.30
RAkEL - 13.70
ML-Forest - 13.20
CC - 12.60
CLR - 12.10
BR_SVM - 12.00
QWML - 11.50
HOMER - 9.40
C-LDsplit_Tree - 8.30
BR_RF - 8.20
ECC_Tree - 7.50
EBR - 7.40
AdaBoost.MH - 6.80
R-LDsplit_Tree - 6.50
R-LDsplit_SVM - 2.80
C-LDsplit_SVM - 2.00
-5 0 5 10 15 20 25
Mean ranks

Appendix B: Additional results for synthetic data study of Chapter 6

This appendix contains the additional heatmaps referred to in Section 6.2.4 and Section 6.2.5.

The local importance heatmaps in **Appendix B.1 – Appendix B.6** are found as follows. For each of the 50 $\mathbf{F}$ matrices found for a case in the synthetic study of Section 6.2, the six, five, four, three, two and one most important variables for Y_1, Y_2, Y_3, Y_4, Y_5 and Y_6 respectively, are marked “relevant”. The $(l,k)^{th}$ entry of a heatmap in Appendix B.1 – Appendix B.6 gives the proportion of time variable X_l is considered “relevant” for label Y_k across the 50 local importance matrices of that case.

The local importance heatmaps in **Appendix B.7 – Appendix B.11** are found as follows. For each LDsplit MDA local importance matrix of a case obtained in the synthetic study of Section 6.2, the median importance value is found per label and all input variables that exceed this importance value per label are marked as “relevant” for the label. With this method of determining relevancy, in Appendix B.7 – Appendix B.11, the $(l,k)^{th}$ entry of a heatmap gives the proportion of time variable X_l is considered relevant for label Y_k across the 50 local importance matrices of that case. Note that Cases $17-32$ are represented in Figure 6.5. in Section 6.2.5 and are therefore not given in this appendix.

B.1 Local heatmaps for Case $1-16$

Case 1

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.04	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.56	0.00
X_5	0.96	0.80	0.80	0.92	1.00	0.00
X_6	0.92	0.20	0.16	0.08	0.44	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 2

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	1.00	0.98	0.00	0.00	0.00	0.00
X_3	0.98	1.00	1.00	0.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.02	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 3

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.02	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	0.08	0.00
X_5	0.88	0.74	0.90	0.90	1.00	0.00
X_6	0.72	0.26	0.08	0.08	0.92	1.00
X_7	0.16	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.12	0.00	0.00	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 4

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.98	0.02	0.00	0.00	0.00	0.00
X_2	0.96	1.00	0.02	0.00	0.00	0.00
X_3	0.92	1.00	0.98	0.00	0.00	0.00
X_4	1.00	0.96	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.02	0.00	0.00	0.00	0.00

Case 5

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.68	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.04	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.98	1.00	1.00	1.00	0.00
X_5	0.90	0.60	0.30	0.94	1.00	0.00
X_6	0.94	0.38	0.02	0.02	0.00	1.00
X_7	0.02	0.02	0.00	0.00	0.00	0.00
X_8	0.02	0.02	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 6

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.74	1.00	0.06	0.00	0.00	0.00
X_2	0.84	0.50	0.14	0.00	0.00	0.00
X_3	1.00	0.60	0.86	0.00	0.00	0.00
X_4	1.00	0.94	0.94	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.06	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.14	0.00	0.00	0.00	0.00	0.00
X_{10}	0.12	0.00	0.00	0.00	0.00	0.00

Case 7

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.88	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.98	1.00	1.00	0.96	0.00
X_5	0.98	0.74	0.12	0.96	1.00	0.00
X_6	0.78	0.22	0.00	0.02	0.04	1.00
X_7	0.14	0.02	0.00	0.00	0.00	0.00
X_8	0.06	0.02	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.02	0.00	0.00	0.00	0.00

Case 8

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.54	1.00	0.54	0.00	0.00	0.00
X_2	0.78	0.60	0.80	0.00	0.00	0.00
X_3	0.78	0.58	0.86	0.00	0.00	0.00
X_4	1.00	0.74	0.40	1.00	0.00	0.00
X_5	1.00	1.00	0.40	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.24	0.02	0.00	0.00	0.00	0.00
X_8	0.22	0.02	0.00	0.00	0.00	0.00
X_9	0.22	0.02	0.00	0.00	0.00	0.00
X_{10}	0.22	0.04	0.00	0.00	0.00	0.00

Case 9

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.66	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.96	0.96	0.34	1.00	1.00	0.00
X_6	0.86	0.04	0.00	0.00	0.00	1.00
X_7	0.08	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 10

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.56	0.00	0.00	0.00
X_3	1.00	0.84	1.00	0.00	0.00	0.00
X_4	1.00	0.38	0.98	1.00	0.00	0.00
X_5	1.00	0.78	0.52	1.00	1.00	0.00
X_6	1.00	1.00	0.96	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 11

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.98	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	1.00	0.00
X_5	0.86	0.90	0.02	0.98	1.00	0.00
X_6	0.70	0.06	0.00	0.00	0.00	1.00
X_7	0.16	0.02	0.00	0.00	0.00	0.00
X_8	0.16	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.02	0.00	0.00	0.00	0.00
X_{10}	0.12	0.00	0.00	0.00	0.00	0.00

Case 12

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.76	0.00	0.00
X_4	1.00	0.14	1.00	1.00	0.00	0.00
X_5	1.00	0.90	0.00	1.00	1.00	0.00
X_6	1.00	0.96	1.00	0.24	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 13

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.90	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.98	0.96	0.10	1.00	1.00	0.00
X_6	0.82	0.04	0.00	0.00	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 14

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.98	0.00	0.00	0.00
X_3	1.00	0.84	1.00	0.04	0.00	0.00
X_4	1.00	0.36	0.94	1.00	0.00	0.00
X_5	1.00	0.80	0.20	1.00	1.00	0.00
X_6	1.00	1.00	0.88	0.96	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 15

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	1.00	0.00
X_5	0.84	0.80	0.00	0.98	1.00	0.00
X_6	0.74	0.18	0.00	0.00	0.00	1.00
X_7	0.14	0.02	0.00	0.00	0.00	0.00
X_8	0.12	0.00	0.00	0.00	0.00	0.00
X_9	0.10	0.00	0.00	0.00	0.00	0.00
X_{10}	0.08	0.02	0.00	0.00	0.00	0.00

Case 16

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.10	1.00	1.00	0.00	0.00
X_5	1.00	0.96	0.00	1.00	1.00	0.00
X_6	1.00	0.94	1.00	0.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.2 Local heatmaps for Case $17-32$

Case 17

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.04	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.96	1.00	1.00	0.54	0.00
X_5	0.94	0.72	0.70	0.96	1.00	0.00
X_6	0.78	0.32	0.26	0.04	0.46	1.00
X_7	0.06	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 18

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	0.98	1.00	0.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 19

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	0.04	0.00
X_5	0.88	0.76	0.88	0.86	1.00	0.00
X_6	0.72	0.22	0.12	0.14	0.96	1.00
X_7	0.14	0.02	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.14	0.00	0.00	0.00	0.00	0.00

Case 20

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.02	0.00	0.00	0.00	0.00
X_2	0.98	1.00	0.00	0.00	0.00	0.00
X_3	0.96	0.98	1.00	0.00	0.00	0.00
X_4	1.00	0.98	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.02	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 21

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.84	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.92	0.90	0.16	0.98	1.00	0.00
X_6	0.86	0.10	0.00	0.02	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 22

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.90	1.00	0.06	0.00	0.00	0.00
X_2	0.94	0.34	0.06	0.00	0.00	0.00
X_3	1.00	0.68	0.96	0.00	0.00	0.00
X_4	1.00	0.98	0.92	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.10	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 23

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.92	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.98	1.00	1.00	0.88	0.00
X_5	0.92	0.72	0.08	0.98	1.00	0.00
X_6	0.82	0.28	0.00	0.00	0.12	1.00
X_7	0.06	0.02	0.00	0.00	0.00	0.00
X_8	0.08	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 24

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.54	1.00	0.56	0.00	0.00	0.00
X_2	0.78	0.70	0.88	0.00	0.00	0.00
X_3	0.92	0.62	0.76	0.00	0.00	0.00
X_4	1.00	0.68	0.22	1.00	0.00	0.00
X_5	1.00	1.00	0.58	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.20	0.00	0.00	0.00	0.00	0.00
X_8	0.16	0.00	0.00	0.00	0.00	0.00
X_9	0.22	0.00	0.00	0.00	0.00	0.00
X_{10}	0.18	0.00	0.00	0.00	0.00	0.00

Case 25

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.80	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.96	0.94	0.22	1.00	1.00	0.00
X_6	0.82	0.06	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.08	0.00	0.00	0.00	0.00	0.00

Case 26

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.56	0.00	0.00	0.00
X_3	1.00	0.90	1.00	0.00	0.00	0.00
X_4	1.00	0.20	0.86	1.00	0.00	0.00
X_5	1.00	0.90	0.58	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 27

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.98	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.84	0.02	1.00	1.00	0.00
X_6	0.82	0.16	0.00	0.00	0.00	1.00
X_7	0.08	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 28

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.72	0.00	0.00
X_4	1.00	0.06	1.00	1.00	0.00	0.00
X_5	1.00	0.96	0.00	0.98	1.00	0.00
X_6	1.00	0.98	1.00	0.30	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 29

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.94	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.98	1.00	0.06	1.00	1.00	0.00
X_6	0.86	0.00	0.00	0.00	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 30

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	0.96	1.00	0.00	0.00	0.00
X_4	1.00	0.10	0.92	1.00	0.00	0.00
X_5	1.00	0.94	0.14	1.00	1.00	0.00
X_6	1.00	1.00	0.94	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 31

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.90	0.90	0.00	1.00	1.00	0.00
X_6	0.84	0.10	0.00	0.00	0.00	1.00
X_7	0.10	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.08	0.00	0.00	0.00	0.00	0.00

Case 32

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.06	1.00	1.00	0.00	0.00
X_5	1.00	0.94	0.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	0.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.3 Local heatmaps for Case $33-48$

Case 33

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.06	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.98	0.98	1.00	0.48	0.00
X_5	0.90	0.66	0.78	0.82	1.00	0.00
X_6	0.74	0.36	0.18	0.18	0.52	1.00
X_7	0.16	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 34

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.94	0.10	0.00	0.00	0.00	0.00
X_2	0.90	0.88	0.00	0.00	0.00	0.00
X_3	0.92	0.92	1.00	0.00	0.00	0.00
X_4	1.00	0.98	0.98	1.00	0.00	0.00
X_5	0.98	0.98	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.02	0.00	0.00	0.00	0.00
X_8	0.04	0.06	0.00	0.00	0.00	0.00
X_9	0.08	0.02	0.00	0.00	0.00	0.00
X_{10}	0.06	0.04	0.02	0.00	0.00	0.00

Case 35

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.96	1.00	1.00	0.00	0.00
X_5	0.96	0.64	0.76	0.90	1.00	0.00
X_6	0.80	0.38	0.24	0.10	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.08	0.02	0.00	0.00	0.00	0.00

Case 36

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.08	0.02	0.00	0.00	0.00
X_2	0.94	0.94	0.00	0.00	0.00	0.00
X_3	0.80	0.90	1.00	0.00	0.00	0.00
X_4	1.00	0.96	0.98	1.00	0.00	0.00
X_5	1.00	1.00	0.98	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.04	0.04	0.00	0.00	0.00	0.00
X_8	0.04	0.04	0.00	0.00	0.00	0.00
X_9	0.12	0.02	0.00	0.00	0.00	0.00
X_{10}	0.06	0.02	0.02	0.00	0.00	0.00

Case 37

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.76	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.94	0.88	0.24	0.94	1.00	0.00
X_6	0.86	0.12	0.00	0.06	0.00	1.00
X_7	0.06	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 38

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.70	0.96	0.02	0.00	0.00	0.00
X_2	0.90	0.40	0.20	0.00	0.00	0.00
X_3	0.96	0.62	0.90	0.00	0.00	0.00
X_4	1.00	1.00	0.90	1.00	0.00	0.00
X_5	1.00	1.00	0.98	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.10	0.00	0.00	0.00	0.00	0.00
X_8	0.08	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.18	0.02	0.00	0.00	0.00	0.00

Case 39

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.90	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.82	0.00
X_5	0.88	0.66	0.10	1.00	1.00	0.00
X_6	0.86	0.30	0.00	0.00	0.18	1.00
X_7	0.06	0.00	0.00	0.00	0.00	0.00
X_8	0.08	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.04	0.00	0.00	0.00	0.00

Case 40

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.54	1.00	0.58	0.00	0.00	0.00
X_2	0.60	0.70	0.74	0.00	0.00	0.00
X_3	0.92	0.72	0.80	0.00	0.00	0.00
X_4	1.00	0.48	0.20	1.00	0.00	0.00
X_5	1.00	1.00	0.68	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.24	0.02	0.00	0.00	0.00	0.00
X_8	0.32	0.02	0.00	0.00	0.00	0.00
X_9	0.22	0.04	0.00	0.00	0.00	0.00
X_{10}	0.16	0.02	0.00	0.00	0.00	0.00

Case 41

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.76	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.98	0.24	1.00	1.00	0.00
X_6	0.94	0.02	0.00	0.00	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 42

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.94	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.48	0.00	0.00	0.00
X_3	1.00	0.74	1.00	0.00	0.00	0.00
X_4	1.00	0.48	0.78	1.00	0.00	0.00
X_5	1.00	0.84	0.74	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 43

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.96	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.90	0.04	1.00	1.00	0.00
X_6	0.86	0.10	0.00	0.00	0.00	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 44

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.86	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.98	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.72	0.00	0.00
X_4	1.00	0.24	0.98	1.00	0.00	0.00
X_5	1.00	0.92	0.04	0.98	1.00	0.00
X_6	1.00	0.98	1.00	0.30	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 45

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.84	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.98	1.00	0.16	1.00	1.00	0.00
X_6	0.92	0.00	0.00	0.00	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 46

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.98	0.00	0.00	0.00
X_3	1.00	0.74	1.00	0.06	0.00	0.00
X_4	1.00	0.38	0.74	1.00	0.00	0.00
X_5	1.00	0.88	0.30	0.96	1.00	0.00
X_6	1.00	1.00	0.98	0.98	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 47

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.96	0.00	0.98	1.00	0.00
X_6	0.92	0.02	0.00	0.00	0.00	1.00
X_7	0.04	0.02	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 48

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.18	0.98	1.00	0.00	0.00
X_5	1.00	0.86	0.02	0.98	1.00	0.00
X_6	1.00	0.96	1.00	0.02	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.4 Local heatmaps for Case $49-64$

Case 49

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.12	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.98	0.98	1.00	0.38	0.00
X_5	0.92	0.58	0.78	0.88	1.00	0.00
X_6	0.80	0.42	0.10	0.12	0.62	1.00
X_7	0.06	0.00	0.02	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.10	0.02	0.00	0.00	0.00	0.00
X_{10}	0.08	0.00	0.00	0.00	0.00	0.00

Case 50

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.00	0.00	0.00	0.00	0.00
X_2	0.94	1.00	0.00	0.00	0.00	0.00
X_3	0.90	0.98	1.00	0.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.02	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 51

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.96	1.00	1.00	0.02	0.00
X_5	0.96	0.84	0.88	0.82	1.00	0.00
X_6	0.80	0.16	0.12	0.18	0.98	1.00
X_7	0.12	0.02	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.02	0.00	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 52

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.02	0.00	0.00	0.00	0.00
X_2	0.94	0.98	0.00	0.00	0.00	0.00
X_3	0.92	0.98	1.00	0.00	0.00	0.00
X_4	1.00	1.00	0.98	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.02	0.02	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 53

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.86	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	1.00	1.00	1.00	1.00	0.00
X_5	0.98	0.88	0.14	0.94	1.00	0.00
X_6	0.92	0.12	0.00	0.04	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 54

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.48	0.98	0.04	0.00	0.00	0.00
X_2	0.78	0.40	0.08	0.00	0.00	0.00
X_3	0.96	0.62	0.92	0.00	0.00	0.00
X_4	1.00	1.00	0.96	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.18	0.00	0.00	0.00	0.00	0.00
X_8	0.16	0.00	0.00	0.00	0.00	0.00
X_9	0.24	0.00	0.00	0.00	0.00	0.00
X_{10}	0.20	0.00	0.00	0.00	0.00	0.00

Case 55

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.78	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.04	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	0.92	0.00
X_5	0.94	0.76	0.22	0.96	1.00	0.00
X_6	0.80	0.24	0.00	0.00	0.08	1.00
X_7	0.08	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.06	0.00	0.00	0.00	0.00	0.00

Case 56

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.50	1.00	0.60	0.00	0.00	0.00
X_2	0.74	0.64	0.90	0.00	0.00	0.00
X_3	0.86	0.82	0.82	0.00	0.00	0.00
X_4	1.00	0.48	0.18	1.00	0.00	0.00
X_5	1.00	1.00	0.50	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.18	0.02	0.00	0.00	0.00	0.00
X_8	0.32	0.00	0.00	0.00	0.00	0.00
X_9	0.22	0.04	0.00	0.00	0.00	0.00
X_{10}	0.18	0.00	0.00	0.00	0.00	0.00

Case 57

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.76	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.98	0.24	1.00	1.00	0.00
X_6	0.98	0.02	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 58

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.88	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.50	0.00	0.00	0.00
X_3	1.00	0.72	1.00	0.00	0.00	0.00
X_4	1.00	0.40	0.76	1.00	0.00	0.00
X_5	1.00	1.00	0.74	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 59

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.98	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	0.98	0.98	0.02	1.00	1.00	0.00
X_6	0.94	0.02	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 60

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.96	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	0.98	1.00	0.84	0.00	0.00
X_4	1.00	0.12	0.96	1.00	0.00	0.00
X_5	1.00	0.94	0.04	0.96	1.00	0.00
X_6	1.00	1.00	1.00	0.20	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 61

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.78	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.22	1.00	1.00	0.00
X_6	1.00	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 62

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.98	0.00	0.00	0.00
X_3	1.00	0.84	1.00	0.00	0.00	0.00
X_4	1.00	0.22	0.68	1.00	0.00	0.00
X_5	1.00	0.94	0.34	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 63

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.98	0.00	1.00	1.00	0.00
X_6	0.96	0.02	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 64

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.02	1.00	1.00	0.00	0.00
X_5	1.00	0.98	0.00	0.96	1.00	0.00
X_6	1.00	1.00	1.00	0.04	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.5 Local heatmaps for Case $65-80$

Case 65

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.04	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.96	0.96	1.00	1.00	0.54	0.00
X_5	0.96	0.60	0.74	0.86	1.00	0.00
X_6	0.66	0.44	0.22	0.14	0.46	1.00
X_7	0.10	0.00	0.00	0.00	0.00	0.00
X_8	0.08	0.00	0.00	0.00	0.00	0.00
X_9	0.12	0.00	0.00	0.00	0.00	0.00
X_{10}	0.12	0.00	0.00	0.00	0.00	0.00

Case 66

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.96	0.04	0.00	0.00	0.00	0.00
X_2	0.88	0.94	0.02	0.00	0.00	0.00
X_3	0.90	0.94	0.94	0.00	0.00	0.00
X_4	0.94	0.96	1.00	1.00	0.00	0.00
X_5	0.98	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.12	0.04	0.02	0.00	0.00	0.00
X_8	0.08	0.02	0.02	0.00	0.00	0.00
X_9	0.10	0.02	0.00	0.00	0.00	0.00
X_{10}	0.04	0.04	0.00	0.00	0.00	0.00

Case 67

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.94	1.00	1.00	0.04	0.00
X_5	0.98	0.68	0.80	0.78	1.00	0.00
X_6	0.54	0.38	0.20	0.20	0.96	1.00
X_7	0.14	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.20	0.00	0.00	0.00	0.00	0.00

Case 68

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.06	0.00	0.00	0.00	0.00
X_2	0.96	0.92	0.00	0.00	0.00	0.00
X_3	0.86	0.94	1.00	0.00	0.00	0.00
X_4	1.00	0.94	0.98	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.04	0.00	0.00	0.00	0.00
X_8	0.02	0.02	0.00	0.00	0.00	0.00
X_9	0.06	0.02	0.00	0.00	0.00	0.00
X_{10}	0.02	0.06	0.02	0.00	0.00	0.00

Case 69

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.64	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.98	1.00	1.00	1.00	0.00
X_5	0.90	0.78	0.36	0.98	1.00	0.00
X_6	0.92	0.24	0.00	0.02	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.08	0.00	0.00	0.00	0.00	0.00
X_{10}	0.02	0.00	0.00	0.00	0.00	0.00

Case 70

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.58	0.98	0.06	0.00	0.00	0.00
X_2	0.72	0.44	0.22	0.00	0.00	0.00
X_3	0.92	0.68	0.92	0.00	0.00	0.00
X_4	1.00	0.88	0.80	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.22	0.00	0.00	0.00	0.00	0.00
X_8	0.26	0.00	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.26	0.02	0.00	0.00	0.00	0.00

Case 71

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.82	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.04	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.90	1.00	1.00	0.84	0.00
X_5	0.92	0.74	0.18	0.96	1.00	0.00
X_6	0.78	0.34	0.00	0.00	0.16	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.12	0.02	0.00	0.00	0.00	0.00
X_9	0.04	0.00	0.00	0.00	0.00	0.00
X_{10}	0.14	0.00	0.00	0.00	0.00	0.00

Case 72

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.52	1.00	0.58	0.00	0.00	0.00
X_2	0.70	0.56	0.92	0.00	0.00	0.00
X_3	0.90	0.68	0.62	0.00	0.00	0.00
X_4	1.00	0.72	0.26	1.00	0.00	0.00
X_5	1.00	1.00	0.62	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.32	0.02	0.00	0.00	0.00	0.00
X_8	0.24	0.00	0.00	0.00	0.00	0.00
X_9	0.20	0.00	0.00	0.00	0.00	0.00
X_{10}	0.14	0.02	0.00	0.00	0.00	0.00

Case 73

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.44	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.96	0.58	1.00	1.00	0.00
X_6	0.96	0.04	0.00	0.00	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 74

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.80	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.52	0.00	0.00	0.00
X_3	1.00	0.68	1.00	0.00	0.00	0.00
X_4	1.00	0.58	0.76	1.00	0.00	0.00
X_5	1.00	0.94	0.72	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 75

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.84	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.16	0.98	1.00	0.00
X_6	0.96	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 76

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.86	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.96	0.00	0.00	0.00
X_3	1.00	0.94	1.00	0.88	0.00	0.00
X_4	1.00	0.20	0.96	1.00	0.00	0.00
X_5	1.00	1.00	0.08	0.96	1.00	0.00
X_6	1.00	1.00	1.00	0.16	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 77

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.54	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.46	1.00	1.00	0.00
X_6	0.98	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 78

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.92	0.00	0.00	0.00
X_3	1.00	0.68	1.00	0.04	0.00	0.00
X_4	1.00	0.38	0.74	1.00	0.00	0.00
X_5	1.00	0.94	0.40	0.98	1.00	0.00
X_6	1.00	1.00	0.94	0.98	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 79

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.94	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.02	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	0.96	0.06	0.98	1.00	0.00
X_6	1.00	0.04	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 80

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	0.96	1.00	1.00	0.00	0.00
X_4	1.00	0.06	1.00	1.00	0.00	0.00
X_5	1.00	0.98	0.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	0.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.6 Local heatmaps for Case $81-96$

Case 81

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.06	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.98	1.00	1.00	0.62	0.00
X_5	0.90	0.68	0.72	0.90	1.00	0.00
X_6	0.86	0.34	0.26	0.10	0.38	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.12	0.00	0.00	0.00	0.00	0.00

Case 82

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.96	0.02	0.00	0.00	0.00	0.00
X_2	0.92	0.94	0.00	0.00	0.00	0.00
X_3	1.00	1.00	0.98	0.00	0.00	0.00
X_4	0.96	1.00	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.02	0.00	0.00	0.00	0.00
X_9	0.02	0.02	0.02	0.00	0.00	0.00
X_{10}	0.10	0.00	0.00	0.00	0.00	0.00

Case 83

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.04	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	0.98	0.96	1.00	1.00	0.08	0.00
X_5	0.90	0.78	0.82	0.86	1.00	0.00
X_6	0.60	0.26	0.18	0.10	0.92	1.00
X_7	0.12	0.00	0.00	0.00	0.00	0.00
X_8	0.06	0.00	0.00	0.00	0.00	0.00
X_9	0.16	0.00	0.00	0.00	0.00	0.00
X_{10}	0.18	0.00	0.00	0.00	0.00	0.00

Case 84

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.04	0.00	0.00	0.00	0.00
X_2	1.00	0.98	0.00	0.00	0.00	0.00
X_3	0.96	0.96	1.00	0.00	0.00	0.00
X_4	1.00	0.94	1.00	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.04	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.06	0.00	0.00	0.00	0.00
X_9	0.00	0.02	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 85

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.84	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.96	1.00	1.00	1.00	0.00
X_5	1.00	0.76	0.16	1.00	1.00	0.00
X_6	0.92	0.28	0.00	0.00	0.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 86

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.72	0.96	0.06	0.00	0.00	0.00
X_2	0.86	0.32	0.20	0.00	0.00	0.00
X_3	0.98	0.74	0.84	0.00	0.00	0.00
X_4	1.00	0.96	0.90	1.00	0.00	0.00
X_5	1.00	1.00	1.00	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.18	0.00	0.00	0.00	0.00	0.00
X_8	0.10	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.10	0.02	0.00	0.00	0.00	0.00

Case 87

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.72	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.04	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.98	1.00	1.00	0.90	0.00
X_5	0.98	0.84	0.28	0.94	1.00	0.00
X_6	0.78	0.18	0.00	0.02	0.10	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.08	0.00	0.00	0.00	0.00	0.00
X_9	0.06	0.00	0.00	0.00	0.00	0.00
X_{10}	0.08	0.00	0.00	0.00	0.00	0.00

Case 88

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.70	1.00	0.50	0.00	0.00	0.00
X_2	0.80	0.68	0.92	0.00	0.00	0.00
X_3	0.94	0.64	0.70	0.00	0.00	0.00
X_4	1.00	0.62	0.30	1.00	0.00	0.00
X_5	1.00	1.00	0.58	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.14	0.02	0.00	0.00	0.00	0.00
X_8	0.08	0.04	0.00	0.00	0.00	0.00
X_9	0.12	0.00	0.00	0.00	0.00	0.00
X_{10}	0.22	0.00	0.00	0.00	0.00	0.00

Case 89

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.32	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.68	1.00	1.00	0.00
X_6	1.00	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 90

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.82	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.42	0.00	0.00	0.00
X_3	1.00	0.80	1.00	0.00	0.00	0.00
X_4	1.00	0.40	0.76	1.00	0.00	0.00
X_5	1.00	0.98	0.82	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 91

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.90	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.10	1.00	1.00	0.00
X_6	1.00	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 92

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.88	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.98	0.00	0.00	0.00
X_3	1.00	1.00	1.00	0.84	0.00	0.00
X_4	1.00	0.16	0.92	1.00	0.00	0.00
X_5	1.00	0.96	0.10	0.96	1.00	0.00
X_6	1.00	1.00	1.00	0.20	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 93

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.44	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.56	1.00	1.00	0.00
X_6	1.00	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 94

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	0.92	0.00	0.00	0.00
X_3	1.00	0.72	1.00	0.00	0.00	0.00
X_4	1.00	0.32	0.74	1.00	0.00	0.00
X_5	1.00	0.96	0.34	1.00	1.00	0.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 95

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.96	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	1.00	1.00	1.00	1.00	0.00
X_5	1.00	1.00	0.04	1.00	1.00	0.00
X_6	1.00	0.00	0.00	0.00	0.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 96

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	0.00	0.00	0.00
X_3	1.00	1.00	1.00	1.00	0.00	0.00
X_4	1.00	0.06	1.00	1.00	0.00	0.00
X_5	1.00	0.96	0.00	1.00	1.00	0.00
X_6	1.00	0.98	1.00	0.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.7 Local heatmaps for Case $1-16$ using a median threshold strategy

Case 1

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.32	0.00	0.10	0.22
X_2	1.00	1.00	1.00	1.00	0.82	0.48
X_3	1.00	1.00	1.00	1.00	1.00	0.78
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.64	0.80	0.94	1.00	1.00	1.00
X_6	0.36	0.20	0.74	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.04	0.10
X_8	0.00	0.00	0.00	0.00	0.02	0.16
X_9	0.00	0.00	0.00	0.00	0.02	0.14
X_{10}	0.02	0.00	0.00	0.00	0.00	0.12

Case 2

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.82	0.00	0.10	0.16	0.18	0.22
X_2	0.72	0.98	0.42	0.52	0.44	0.58
X_3	0.66	1.00	1.00	0.82	0.84	0.96
X_4	0.82	1.00	1.00	1.00	0.98	1.00
X_5	0.98	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.18	0.16	0.12	0.04
X_8	0.00	0.00	0.08	0.10	0.18	0.02
X_9	0.00	0.02	0.14	0.18	0.16	0.06
X_{10}	0.00	0.00	0.08	0.08	0.10	0.12

Case 3

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.22	0.00	0.10	0.24
X_2	1.00	1.00	1.00	0.98	0.56	0.38
X_3	1.00	1.00	1.00	1.00	1.00	0.90
X_4	0.92	1.00	1.00	1.00	1.00	1.00
X_5	0.66	0.74	1.00	1.00	1.00	1.00
X_6	0.28	0.26	0.78	0.98	1.00	1.00
X_7	0.04	0.00	0.00	0.00	0.10	0.10
X_8	0.02	0.00	0.00	0.00	0.10	0.14
X_9	0.02	0.00	0.00	0.02	0.06	0.12
X_{10}	0.06	0.00	0.00	0.02	0.10	0.12

Case 4

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.90	0.02	0.20	0.46	0.28	0.22
X_2	0.66	1.00	0.54	0.44	0.54	0.48
X_3	0.42	1.00	1.00	0.56	0.64	0.84
X_4	1.00	0.96	1.00	1.00	0.64	0.90
X_5	1.00	1.00	1.00	1.00	1.00	0.98
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.08	0.12	0.18	0.12
X_8	0.00	0.00	0.04	0.08	0.20	0.16
X_9	0.00	0.00	0.06	0.16	0.32	0.14
X_{10}	0.02	0.02	0.08	0.20	0.20	0.18

Case 5

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.14	0.18
X_2	1.00	1.00	1.00	1.00	0.86	0.68
X_3	0.98	1.00	1.00	1.00	1.00	0.96
X_4	0.96	0.98	1.00	1.00	1.00	1.00
X_5	0.68	0.60	0.90	1.00	1.00	1.00
X_6	0.34	0.38	0.10	1.00	1.00	1.00
X_7	0.02	0.02	0.00	0.00	0.00	0.02
X_8	0.00	0.02	0.00	0.00	0.00	0.10
X_9	0.02	0.00	0.00	0.00	0.00	0.02
X_{10}	0.00	0.00	0.00	0.00	0.00	0.04

Case 6

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.36	1.00	0.36	0.48	0.64	0.90
X_2	0.52	0.50	0.68	0.56	0.46	0.38
X_3	0.98	0.60	0.98	0.94	0.82	0.68
X_4	1.00	0.94	0.98	1.00	0.98	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.02	0.00
X_8	0.06	0.00	0.00	0.00	0.04	0.00
X_9	0.04	0.00	0.00	0.02	0.02	0.04
X_{10}	0.04	0.00	0.00	0.00	0.02	0.02

Case 7

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.02	0.16	0.06
X_2	1.00	1.00	1.00	1.00	0.82	0.78
X_3	0.96	1.00	1.00	1.00	1.00	0.98
X_4	0.92	0.98	1.00	1.00	1.00	1.00
X_5	0.54	0.74	0.96	1.00	1.00	1.00
X_6	0.50	0.22	0.04	0.98	1.00	1.00
X_7	0.02	0.02	0.00	0.00	0.02	0.00
X_8	0.04	0.02	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.08
X_{10}	0.02	0.02	0.00	0.00	0.00	0.10

Case 8

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.28	1.00	0.88	0.36	0.98	1.00
X_2	0.58	0.60	0.98	0.56	0.30	0.96
X_3	0.58	0.58	0.96	0.98	0.64	0.38
X_4	1.00	0.74	0.56	1.00	0.82	0.38
X_5	1.00	1.00	0.62	1.00	1.00	0.98
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.14	0.02	0.00	0.02	0.02	0.14
X_8	0.14	0.02	0.00	0.00	0.04	0.12
X_9	0.10	0.02	0.00	0.06	0.06	0.02
X_{10}	0.18	0.04	0.00	0.02	0.14	0.02

Case 9

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.04	0.14
X_2	1.00	1.00	1.00	1.00	0.96	0.86
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.78	0.96	1.00	1.00	1.00	1.00
X_6	0.26	0.04	0.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 10

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.04	0.04
X_2	0.92	1.00	1.00	1.00	0.96	0.94
X_3	0.28	0.84	1.00	1.00	1.00	1.00
X_4	0.84	0.38	1.00	1.00	1.00	1.00
X_5	0.96	0.78	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.02
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 11

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.04	0.00	0.10
X_2	1.00	1.00	1.00	1.00	1.00	0.90
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.96	1.00	1.00	1.00	1.00	1.00
X_5	0.56	0.90	1.00	1.00	1.00	1.00
X_6	0.36	0.06	0.00	0.96	1.00	1.00
X_7	0.06	0.02	0.00	0.00	0.00	0.00
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.02	0.00	0.00	0.00	0.00
X_{10}	0.04	0.00	0.00	0.00	0.00	0.00

Case 12

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.26	0.02
X_2	0.96	1.00	1.00	1.00	0.72	0.98
X_3	0.18	1.00	1.00	1.00	1.00	0.98
X_4	0.98	0.14	1.00	1.00	1.00	1.00
X_5	0.92	0.90	1.00	1.00	1.00	1.00
X_6	0.96	0.96	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.02	0.02
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 13

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.02	0.18
X_2	1.00	1.00	1.00	1.00	0.98	0.82
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.86	0.96	1.00	1.00	1.00	1.00
X_6	0.16	0.04	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 14

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.04
X_2	0.92	1.00	1.00	1.00	1.00	0.96
X_3	0.28	0.84	1.00	1.00	1.00	1.00
X_4	0.82	0.36	1.00	1.00	1.00	1.00
X_5	0.98	0.80	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 15

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.08	0.00	0.16
X_2	1.00	1.00	1.00	1.00	1.00	0.82
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.96	1.00	1.00	1.00	1.00	1.00
X_5	0.70	0.80	1.00	1.00	1.00	1.00
X_6	0.30	0.18	0.00	0.92	1.00	1.00
X_7	0.02	0.02	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.02	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.02	0.00	0.00	0.00	0.02

Case 16

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.18	0.00	0.22	0.00
X_2	0.94	1.00	1.00	1.00	0.78	1.00
X_3	0.10	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.96	0.96	0.82	1.00	1.00	1.00
X_6	1.00	0.94	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.8 Local heatmaps for Case $33-48$ using a median threshold strategy

Case 33

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.32	0.08	0.18	0.30
X_2	1.00	1.00	1.00	0.92	0.46	0.40
X_3	1.00	1.00	1.00	1.00	0.96	0.74
X_4	0.94	0.98	1.00	1.00	1.00	1.00
X_5	0.66	0.66	0.94	1.00	1.00	1.00
X_6	0.38	0.36	0.70	0.98	1.00	1.00
X_7	0.00	0.00	0.00	0.02	0.04	0.18
X_8	0.02	0.00	0.02	0.00	0.08	0.12
X_9	0.00	0.00	0.00	0.00	0.18	0.16
X_{10}	0.00	0.00	0.02	0.00	0.10	0.10

Case 34

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.74	0.10	0.18	0.22	0.24	0.14
X_2	0.58	0.88	0.26	0.44	0.36	0.46
X_3	0.76	0.92	1.00	0.48	0.76	0.84
X_4	0.86	0.98	1.00	1.00	0.96	1.00
X_5	0.94	0.98	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.02	0.12	0.24	0.06	0.12
X_8	0.02	0.06	0.20	0.16	0.18	0.06
X_9	0.02	0.02	0.08	0.38	0.26	0.20
X_{10}	0.00	0.04	0.16	0.08	0.18	0.18

Case 35

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.10	0.02	0.12	0.22
X_2	1.00	1.00	1.00	0.98	0.66	0.36
X_3	1.00	1.00	1.00	1.00	1.00	0.92
X_4	0.90	0.96	1.00	1.00	1.00	1.00
X_5	0.78	0.64	1.00	1.00	1.00	1.00
X_6	0.26	0.38	0.90	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.06	0.16
X_8	0.00	0.00	0.00	0.00	0.06	0.14
X_9	0.04	0.00	0.00	0.00	0.06	0.12
X_{10}	0.02	0.02	0.00	0.00	0.04	0.08

Case 36

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.96	0.08	0.22	0.50	0.32	0.24
X_2	0.66	0.94	0.32	0.38	0.54	0.42
X_3	0.36	0.90	1.00	0.36	0.64	0.80
X_4	1.00	0.96	0.98	1.00	0.50	0.84
X_5	1.00	1.00	0.98	1.00	1.00	0.92
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.04	0.12	0.10	0.18	0.18
X_8	0.00	0.04	0.12	0.14	0.26	0.12
X_9	0.00	0.02	0.04	0.26	0.36	0.16
X_{10}	0.02	0.02	0.22	0.28	0.20	0.32

Case 37

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.08	0.28
X_2	1.00	1.00	1.00	1.00	0.92	0.62
X_3	0.98	1.00	1.00	1.00	1.00	0.92
X_4	0.94	1.00	1.00	1.00	1.00	1.00
X_5	0.74	0.88	0.92	1.00	1.00	1.00
X_6	0.34	0.12	0.08	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.02
X_8	0.00	0.00	0.00	0.00	0.00	0.02
X_9	0.00	0.00	0.00	0.00	0.00	0.06
X_{10}	0.00	0.00	0.00	0.00	0.00	0.08

Case 38

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.36	0.96	0.36	0.36	0.68	0.80
X_2	0.56	0.40	0.66	0.72	0.52	0.42
X_3	0.90	0.62	0.98	0.90	0.68	0.48
X_4	1.00	1.00	1.00	1.00	0.94	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.02	0.08
X_8	0.02	0.00	0.00	0.00	0.12	0.12
X_9	0.04	0.00	0.00	0.02	0.02	0.06
X_{10}	0.10	0.02	0.00	0.00	0.02	0.06

Case 39

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.06	0.12	0.14
X_2	1.00	1.00	1.00	1.00	0.88	0.80
X_3	0.98	1.00	1.00	1.00	1.00	1.00
X_4	0.90	1.00	1.00	1.00	1.00	1.00
X_5	0.70	0.66	0.94	1.00	1.00	1.00
X_6	0.38	0.30	0.06	0.94	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.04
X_8	0.04	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.04	0.00	0.00	0.00	0.02

Case 40

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.38	1.00	0.90	0.38	0.98	1.00
X_2	0.48	0.70	0.94	0.54	0.26	0.96
X_3	0.74	0.72	0.94	0.98	0.64	0.40
X_4	1.00	0.48	0.34	1.00	0.82	0.48
X_5	1.00	1.00	0.88	1.00	1.00	0.92
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.14	0.02	0.00	0.02	0.04	0.04
X_8	0.16	0.02	0.00	0.02	0.04	0.06
X_9	0.04	0.04	0.00	0.04	0.12	0.08
X_{10}	0.06	0.02	0.00	0.02	0.10	0.06

Case 41

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.02	0.02
X_2	1.00	1.00	1.00	1.00	0.98	0.98
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.92	0.98	1.00	1.00	1.00	1.00
X_6	0.08	0.02	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.04	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 42

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.94	0.00	0.00	0.04	0.04
X_2	0.74	1.00	1.00	1.00	0.96	0.96
X_3	0.32	0.74	1.00	1.00	1.00	1.00
X_4	0.94	0.48	1.00	1.00	1.00	1.00
X_5	1.00	0.84	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 43

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.88	0.90	1.00	1.00	1.00	1.00
X_6	0.12	0.10	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 44

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.86	0.00	0.04	0.08	0.00
X_2	0.96	1.00	1.00	0.96	0.90	1.00
X_3	0.14	1.00	1.00	1.00	1.00	1.00
X_4	0.98	0.24	1.00	1.00	1.00	1.00
X_5	0.92	0.92	1.00	1.00	1.00	1.00
X_6	1.00	0.98	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.02	0.00

Case 45

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.02	0.00	0.02
X_2	1.00	1.00	1.00	1.00	1.00	0.98
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.94	1.00	1.00	1.00	1.00	1.00
X_6	0.06	0.00	0.00	0.98	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.02	0.00

Case 46

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.02	0.02
X_2	0.78	1.00	1.00	1.00	0.98	0.98
X_3	0.40	0.74	1.00	1.00	1.00	1.00
X_4	0.84	0.38	1.00	1.00	1.00	1.00
X_5	0.98	0.88	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.02	0.00

Case 47

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.06	0.00	0.02
X_2	1.00	1.00	1.00	1.00	1.00	0.98
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.82	0.96	1.00	1.00	1.00	1.00
X_6	0.18	0.02	0.00	0.94	1.00	1.00
X_7	0.00	0.02	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 48

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.08	0.00	0.06	0.00
X_2	0.88	1.00	1.00	1.00	0.94	1.00
X_3	0.18	1.00	1.00	1.00	1.00	1.00
X_4	1.00	0.18	1.00	1.00	1.00	1.00
X_5	0.94	0.86	0.92	1.00	1.00	1.00
X_6	1.00	0.96	1.00	1.00	1.00	1.00
X_7	0.00	0.02	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.9 Local heatmaps for Case $49-64$ using a median threshold strategy

Case 49

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.54	0.04	0.14	0.24
X_2	1.00	1.00	1.00	0.96	0.68	0.44
X_3	1.00	1.00	1.00	1.00	1.00	0.72
X_4	0.94	0.98	1.00	1.00	1.00	0.96
X_5	0.66	0.58	0.88	1.00	1.00	1.00
X_6	0.38	0.42	0.56	1.00	1.00	1.00
X_7	0.00	0.00	0.02	0.00	0.06	0.00
X_8	0.00	0.00	0.00	0.00	0.02	0.12
X_9	0.00	0.02	0.00	0.00	0.02	0.12
X_{10}	0.02	0.00	0.00	0.00	0.08	0.20

Case 50

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.88	0.00	0.12	0.16	0.22	0.16
X_2	0.62	1.00	0.40	0.50	0.36	0.48
X_3	0.60	0.98	1.00	0.66	0.74	0.88
X_4	0.90	1.00	1.00	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.16	0.32	0.18	0.08
X_8	0.00	0.02	0.10	0.10	0.20	0.08
X_9	0.00	0.00	0.10	0.08	0.18	0.14
X_{10}	0.00	0.00	0.12	0.18	0.12	0.18

Case 51

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.18	0.04	0.10	0.12
X_2	1.00	1.00	1.00	0.96	0.72	0.46
X_3	0.98	1.00	1.00	1.00	1.00	0.90
X_4	0.96	0.96	1.00	1.00	1.00	1.00
X_5	0.60	0.84	0.98	1.00	1.00	1.00
X_6	0.42	0.16	0.84	1.00	1.00	1.00
X_7	0.00	0.02	0.00	0.00	0.02	0.16
X_8	0.00	0.00	0.00	0.00	0.06	0.16
X_9	0.02	0.02	0.00	0.00	0.04	0.12
X_{10}	0.02	0.00	0.00	0.00	0.06	0.08

Case 52

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.94	0.02	0.18	0.64	0.16	0.28
X_2	0.52	0.98	0.42	0.34	0.46	0.20
X_3	0.54	0.98	1.00	0.52	0.68	0.92
X_4	1.00	1.00	1.00	1.00	0.72	0.94
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.02	0.20	0.28	0.12
X_8	0.00	0.02	0.14	0.12	0.22	0.22
X_9	0.00	0.00	0.14	0.14	0.30	0.12
X_{10}	0.00	0.00	0.10	0.04	0.18	0.20

Case 53

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.10	0.30
X_2	1.00	1.00	1.00	1.00	0.90	0.58
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.88	1.00	1.00	1.00	1.00	1.00
X_5	0.74	0.88	0.92	1.00	1.00	1.00
X_6	0.34	0.12	0.08	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.02
X_8	0.00	0.00	0.00	0.00	0.00	0.02
X_9	0.02	0.00	0.00	0.00	0.00	0.04
X_{10}	0.00	0.00	0.00	0.00	0.00	0.04

Case 54

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.24	0.98	0.36	0.56	0.78	0.88
X_2	0.58	0.40	0.64	0.48	0.44	0.50
X_3	0.94	0.62	1.00	0.96	0.74	0.54
X_4	1.00	1.00	1.00	1.00	1.00	0.98
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.06	0.00	0.00	0.00	0.02	0.02
X_8	0.02	0.00	0.00	0.00	0.00	0.04
X_9	0.10	0.00	0.00	0.00	0.02	0.02
X_{10}	0.06	0.00	0.00	0.00	0.00	0.02

Case 55

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.02	0.04	0.12
X_2	1.00	1.00	1.00	0.98	0.96	0.86
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.92	1.00	1.00	1.00	1.00	1.00
X_5	0.68	0.76	0.90	1.00	1.00	1.00
X_6	0.38	0.24	0.10	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.02

Case 56

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.34	1.00	0.80	0.36	1.00	1.00
X_2	0.52	0.64	0.98	0.64	0.40	0.96
X_3	0.74	0.82	0.94	0.98	0.64	0.52
X_4	1.00	0.48	0.54	1.00	0.76	0.46
X_5	1.00	1.00	0.74	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.02	0.00	0.02	0.02	0.00
X_8	0.12	0.00	0.00	0.00	0.04	0.02
X_9	0.10	0.04	0.00	0.00	0.06	0.04
X_{10}	0.10	0.00	0.00	0.00	0.08	0.00

Case 57

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.02
X_2	1.00	1.00	1.00	1.00	1.00	0.98
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.96	0.98	1.00	1.00	1.00	1.00
X_6	0.04	0.02	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 58

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.88	0.00	0.00	0.00	0.02
X_2	0.76	1.00	1.00	1.00	1.00	0.98
X_3	0.32	0.72	1.00	1.00	1.00	1.00
X_4	0.92	0.40	1.00	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 59

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.92	0.98	1.00	1.00	1.00	1.00
X_6	0.06	0.02	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 60

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.96	0.00	0.02	0.04	0.00
X_2	0.96	1.00	1.00	0.98	0.96	1.00
X_3	0.18	0.98	1.00	1.00	1.00	1.00
X_4	0.98	0.12	1.00	1.00	1.00	1.00
X_5	0.88	0.94	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 61

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.02
X_2	1.00	1.00	1.00	1.00	1.00	0.98
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.96	1.00	1.00	1.00	1.00	1.00
X_6	0.04	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 62

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	0.74	1.00	1.00	1.00	1.00	1.00
X_3	0.38	0.84	1.00	1.00	1.00	1.00
X_4	0.88	0.22	1.00	1.00	1.00	1.00
X_5	1.00	0.94	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 63

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.94	0.98	1.00	1.00	1.00	1.00
X_6	0.06	0.02	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 64

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.08	0.00
X_2	0.80	1.00	1.00	1.00	0.92	1.00
X_3	0.20	1.00	1.00	1.00	1.00	1.00
X_4	1.00	0.02	1.00	1.00	1.00	1.00
X_5	1.00	0.98	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.10 Local heatmaps for Case $65-80$ using a median threshold strategy

Case 65

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.26	0.02	0.06	0.20
X_2	1.00	1.00	1.00	0.98	0.64	0.40
X_3	1.00	1.00	1.00	1.00	1.00	0.74
X_4	0.86	0.96	1.00	1.00	1.00	0.96
X_5	0.86	0.60	0.94	1.00	1.00	1.00
X_6	0.22	0.44	0.78	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.02	0.04
X_8	0.00	0.00	0.00	0.00	0.04	0.26
X_9	0.02	0.00	0.00	0.00	0.08	0.22
X_{10}	0.02	0.00	0.02	0.00	0.16	0.18

Case 66

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.86	0.04	0.14	0.20	0.18	0.18
X_2	0.58	0.94	0.48	0.28	0.26	0.46
X_3	0.66	0.94	1.00	0.62	0.72	0.70
X_4	0.84	0.96	1.00	1.00	1.00	0.98
X_5	0.92	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.06	0.04	0.18	0.24	0.26	0.16
X_8	0.04	0.02	0.10	0.28	0.28	0.18
X_9	0.04	0.02	0.04	0.20	0.16	0.12
X_{10}	0.00	0.04	0.06	0.18	0.18	0.22

Case 67

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.12	0.02	0.10	0.22
X_2	1.00	1.00	1.00	0.92	0.68	0.54
X_3	0.98	1.00	1.00	1.00	1.00	0.86
X_4	0.94	0.94	1.00	1.00	1.00	1.00
X_5	0.84	0.68	0.98	1.00	1.00	1.00
X_6	0.16	0.38	0.90	1.00	1.00	1.00
X_7	0.04	0.00	0.00	0.00	0.04	0.10
X_8	0.02	0.00	0.00	0.02	0.08	0.08
X_9	0.02	0.00	0.00	0.02	0.06	0.12
X_{10}	0.00	0.00	0.00	0.02	0.04	0.08

Case 68

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.88	0.06	0.16	0.44	0.32	0.26
X_2	0.56	0.92	0.28	0.32	0.34	0.26
X_3	0.56	0.94	1.00	0.38	0.62	0.82
X_4	1.00	0.94	1.00	1.00	0.76	0.80
X_5	1.00	1.00	1.00	1.00	1.00	0.94
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.04	0.22	0.24	0.26	0.28
X_8	0.00	0.02	0.06	0.24	0.26	0.28
X_9	0.00	0.02	0.12	0.16	0.24	0.22
X_{10}	0.00	0.06	0.16	0.22	0.28	0.24

Case 69

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.98	0.00	0.16	0.28
X_2	1.00	1.00	1.00	1.00	0.84	0.70
X_3	1.00	1.00	1.00	1.00	1.00	0.88
X_4	0.90	0.98	1.00	1.00	1.00	1.00
X_5	0.66	0.78	0.96	1.00	1.00	1.00
X_6	0.44	0.24	0.06	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.06
X_8	0.00	0.00	0.00	0.00	0.00	0.06
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.02

Case 70

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.34	0.98	0.42	0.56	0.62	0.80
X_2	0.38	0.44	0.66	0.44	0.44	0.48
X_3	0.84	0.68	0.96	0.88	0.74	0.54
X_4	1.00	0.88	0.96	1.00	0.92	0.96
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.00	0.00	0.04	0.08	0.08
X_8	0.16	0.00	0.00	0.06	0.10	0.08
X_9	0.02	0.00	0.00	0.02	0.04	0.04
X_{10}	0.18	0.02	0.00	0.00	0.06	0.02

Case 71

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.06	0.08
X_2	1.00	1.00	1.00	1.00	0.92	0.84
X_3	0.96	1.00	1.00	1.00	1.00	1.00
X_4	1.00	0.90	1.00	1.00	1.00	1.00
X_5	0.66	0.74	0.92	1.00	1.00	1.00
X_6	0.36	0.34	0.08	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.04
X_8	0.02	0.02	0.00	0.00	0.00	0.02
X_9	0.00	0.00	0.00	0.00	0.02	0.02
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 72

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.32	1.00	0.80	0.40	1.00	0.98
X_2	0.52	0.56	0.98	0.60	0.30	1.00
X_3	0.76	0.68	0.78	0.96	0.58	0.38
X_4	1.00	0.72	0.54	1.00	0.84	0.52
X_5	1.00	1.00	0.92	1.00	1.00	0.92
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.16	0.02	0.00	0.00	0.06	0.10
X_8	0.08	0.00	0.00	0.00	0.08	0.04
X_9	0.08	0.00	0.00	0.02	0.06	0.02
X_{10}	0.08	0.02	0.00	0.02	0.08	0.04

Case 73

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.94	0.00	0.00	0.04
X_2	1.00	1.00	1.00	1.00	1.00	0.96
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	0.90	0.96	1.00	1.00	1.00	1.00
X_6	0.12	0.04	0.06	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 74

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.80	0.00	0.00	0.02	0.00
X_2	0.74	1.00	1.00	1.00	0.98	1.00
X_3	0.40	0.68	1.00	1.00	1.00	1.00
X_4	0.86	0.58	1.00	1.00	1.00	1.00
X_5	1.00	0.94	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 75

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.94	1.00	1.00	1.00	1.00	1.00
X_6	0.06	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 76

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.86	0.00	0.02	0.06	0.00
X_2	0.94	1.00	1.00	0.98	0.94	1.00
X_3	0.22	0.94	1.00	1.00	1.00	1.00
X_4	0.96	0.20	1.00	1.00	1.00	1.00
X_5	0.88	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 77

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.96	0.00	0.00	0.02
X_2	1.00	1.00	1.00	1.00	1.00	0.98
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.96	1.00	1.00	1.00	1.00	1.00
X_5	0.98	1.00	1.00	1.00	1.00	1.00
X_6	0.06	0.00	0.04	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 78

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.02	0.04
X_2	0.58	1.00	1.00	1.00	0.98	0.96
X_3	0.50	0.68	1.00	1.00	1.00	1.00
X_4	0.92	0.38	1.00	1.00	1.00	1.00
X_5	1.00	0.94	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 79

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.02	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.96	0.96	1.00	1.00	1.00	1.00
X_6	0.04	0.04	0.00	0.98	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 80

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.02	0.00	0.02	0.00
X_2	0.82	1.00	1.00	1.00	0.98	1.00
X_3	0.22	0.96	1.00	1.00	1.00	1.00
X_4	0.98	0.06	1.00	1.00	1.00	1.00
X_5	1.00	0.98	0.98	1.00	1.00	1.00
X_6	0.98	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

B.11 Local heatmaps for Case $81-96$ using a median threshold strategy

Case 81

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.36	0.00	0.12	0.34
X_2	1.00	1.00	1.00	1.00	0.62	0.36
X_3	0.98	1.00	1.00	1.00	1.00	0.70
X_4	0.94	0.98	1.00	1.00	1.00	0.92
X_5	0.68	0.68	0.98	1.00	1.00	1.00
X_6	0.32	0.34	0.66	1.00	1.00	1.00
X_7	0.02	0.00	0.00	0.00	0.10	0.14
X_8	0.00	0.00	0.00	0.00	0.08	0.22
X_9	0.02	0.00	0.00	0.00	0.06	0.22
X_{10}	0.04	0.00	0.00	0.00	0.02	0.10

Case 82

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.82	0.02	0.20	0.20	0.14	0.28
X_2	0.70	0.94	0.28	0.36	0.36	0.36
X_3	0.70	1.00	1.00	0.72	0.68	0.90
X_4	0.76	1.00	1.00	1.00	0.96	1.00
X_5	0.98	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.08	0.16	0.26	0.20
X_8	0.02	0.02	0.12	0.20	0.20	0.08
X_9	0.00	0.02	0.22	0.18	0.22	0.12
X_{10}	0.02	0.00	0.10	0.18	0.18	0.06

Case 83

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.10	0.02	0.08	0.12
X_2	1.00	1.00	1.00	0.96	0.70	0.54
X_3	1.00	1.00	1.00	1.00	1.00	0.96
X_4	0.94	0.96	1.00	1.00	1.00	0.98
X_5	0.72	0.78	1.00	1.00	1.00	1.00
X_6	0.20	0.26	0.90	1.00	1.00	1.00
X_7	0.06	0.00	0.00	0.00	0.08	0.06
X_8	0.00	0.00	0.00	0.00	0.04	0.14
X_9	0.04	0.00	0.00	0.02	0.02	0.12
X_{10}	0.04	0.00	0.00	0.00	0.08	0.08

Case 84

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.88	0.04	0.20	0.50	0.24	0.28
X_2	0.62	0.98	0.36	0.52	0.48	0.26
X_3	0.50	0.96	1.00	0.36	0.70	0.82
X_4	1.00	0.94	1.00	1.00	0.80	0.88
X_5	1.00	1.00	1.00	1.00	1.00	0.98
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.10	0.14	0.18	0.24
X_8	0.00	0.06	0.08	0.16	0.20	0.24
X_9	0.00	0.02	0.08	0.12	0.24	0.16
X_{10}	0.00	0.00	0.18	0.20	0.16	0.14

Case 85

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.08	0.14
X_2	1.00	1.00	1.00	1.00	0.92	0.82
X_3	0.96	1.00	1.00	1.00	1.00	0.92
X_4	0.92	0.96	1.00	1.00	1.00	1.00
X_5	0.74	0.76	0.94	1.00	1.00	1.00
X_6	0.38	0.28	0.06	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.02
X_8	0.00	0.00	0.00	0.00	0.00	0.06
X_9	0.00	0.00	0.00	0.00	0.00	0.02
X_{10}	0.00	0.00	0.00	0.00	0.00	0.02

Case 86

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.44	0.96	0.34	0.52	0.76	0.84
X_2	0.56	0.32	0.76	0.52	0.50	0.50
X_3	0.88	0.74	0.98	0.90	0.64	0.58
X_4	1.00	0.96	0.92	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.08	0.00	0.00	0.02	0.04	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.02
X_9	0.02	0.00	0.00	0.02	0.02	0.04
X_{10}	0.00	0.02	0.00	0.02	0.04	0.02

Case 87

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.96	0.02	0.06	0.12
X_2	1.00	1.00	1.00	1.00	0.94	0.82
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	0.98	1.00	1.00	1.00	1.00
X_5	0.76	0.84	0.90	1.00	1.00	1.00
X_6	0.24	0.18	0.14	0.98	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.02	0.00	0.00	0.00	0.00	0.02
X_9	0.00	0.00	0.00	0.00	0.00	0.04
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 88

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	0.38	1.00	0.90	0.40	1.00	1.00
X_2	0.50	0.68	0.98	0.56	0.34	0.98
X_3	0.86	0.64	0.88	0.96	0.78	0.52
X_4	1.00	0.62	0.46	1.00	0.80	0.46
X_5	1.00	1.00	0.78	1.00	1.00	0.98
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.04	0.02	0.00	0.06	0.04	0.00
X_8	0.06	0.04	0.00	0.00	0.00	0.02
X_9	0.02	0.00	0.00	0.02	0.02	0.00
X_{10}	0.14	0.00	0.00	0.00	0.02	0.04

Case 89

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	0.98	1.00	1.00	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	0.02	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 90

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.82	0.00	0.00	0.00	0.00
X_2	0.72	1.00	1.00	1.00	1.00	1.00
X_3	0.34	0.80	1.00	1.00	1.00	1.00
X_4	0.94	0.40	1.00	1.00	1.00	1.00
X_5	1.00	0.98	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 91

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	0.98	1.00	1.00	1.00	1.00	1.00
X_6	0.02	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 92

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	0.88	0.00	0.00	0.02	0.00
X_2	0.92	1.00	1.00	1.00	0.98	1.00
X_3	0.18	1.00	1.00	1.00	1.00	1.00
X_4	0.90	0.16	1.00	1.00	1.00	1.00
X_5	1.00	0.96	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 93

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	0.00	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 94

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	0.46	1.00	1.00	1.00	1.00	1.00
X_3	0.58	0.72	1.00	1.00	1.00	1.00
X_4	0.96	0.32	1.00	1.00	1.00	1.00
X_5	1.00	0.96	1.00	1.00	1.00	1.00
X_6	1.00	1.00	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 95

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	1.00	0.00	0.00	0.00
X_2	1.00	1.00	1.00	1.00	1.00	1.00
X_3	1.00	1.00	1.00	1.00	1.00	1.00
X_4	1.00	1.00	1.00	1.00	1.00	1.00
X_5	1.00	1.00	1.00	1.00	1.00	1.00
X_6	0.00	0.00	0.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Case 96

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	1.00	1.00	0.00	0.00	0.00	0.00
X_2	0.78	1.00	1.00	1.00	1.00	1.00
X_3	0.22	1.00	1.00	1.00	1.00	1.00
X_4	1.00	0.06	1.00	1.00	1.00	1.00
X_5	1.00	0.96	1.00	1.00	1.00	1.00
X_6	1.00	0.98	1.00	1.00	1.00	1.00
X_7	0.00	0.00	0.00	0.00	0.00	0.00
X_8	0.00	0.00	0.00	0.00	0.00	0.00
X_9	0.00	0.00	0.00	0.00	0.00	0.00
X_{10}	0.00	0.00	0.00	0.00	0.00	0.00

Appendix C: R-functions for LDsplit as introduced in Chapter 3

This appendix contains all written R-functions to fit LDsplit as introduced in Chapter 3 and to classify observations using a fitted LDsplit ensemble. For each subsection of the appendix a diagram is provided to summarise the interaction of the functions.

Appendix C.1 contains all functions required to fit **Random LDsplit** as described in Section 3.2 when an **SVM** is used as base classifier.

Appendix C.2 contains additional functions required to fit **Random LDsplit** as described in Section 3.2 when an **SVM** is used as base classifier and functions run **parallel**.

Appendix C.3 contains additional functions required to fit **Conditional LDsplit** as described in Section 3.3.3 when an **SVM** is used as base classifier.

Appendix C.4 contains additional functions required to fit **Conditional LDsplit** as described in Section 3.3.3 when an **SVM** is used as base classifier and functions run **parallel**.

Appendix C.5 contains additional functions required to fit **Random LDsplit** as described in Section 3.2 when a **decision tree** is used as base classifier.

Appendix C.6 contains additional functions required to fit **Conditional LDsplit** as described in Section 3.3.3 when a **decision tree** is used as base classifier.

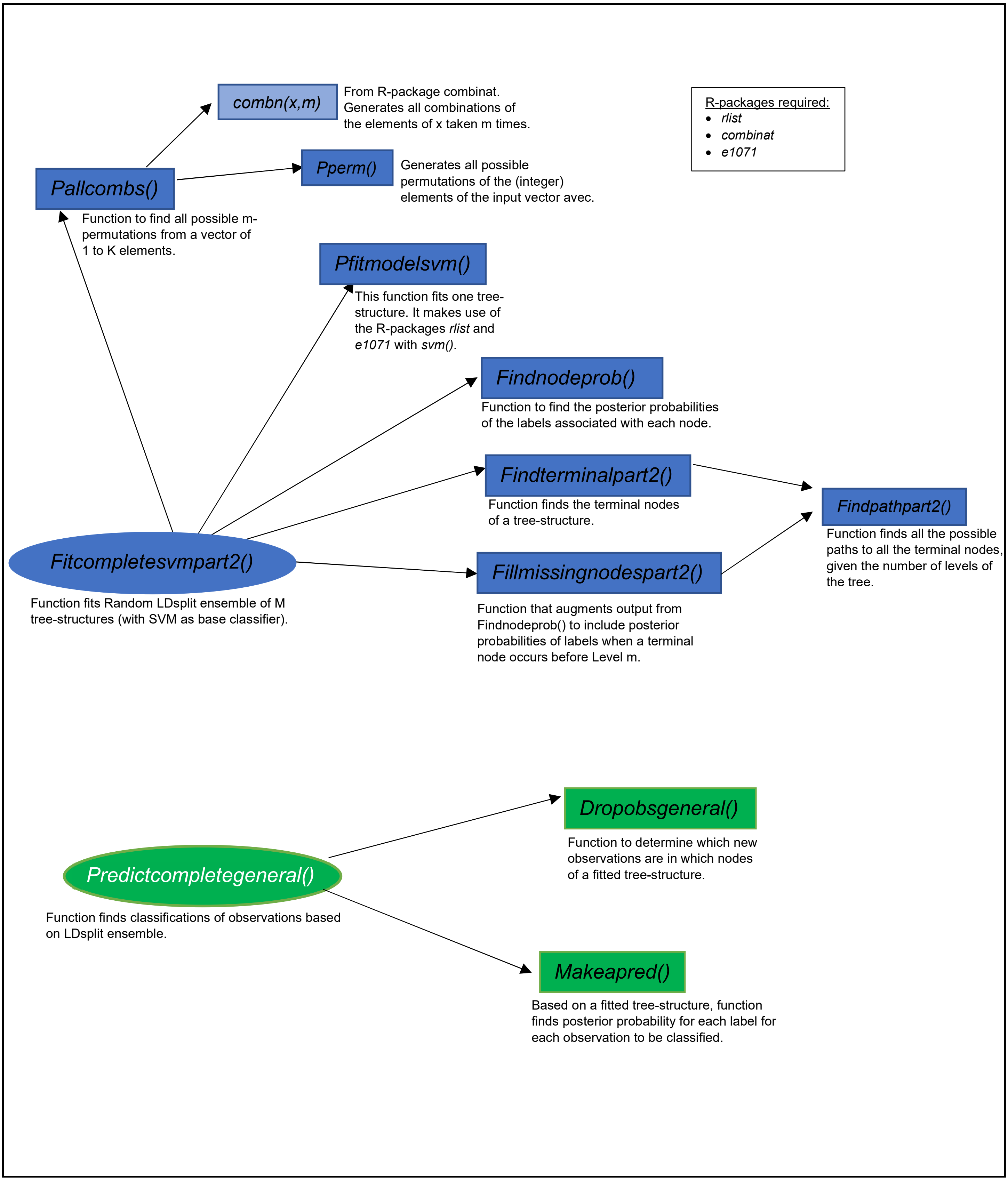
Appendix C.7 contains additional functions required to apply the **scaling** techniques as outlined in Section 3.4 when fitting **Random LDsplit** with an **SVM** base classifier.

Appendix C.8 contains additional functions required to apply the **scaling** techniques as outlined in Section 3.4 when fitting **Conditional LDsplit** with an **SVM** base classifier.

Appendix C.9 contains additional functions required to apply the **scaling** techniques as outlined in Section 3.4 when fitting **Random LDsplit** with a **decision tree** base classifier.

Appendix C.10 contains additional functions required to apply the **scaling** techniques as outlined in Section 3.4 when fitting **Conditional LDsplit** with a **decision tree** base classifier.

C.1 Functions to fit Random LDsplit with an SVM base classifier



Pallcombs:

```
Pallcombs<-function (K,m)
{
############################################################
# Function to find all possible m-permutations from a vector
# of 1 to K elements.
# Uses package combinat
# Inputs:
#   K = total number of labels
#   m = number of levels of each tree
# Output:
#   (K*(K-1)*...*(K-m+1)) * m matrix of all possibilities
############################################################

labels<-c(1:K)
allcombn<-t(combn(labels,m))
Ctotal<-nrow(allcombn)
Ttotal<-prod(c((K-m+1):K))
percomb<-prod(c(1:m))        #m!

allpossible<-matrix(0,nrow=Ttotal,ncol=m)
allpossible[1:percomb,]<-Pperm(allcombn[1,])
if(Ctotal>1){
for(k in 1:(Ctotal-1)){
allpossible[((k*percomb)+1):(percomb*(k+1)),]<-Pperm(allcombn[(k+1),])}}

return(allpossible)
}
```

Pperm:

```
Pperm<-function(avec)
{
########################################################################

# This function generates all possible permutations of the (integer)
# elements of the input vector avec

########################################################################

 n = length(avec)
 Amat = matrix(0,nrow=factorial(n),ncol=n)
 count = 1
 cvec = rep(0,n)
 Amat[count,] = avec

 i = 0
 while (i<n) {
    if (cvec[i+1]<i) {

       if (2*(i%/%2)==i) {
          temp = avec[1]
          avec[1] = avec[i+1]
          avec[i+1] = temp }

       if (2*(i%/%2)!=i) {
          temp = avec[cvec[i+1]+1]
          avec[cvec[i+1]+1] = avec[i+1]
          avec[i+1] = temp }

       count = count+1
       Amat[count,] = avec
       cvec[i+1] = cvec[i+1]+1
       i = 0   }


    if (cvec[i+1]>=i) {
       cvec[i+1] = 0
       i = i+1 }
    }

 return(Amat)
}
```

Pfitmodelsvm:

```
Pfitmodelsvm<-function(Xmat,Ymat,yperm,minsize,m,labelthr)
{
##########################################################################
# This function fits an LDsplit tree-structure to a given dataset.
# Inputs:
#   Xmat = the matrix of input values
#   Ymat = the matrix of 0-1 label vectors
#   yperm = the vector defining the label order of the tree-structure
#   minsize = minimum node-size
#   m = number of levels of the tree-structure
#   labelthr = vector of size K containing threshold to apply for
#              each label
#
# Outputs:
#   binarymodels = list of binary models used to perform the splitting
#   obsinnodes = a list containing the indices of the cases in all of
#                the nodes
#   keepprobs = posterior probabilities of training observations in
#               each node
#   emptyyesno = vector with -1 if node is empty, and 1 if node is
#                not empty
#   splityesno = vector with 1 if node was split, and -1 if node was
#                not split
#
##########################################################################
 n<-nrow(Xmat)
 p<-ncol(Xmat)
 K<-ncol(Ymat)

binarymodels<-vector("list",(2^m)-1)
                       #(2^m)-1 splits/binary classifiers for a tree

#emptyyesno<-rep(0,(2^(m+1))-1)

splityesno<-rep(0,(2^m)-1)

Ymatperm<-matrix(0,nrow=n,ncol=m)
Ythr<-rep(0,m)
 for (k in 1:m) Ymatperm[,k] = Ymat[,yperm[k]]     #organise labels
 for (k in 1:m) Ythr[k] = labelthr[yperm[k]]       #organise thresholds
```

```
######################
## Root node split  ##
######################
xmat<-Xmat
yvec<-Ymatperm[,1]
thresh<-Ythr[1]

uit1<-svm(xmat,factor(yvec),probability=TRUE)   #fit a binary classifier
binarymodels[[1]]<-uit1
emptyyesno<-c(1)
splityesno[1]<-1

predictions<-predict(uit1,xmat,probability=TRUE)
probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
                      #making sure correct posterior probability is used

classes<-ifelse(probs>=thresh,1,0)           #Classifying training cases
ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
               #Split the x-values based on the classifier and threshold
obsinnodes = list(1:n, ygroup1, ygroup0)
                         #Observations in root/node 1, node 2 and node 3
keepprobs<-list(probs[ygroup1],probs[ygroup0])
emptyyesno<-c(emptyyesno,ifelse(length(ygroup1)>0,1,-1),
ifelse(length(ygroup0)>0,1,-1))

######################
## Level k's splits ##
######################
 for (k in 1:(m-1)) {
    yvec = Ymatperm[,k+1]
    thresh = Ythr[k+1]
    beginn = 2^k                    ##first node number of level k
    eindig = 2^(k+1) -1             ##last node number of level k
    for (t in beginn:eindig) {
       indekse = obsinnodes[[t]]
       if ( (length(indekse)>minsize) && (var(yvec[indekse])>0) ){
          uit1 = svm(xmat[indekse,],factor(yvec[indekse]),
probability=TRUE)
                #fit a binary classifier

          binarymodels[[t]]<-uit1

          predictions<-predict(uit1,xmat[indekse,],probability=TRUE)
          probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
                   #making sure correct posterior probability is used
```

```
          classes<-ifelse(probs>=thresh,1,0)
                                    #Classifying training cases
          ygroup1<-as.numeric(which(classes>0.5))
          ygroup0<-as.numeric(which(classes<0.5))
           #Split the x-values based on the classifier and threshold

          obsinnodes = list.append(obsinnodes,obsinnodes[[t]]
[ygroup1],obsinnodes[[t]][ygroup0])
               #Observations in previous nodes plus resulting two

          keepprobs = list.append(keepprobs,probs[ygroup1],
probs[ygroup0])
        #Posterior probabilities in previous nodes plus resulting two

          emptyyesno<-c(emptyyesno,ifelse(length(ygroup1)>0,1,-1),
ifelse(length(ygroup0)>0,1,-1))

          splityesno[t]<-1
                                                                }
       if ( (length(indekse)<=minsize) || (var(yvec[indekse])==0) ) {
          binarymodels[[t]]<--1
          emptyyesno<-c(emptyyesno,-1,-1)
          obsinnodes = list.append(obsinnodes,-1,-1)
                                    #No observations in resulting nodes
          keepprobs = list.append(keepprobs,-1,-1)
          splityesno[t]<--1
                                                    }
                                           }
                                  }
uit = list(binarymodels,obsinnodes,keepprobs,emptyyesno,splityesno)
                 #Storing the classifier and observations in each node

return(uit)
}
```

Findnodeprob:

```
Findnodeprob<-function (Ymat,yperm,obslist,m)
{
########################################################################
# Function to find the posterior probabilities
# of the labels associated with each node.
# Inputs:
#   Ymat = the matrix of 0-1 label vectors
#   yperm = the vector giving the order in which the labels were used
#   obslist = a list containing the indices of the cases in all
#             of the nodes
#   m = number of levels
#
# Outputs:
#    probs = vector of posterior probabilities of 1 for corresponding
#          label associated with each node (excluding the first node)
#
#######################################################################
n<-nrow(Ymat)
K<-ncol(Ymat)

Ymatperm<-matrix(0,nrow=n,ncol=m)
for(k in 1:m){Ymatperm[,k]<-Ymat[,yperm[k]]}

probs<-rep(0,(2^(m+1))-1)

for(k in 1:m){
beginn<-2^k
eindig<-2^(k+1)-1

for(t in beginn:eindig){
indekse<-obslist[[t]]


if(sum(indekse)>0.5){
need<-Ymatperm[indekse,k]
ene<-sum(need)
aantal<-length(indekse)
probs[t]<-ene/aantal}

if(sum(indekse)<0.5){
probs[t]<--1}
}}

probs<-probs[-1]
return(probs)
}
```

Findterminalpart2:

```
Findterminalpart2<-function(emptyyesno,m)
{
########################################################################
# Function to find terminal nodes of a tree-structure
# Inputs:
#       emptyyesno = vector with -1 if node is empty, and 1 if node is
#                    not empty. (Obtained as output of Pfitmodelsvm() )
#       m = number of levels
# Outputs:
#       terminals = vector where each entry corresponds to the terminal
#                   node for that particular path
#       pathindices = number of the terminal node entry in path
#######################################################################
paths<-Findpathpart2(m)

terminals<-rep(0,2^m)
pathindices<-rep(0,2^m)

for(j in 1:(2^m)){

if( sum(emptyyesno[paths[j,]]<0) >0){
terminals[j]<- paths[j,min(which(emptyyesno[paths[j,]]<0))-1]
pathindices[j]<-min(which(emptyyesno[paths[j,]]<0))-1}

if( sum(emptyyesno[paths[j,]]<0) ==0){
terminals[j]<- paths[j,m+1]
pathindices[j]<-m+1}

}
return(list(terminals=terminals,pathindices=pathindices))
}
```

Findpathpart2:

```
Findpathpart2<-function(m)
{
##################################################################
# Function finds all the possible paths to all the terminal nodes,
# given the number of levels of the tree.
# Input:
#  m = number of levels
# Output:
# genpath = matrix with all possible paths to terminal nodes
#################################################################
paths<-matrix(0,nrow=2^m,ncol=m)
for(k in 1:m){
beginn<-2^k
eindig<-2^(k+1)-1
lengte<-2^k
use<-c(beginn:eindig)
veelvoud<-(2^m)/lengte
paths[1:veelvoud,k]<-rep(use[1],2^(m-k))
for(j in 2:lengte){
paths[((veelvoud*(j-1))+1):(veelvoud*(j)),k]<-rep(use[j],2^(m-k))}
}
onemat<-matrix(1,nrow=2^m,ncol=1)
genpath<-cbind(onemat,paths)
return(genpath)
}
```

Fillmissingnodespart2:

```
Fillmissingnodespart2<-function(pathind,terminal,Ymat,yperm,m,
obslist,probs)
{
########################################################################
# Function that augments output from Findnodeprob() to include posterior
# probabilities of labels when a terminal node occurs before Level m.
# Inputs:
#       pathind = number of the terminal node entry in path
#       terminal = vector of nodes that are terminal
#       Ymat = the matrix of 0-1 label vectors
#       yperm = the vector giving the order in which the labels were used
#       m = number of levels
#       obslist = a list containing the indices of the cases in all
#                 of the nodes
#       probs = output from Findnodeprob()
# Outputs:
# probsextend = complete vector of the posterior probabilities of 1 for
#               corresponding label associated with each node
#               (excluding first node)
#######################################################################
n<-nrow(Ymat)
Ymatperm<-matrix(0,nrow=n,ncol=m)
for(k in 1:m){Ymatperm[,k]<-Ymat[,yperm[k]]}

probsextend<-c(-1,probs)

missingreplace<-rep(-1,(2^(m+1))-1)
paths<-Findpathpart2(m)
rownumbers<-which(pathind<m+1)

if(sum(rownumbers)>0.5){
colnumbers<-pathind[rownumbers]
nodenumbers<-terminal[rownumbers]
howmany<-length(rownumbers)

for(k in 1:howmany){
colnow<-colnumbers[k]
rownow<-rownumbers[k]
nodenow<-nodenumbers[k]
indekse<-obslist[[nodenow]]
```

```
begin<-colnow
eindig<-m
for(j in begin:eindig){
need<-Ymatperm[indekse,j]
ene<-sum(need)
aantal<-length(indekse)
missingreplace[paths[rownow,j+1]]<-ene/aantal
}}

nonentries<-which(missingreplace>-1)

probsextend[nonentries]<-missingreplace[nonentries]}
probsextend<-probsextend[-1]
return(probsextend)
}
```

Fitcompletesvmpart2:

```
Fitcompletesvmpart2<-function(Xmat,Ymat,K,M,m,minsize,labelthr)
{
###############################
# Function to fit ensemble of M tree-structures.
# Inputs
#      Xmat = the matrix of input values
#      Ymat = the matrix of 0-1 label vectors
#      K = total number of labels
#      M = number of permutations to use (size of ensemble)
#      m = number of levels
#      minsize = minimum node-size
#      labelthr = vector of size K containing threshold to apply
#                 for each label
# Outputs:
#     keepmodels = list of all models used
#     keepnodeprobs = matrix where each row is node probabilities
#                     of a permutation
#     theperms = the matrix of permutations used
#     splityesno = matrix where each row gives code if node was
#                  split or not split (1 split, -1 no split)
################################
p<-ncol(Xmat)

#Sample M permutations using Pallcombs
theperms<-matrix(0,nrow=M,ncol=m)
allperms<-Pallcombs(K,m)
numberofperms<-nrow(allperms)
permsused<-sample(c(1:numberofperms),M)


for(k in 1:M){
theperms[k,]<-allperms[permsused[k],]}
```

```
#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
       #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
       #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)

for(j in 1:M){

#Use Pfitmodel() for each permutation
out<-Pfitmodelsvm(Xmat,Ymat,theperms[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]

#Find node probabilities for a permutation
nodeprobsout<-Findnodeprob(Ymat,theperms[j,],out[[2]],m)

#Find terminal nodes of a permutation
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theperms[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
      #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
      #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theperms,splityesno))
}
```

Dropobsgeneral:

```
Dropobsgeneral<-function(Xmatnew,modelsused,m,yperm,labelthr,splityesno)
{
####################################################################
# This function is used to fit the LDsplit model to new observations.
# More specifically, the function determines which of the (new)
# observations are in which nodes of a fitted tree-structure.
#Inputs:
#   Xmatnew = the matrix of new input values
#   modelsused = list of models used to split nodes of
#                fitted tree-structure
#   m = number of levels
#   yperm = vector giving the order in which the labels were used
#           when the tree-structure was fit
#   labelthr = vector of size K containing threshold to apply for
#              each label
#   splityesno = vector with 1 if node of fitted tree-structure was
#                split and -1 if node was not split
#Output:
#   obsinnodes = a list containing the indices of the new cases in
#                 all of the nodes
##################################################################
n<-nrow(Xmatnew)
#############
# Root node #
#############
label<-yperm[1]
predictions<-predict(modelsused[[1]],Xmatnew,probability=TRUE)

probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
          #making sure correct posterior probability is used

classes<-ifelse(probs>=labelthr[label],1,0)
          #Classifying test cases

ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
          #Split the x-values based on the classifier and threshold
obsinnodes = list(1:n, ygroup1, ygroup0)
          #Observations in root/node 1, node 2 and node 3
```

```
###################
# Level k's split #
###################
for(k in 1:(m-1)){
beginn<-2^k                  #first node number of level k
eindig<-2^(k+1)-1            #last node number of level k
label<-yperm[k+1]

for(t in beginn:eindig){
indekse<-obsinnodes[[t]]

if( (splityesno[t]>0) && (sum(indekse)>0.5) ) {
         #if node 'was' split and there are currently
         #test observations in the node

if( length(indekse)>1 ){
predictions<-predict(modelsused[[t]],Xmatnew[indekse,],
probability=TRUE)
probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
         #making sure correct posterior probability is used


classes<-ifelse(probs>=labelthr[label],1,0)
         #Classifying test cases
ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
         #Split the x-values based on the classifier and threshold
obsinnodes<-list.append(obsinnodes,obsinnodes[[t]][ygroup1],
obsinnodes[[t]][ygroup0])
         #Observations in previous nodes plus resulting two
}



if( length(indekse)==1 ){
predictions<-predict(modelsused[[t]],matrix(Xmatnew[indekse,],nrow=1),
probability=TRUE)
        #transform to rowvector
probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
       #making sure correct posterior probability is used

classes<-ifelse(probs>=labelthr[label],1,0)
       #Classifying test cases
ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
       #Split the x-values based on the classifier and threshold
obsinnodes<-list.append(obsinnodes,obsinnodes[[t]]
[ygroup1],obsinnodes[[t]][ygroup0])
       #Observations in previous nodes plus resulting two
```

```
}
}


if( (splityesno[t]>0) && (sum(indekse)<0.5) ) {
      #if node 'was' split but there are currently no
      #test observations in node
obsinnodes<-list.append(obsinnodes,-1,-1)}


if(splityesno[t]<0){
      #if node did not split
obsinnodes<-list.append(obsinnodes,-1,-1)}
}}
return(obsinnodes)
}
```

Makeapred:

```
Makeapred<-function(obslist,probs,K,Nnew,yperm,m)
{
########################################################################
# Based on a fitted tree-structure, function finds posterior probability
# for each label for each observation to be classified
# Inputs:
#     obslist = list of observations in each node
#     probs = vector of posterior probabilities of 1 for each node
#             (excluding first)
#     K = number of labels
#     Nnew = number of observations to be classified
#     yperm = the vector giving the order in which the labels were used
#            when the tree-structure was fit
#     m = number of levels
#   Output:
#    postprobmatorder = Nnew*K matrix of posterior probabilities
#######################################################################

postprobmat<-matrix(-1,nrow=Nnew,ncol=m)
probs<-c(-1,probs)

for(k in 1:m){
beginn<-2^k
eindig<-2^(k+1)-1

for(t in beginn:eindig){
indekse<-obslist[[t]]

if(sum(indekse)>0.5){
postprobmat[indekse,k]<-probs[t]}
```

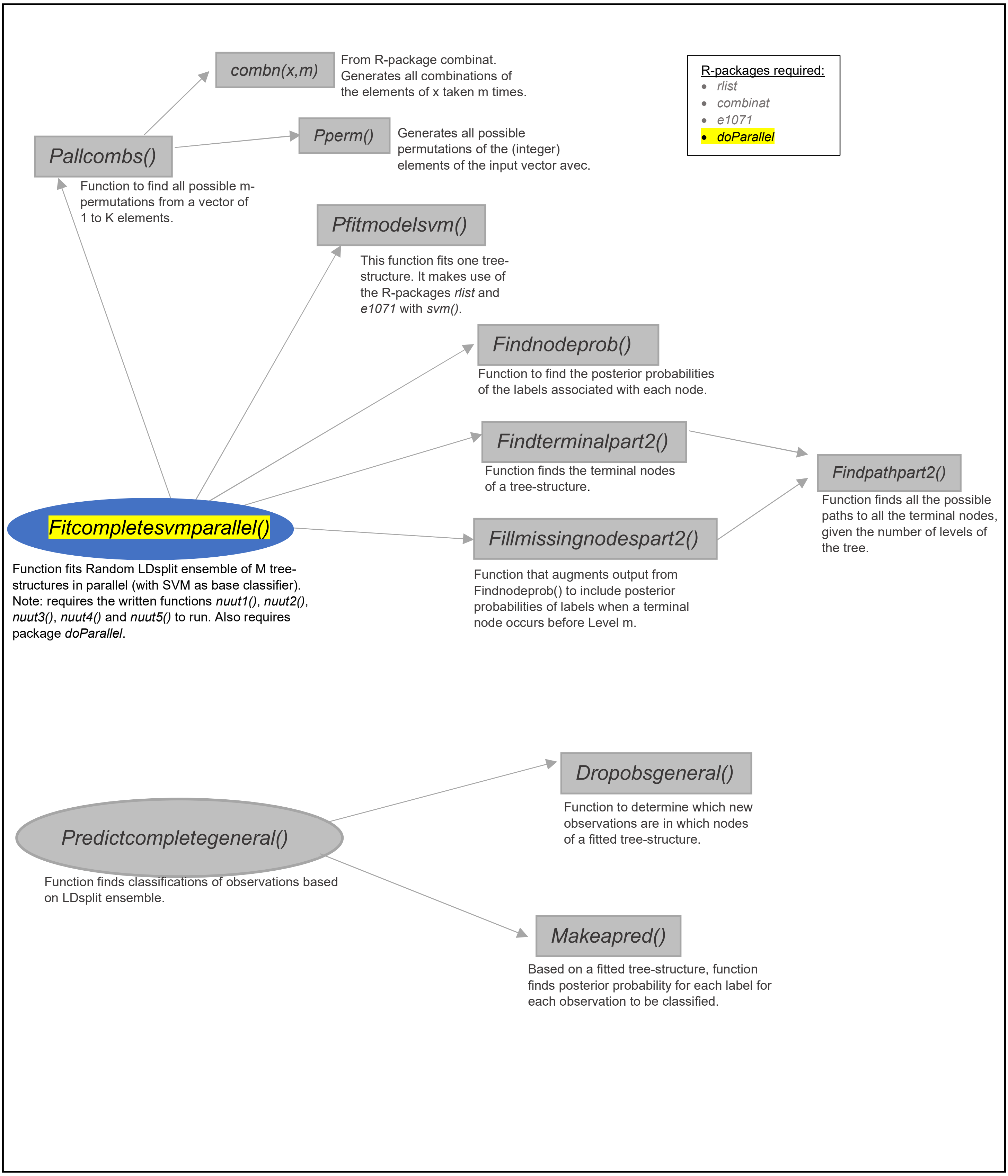
```
if(sum(indekse)<0.5){
postprobmat[indekse,k]<-probs[t]}
}
}

postprobmatorder<-matrix(-1,nrow=Nnew,ncol=K)
for(k in 1:m){
postprobmatorder[,yperm[k]]<-postprobmat[,k]}

return(postprobmatorder)
}
```

Predictcompletegeneral:

```
Predictcompletegeneral<-function(Xmatnew,modelsused,nodeprobs,perms,
splityesno,M,K,m,labelthr)
{
#######################################################################
# Function finds classifications of observations based on
# LDsplit ensemble.
# Inputs:
#   Xmatnew = the matrix of new input values
#   modelsused = list of trained models used to split data
#   nodeprobs = matrix where each row is node probabilities
#               of a permutation
#   perms = matrix where each row gives permutation used
#   splityesno = matrix where each row gives code if node was split
#                or not (1 = split, -1 = no split)
#   M = number of permutations (size of ensemble)
#   K = total number of labels
#   m = total number of levels
#   labelthr = vector of size K containing threshold to apply
#              for each label
# Output:
#   finalresults = Nnew*K matrix of posterior probabilities
#####################################################################

Nnew<-nrow(Xmatnew)

### Keep predictions ###
allpredictions<-array(-1,dim=c(Nnew,K,M))

for(j in 1:M){

### Drop observations in each tree-structure ###
obslist<-Dropobsgeneral(Xmatnew,modelsused[[j]],m,perms[j,],labelthr,
splityesno[j,])
```

```
### Find predictions of permutations ###
thepredictions<-Makeapred(obslist,nodeprobs[j,],K,Nnew,perms[j,],m)
allpredictions[,,j]<-thepredictions
}

finalresults<-matrix(0,nrow=Nnew,ncol=K)

for(n in 1:Nnew){
for(k in 1:K){
tel<-0
for(j in 1:M){
if(allpredictions[n,k,j]>=0){
tel<-tel+1
finalresults[n,k]<-finalresults[n,k]+allpredictions[n,k,j]}
}
finalresults[n,k]<-(finalresults[n,k])/tel
}}

finalresults[finalresults>=0.5]<-1
finalresults[finalresults<0.5]<-0
return(finalresults)
}
```

C.2 Functions to fit Random LDsplit with an SVM base classifier (written in parallel)



Fitcompletesvmparallel:

```
Fitcompletesvmparallel<-function(Xmat,Ymat,K,M,m,minsize,labelthr)
{
#######################################################################
# Function to fit ensemble of M tree-structures in parallel
# Inputs:
#      Xmat = the matrix of input values
#      Ymat = the matrix of 0-1 label vectors
#      K = total number of labels
#      M = number of permutations to use (size of ensemble)
#      m = number of levels
#      minsize = minimum node-size
#      labelthr = vector of size K containing threshold to apply
#                 for each label
# Outputs:
#     keepmodels = list of all models used
#     keepnodeprobs = matrix where each row is node probabilities
#                     of a permutation
#     theperms = the matrix of permutations used
#     splityesno = matrix where each row gives code if node was
#                  split or not (1 = split, -1 = no split)
#####################################################################
clust<-makeCluster(4)         #needed to run parallel
registerDoParallel(clust)     #needed to run parallel

p<-ncol(Xmat)

#Sample M permutations using Pallcombs
theperms<-matrix(0,nrow=M,ncol=m)
ttheperms<-matrix(0,nrow=m,ncol=M)      #needed to run parallel
allperms<-Pallcombs(K,m)
numberofperms<-nrow(allperms)
permsused<-sample(c(1:numberofperms),M)

for(k in 1:M){
theperms[k,]<-allperms[permsused[k],]}


for(k in 1:M){
ttheperms[,k]<-allperms[permsused[k],]}  #needed to run parallel

#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
       #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
       #keep the posterior probabilites of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)
```

```
### Needed to run parallel
fullsized<-parLapply(clust,as.data.frame(ttheperms),Pfitmodelsvm,
Xmat=Xmat,Ymat=Ymat,minsize=minsize,m=m,labelthr=labelthr)
fullsized1<-lapply(fullsized,nuut1)  #nuut1 is a written function
fullsized2<-lapply(fullsized,nuut2)  #nuut2 is a written function
fullsized3<-lapply(fullsized,nuut3)  #nuut3 is a written function
fullsized4<-lapply(fullsized,nuut4)  #nuut4 is a written function
fullsized5<-lapply(fullsized,nuut5)  #nuut5 is a written function

for(j in 1:M){
splityesno[j,]<-fullsized5[[j]][[1]]

#Find node probabilities for a permutation
nodeprobsout<-Findnodeprob(Ymat,theperms[j,],fullsized2[[j]][[1]],m)

#Find terminal nodes of a permutation
this<-Findterminalpart2(fullsized4[[j]][[1]],m)
terminalout<-this$terminals
pathindout<-this$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theperms[j,],m,fullsized2[[j]][[1]],nodeprobsout)

keepmodels[[j]]<-fullsized1[[j]][[1]]
        #keep the binary models


keepnodeprobs[j,]<-fillednodeprobsout
        #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theperms,splityesno))
}
```

nuut1, nuut2, nuut3, nuut4, nuut5:

```
nuut1<-function(x)
{uit<-x[1]}
```

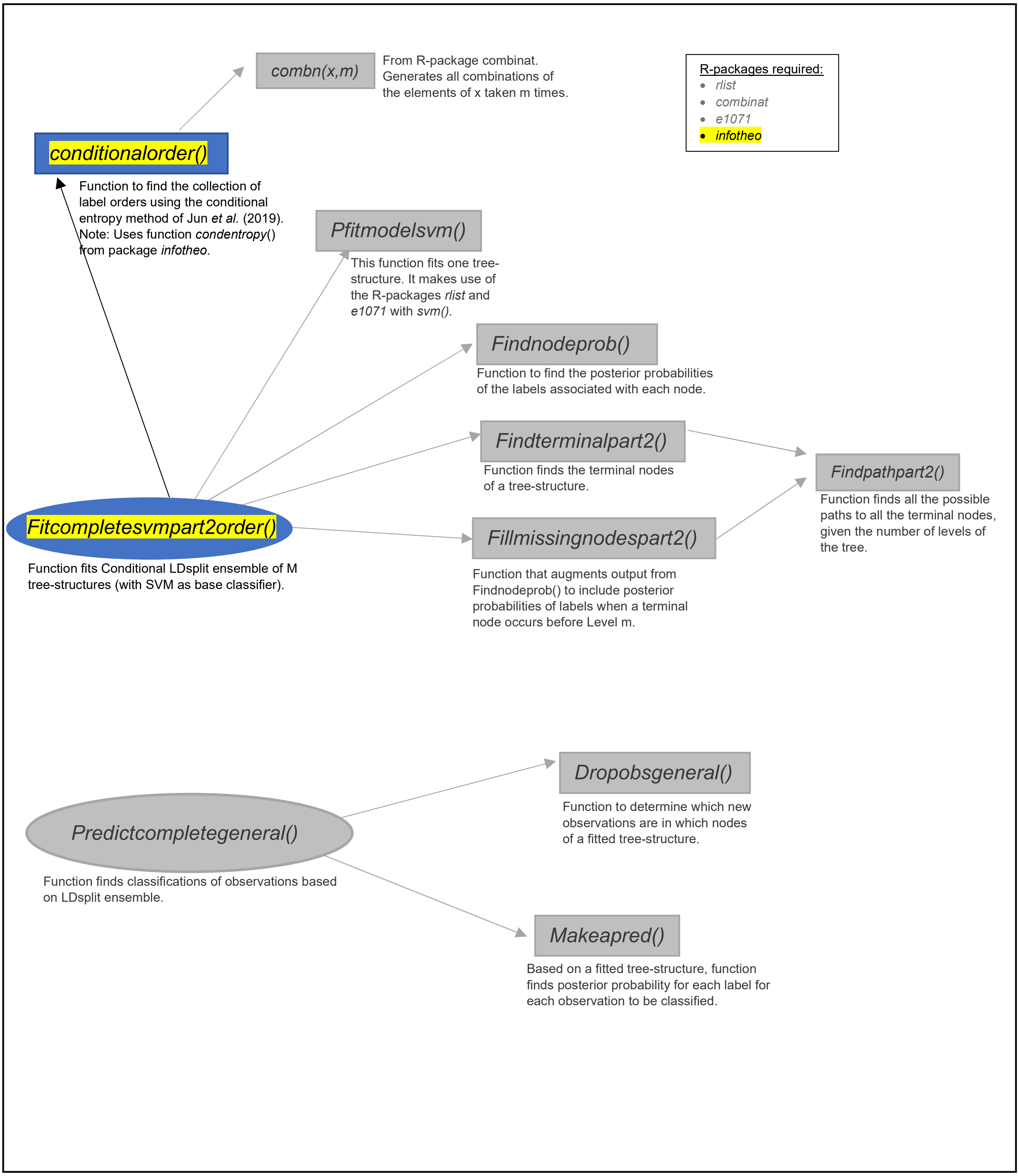
```
nuut2<-function(x)
{uit<-x[2]}
```

```
nuut3<-function(x)
{uit<-x[3]}
```

```
nuut4<-function(x)
{uit<-x[4]}
```

```
nuut5<-function(x)
{uit<-x[5]}
```

C.3 Functions to fit Conditional LDsplit with an SVM base classifier



conditionalorder():

```
conditionalorder<-function (labelset,m,K)
{
######################################################################
#Function forms the collection of label orders using the conditional
#entropy method of Jun et al. (2019).
#Note: function uses condentropy() from package infotheo.

#Inputs:
#   labelset = matrix of 0-1 label vectors
#   m = number of levels
#   K = total number of labels
#
#Output:
#   theordersmat = matrix with label orders given as rows

######################################################################
### How many orders ###
if(K==m){
howmany<-1}
if(m<K){
howmany<-ncol(combn(K,m))}
allcomb<-combn(K,m)

### Finding orders ###
theordersmat<-matrix(0,nrow=howmany,ncol=m)

if(m>2){
for(h in 1:howmany){

### Per order:
## Labels that are considered:
if(howmany==1){
thecurrentlabels<-allcomb}
if(howmany>1){
thecurrentlabels<-allcomb[,h]}


## Finding matrix of conditional entropies:
thecurrentlabelset<-labelset[,thecurrentlabels]
entropymat<-matrix(0,nrow=m,ncol=m)
for(i in 1:m){
for(j in 1:m){
entropymat[i,j]<-condentropy(thecurrentlabelset[,j],
thecurrentlabelset[,i])}}
```

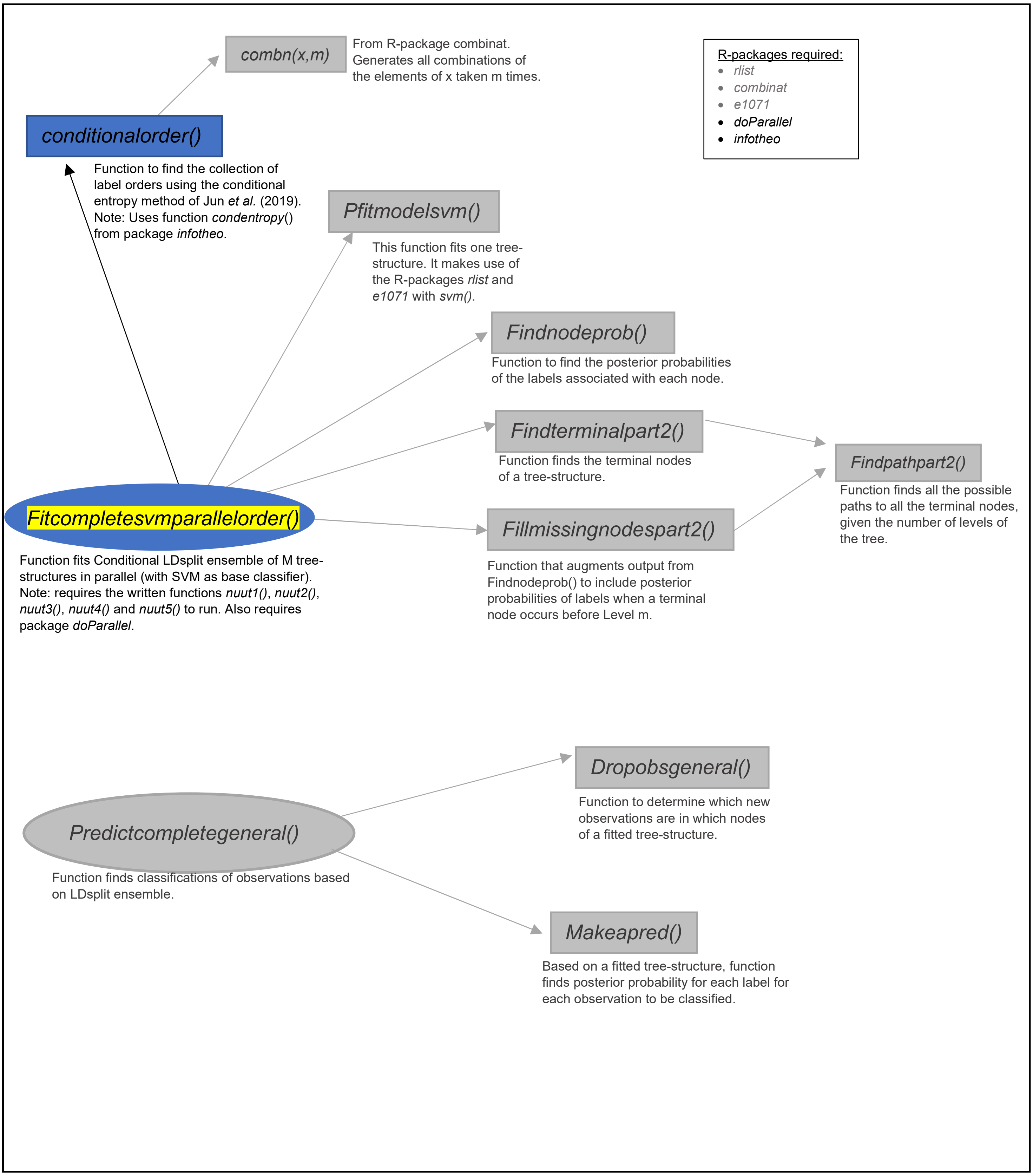
```
tel<-0
usingmat<-entropymat
thelabels<-thecurrentlabels
for(k in 1:(m-2)){
thesums<-apply(usingmat,1,sum)
ith<-which(thesums==min(thesums))
theordersmat[h,m-tel]<-thelabels[ith]
tel<-tel+1
usingmat<-usingmat[,-ith]
usingmat<-usingmat[-ith,]
thelabels<-thelabels[-ith]
}
## m-1
thesums<-apply(usingmat,1,sum)
ith<-which(thesums==min(thesums))
other<-which(thesums==max(thesums))
theordersmat[h,m-tel]<-thelabels[ith]
theordersmat[h,m-tel-1]<-thelabels[other]
}
}
if(m<=2){
for(h in 1:howmany){

### Per order:
## Labels that are considered:
if(howmany==1){
thecurrentlabels<-allcomb}
if(howmany>1){
thecurrentlabels<-allcomb[,h]}

## Finding matrix of conditional entropies:
thecurrentlabelset<-labelset[,thecurrentlabels]
entropymat<-matrix(0,nrow=m,ncol=m)
for(i in 1:m){
for(j in 1:m){
entropymat[i,j]<-condentropy(thecurrentlabelset[,j],
thecurrentlabelset[,i])}}

tel<-0
usingmat<-entropymat
thelabels<-thecurrentlabels
## m-1
thesums<-apply(usingmat,1,sum)
ith<-which(thesums==min(thesums))
other<-which(thesums==max(thesums))
theordersmat[h,m-tel]<-thelabels[ith]
theordersmat[h,m-tel-1]<-thelabels[other]
}
}
return((theordersmat))
}
```

Fitcompletesvmpart2order:

```
Fitcompletesvmpart2order<-function(Xmat,Ymat,K,M,m,minsize,labelthr)
{
##########################################################################
# Function fits Conditional LDsplit ensemble of M tree-structures
# (with SVM as base classifier).
# Inputs
#     Xmat = the matrix of input values
#     Ymat = the matrix of 0-1 label vectors
#     K = total number of labels
#     M = number of label orders to use (ensemble size)
#     m = number of levels
#     minsize = minimum node-size
#     labelthr = vector of size K containing threshold to apply
#                for each label
# Outputs:
#    keepmodels = list of all models used
#    keepnodeprobs = matrix where each row is node probabilities of
#                    a label order
#    theperms = the matrix of label orders used
#    splityesno = matrix where each row gives code if node was split
#                 or not split (1 = split, -1 = no split)
#########################################################################
p<-ncol(Xmat)

#Sample M label orders using CONDITIONAL ENTROPY
theorders<-matrix(0,nrow=M,ncol=m)
allorders<-conditionalorder(labelset=Ymat,m,K)
numberoforders<-choose(K,m)
ordersused<-sample(c(1:numberoforders),M)

for(k in 1:M){
theorders[k,]<-allorders[ordersused[k],]}


#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
       #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
       #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)

for(j in 1:M){
```

```
#Use Pfitmodel for each label order
out<-Pfitmodelsvm(Xmat,Ymat,theorders[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]

#Find node probabilities for a label ordering
nodeprobsout<-Findnodeprob(Ymat,theorders[j,],out[[2]],m)

#Find terminal nodes of a label ordering
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theorders[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
      #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
      #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theorders,splityesno))
}
```

C.4 Functions to fit Conditional LDsplit with an SVM base classifier (written in parallel)

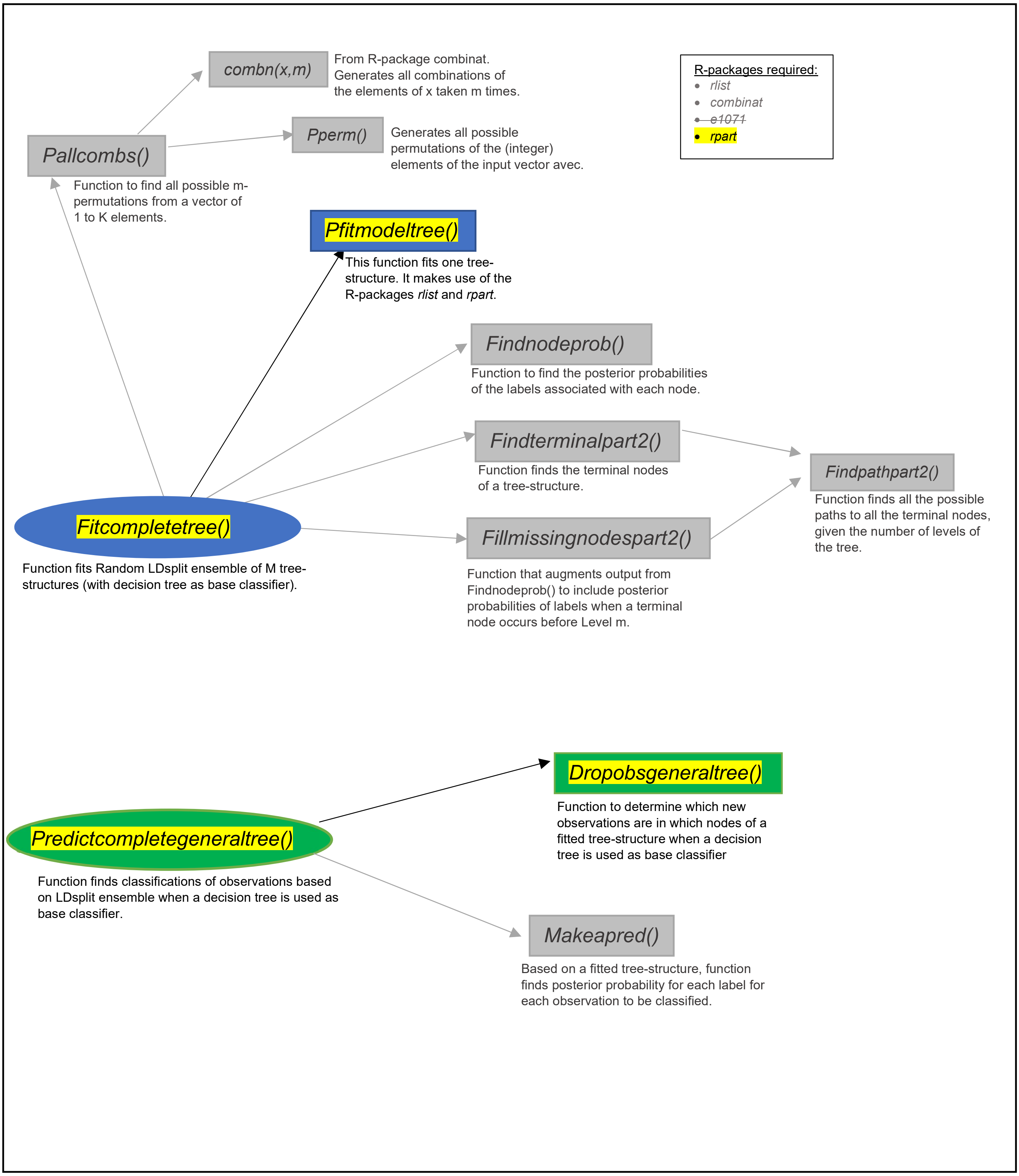


Fitcompletesvmparallelorder:

```
Fitcompletesvmparallelorder<-function(Xmat,Ymat,K,M,m,minsize,labelthr)
{
########################################################################
# Function fits Conditional LDsplit ensemble of M tree-structures in
# parallel (with SVM as base classifier).
# Note: requires the written functions nuut1(), nuut2(), nuut3(), nuut4()
# and nuut5() to run. Also requires package doParallel.
#
# Inputs:
#      Xmat = the matrix of input values
#      Ymat = the matrix of 0-1 label vectors
#      K = total number of labels
#      M = number of permutations to use (size of ensemble)
#      m = number of levels
#      minsize = minimum node-size
#      labelthr = vector of size K containing threshold to
#                 apply for each label
# Outputs:
#     keepmodels = list of all models used
#     keepnodeprobs = matrix where each row is node probabilities
#                     of a permutation
#     theorders = the matrix of label orders used
#     splityesno: matrix where each row gives code if node was split
#                 or not split (1 = split, -1 = no split)
#######################################################################
clust<-makeCluster(3)          #needed to run parallel
registerDoParallel(clust)      #needed to run parallel
p<-ncol(Xmat)

#Sample M orders using CONDITIONAL ENTROPY
theorders<-matrix(0,nrow=M,ncol=m)
ttheorders<-matrix(0,nrow=m,ncol=M)        #needed to run parallel
allorders<-conditionalorder(labelset=Ymat,m,K)
numberoforders<-choose(K,m)
ordersused<-sample(c(1:numberoforders),M)


for(k in 1:M){
theorders[k,]<-allorders[ordersused[k],]}

for(k in 1:M){
ttheorders[,k]<-allorders[ordersused[k],]}  #needed to run parallel
```

```
#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
       #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
       #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)

### Needed to run parallel
fullsized<-parLapply(clust,as.data.frame(ttheorders),Pfitmodelsvm,
Xmat=Xmat,Ymat=Ymat,minsize=minsize,m=m,labelthr=labelthr)
fullsized1<-lapply(fullsized,nuut1)
fullsized2<-lapply(fullsized,nuut2)
fullsized3<-lapply(fullsized,nuut3)
fullsized4<-lapply(fullsized,nuut4)
fullsized5<-lapply(fullsized,nuut5)

for(j in 1:M){
splityesno[j,]<-fullsized5[[j]][[1]]

#Find node probabilities for a label ordering
nodeprobsout<-Findnodeprob(Ymat,theorders[j,],fullsized2[[j]][[1]],m)

#Find terminal nodes of a label ordering
this<-Findterminalpart2(fullsized4[[j]][[1]],m)
terminalout<-this$terminals
pathindout<-this$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theorders[j,],m,fullsized2[[j]][[1]],nodeprobsout)

keepmodels[[j]]<-fullsized1[[j]][[1]]
       #keep the binary models

keepnodeprobs[j,]<-fillednodeprobsout
       #keep the posterior probabilities of each node (excluding first)


}
return(list(keepmodels,keepnodeprobs,theorders,splityesno))
}
```

C.5 Functions to fit Random LDsplit with a decision tree base classifier



Pfitmodeltree:

```
Pfitmodeltree<-function(Xmat,Ymat,yperm,minsize,m,labelthr)
{
########################################################################
# This function fits an LDsplit tree-structure to a given dataset when
# a decision tree is used as base classifier.
# The function requires R packages rlist and rpart.
# Inputs:
#   Xmat = the matrix of input values
#   Ymat = the matrix of 0-1 label vectors
#   yperm = the vector defining the label order of the tree-structure
#   minsize = minimum node-size
#   m = number of levels
#   labelthr = vector of size K containing threshold to apply for
#              each label
#
# Outputs:
#   binarymodels = list of binary models used to perform the splitting
#   obsinnodes = a list containing the indices of the cases in all of
#                the nodes
#   keepprobs = posterior probabilities of training observations in
#               each node
#   emptyyesno = vector with -1 if node is empty, and 1 if node is
#                not empty
#   splityesno = vector with 1 if node was split and -1 if node was
#                not split
########################################################################
n<-nrow(Xmat)
p<-ncol(Xmat)
K<-ncol(Ymat)

binarymodels<-vector("list",(2^m)-1)
                      #(2^m)-1 splits/binary classifiers for a tree

#emptyyesno<-rep(0,(2^(m+1))-1)
splityesno<-rep(0,(2^m)-1)


Ymatperm<-matrix(0,nrow=n,ncol=m)
Ythr<-rep(0,m)
 for (k in 1:m) Ymatperm[,k] = Ymat[,yperm[k]]   #organise labels
 for (k in 1:m) Ythr[k] = labelthr[yperm[k]]     #organise thresholds

######################
## Root node split  ##
######################
xmat<-Xmat
yvec<-Ymatperm[,1]
thresh<-Ythr[1]
```

```
datfr<-as.data.frame(cbind(xmat,yvec))
colnames(datfr)[p+1]<-"label"
uit1<-rpart(label~., data=datfr , method = 'class')
              #Fit decision tree
binarymodels[[1]]<-uit1
emptyyesno<-c(1)
splityesno[1]<-1

predictions<-predict(uit1,newdata=as.data.frame(xmat),type="prob")
              #Decision tree predictions
probs<-(predictions)[,which(as.numeric(colnames(predictions))==1)]
              #Making sure correct posterior probability is used

classes<-ifelse(probs>=thresh,1,0)
             #Classifying training cases
ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
             #Split the x-values based on the classifier and threshold
obsinnodes = list(1:n, ygroup1, ygroup0)
             #Observations in root/node 1, node 2 and node 3
keepprobs<-list(probs[ygroup1],probs[ygroup0])
emptyyesno<-c(emptyyesno,ifelse(length(ygroup1)>0,1,-1),
ifelse(length(ygroup0)>0,1,-1))

######################
## Level k's splits ##
######################
 for (k in 1:(m-1)) {
    yvec = Ymatperm[,k+1]
    thresh = Ythr[k+1]
    beginn = 2^k                  #first node number of level k
    eindig = 2^(k+1) -1           #last node number of level k
    for (t in beginn:eindig) {
       indekse = obsinnodes[[t]]
       if ( (length(indekse)>minsize) && (var(yvec[indekse])>0) ){

          datfr<-as.data.frame(cbind(xmat[indekse,],yvec[indekse]))
          colnames(datfr)[p+1]<-"label"
          uit1<-rpart(label~., data=datfr , method = 'class')
                                #Fit decision tree
          binarymodels[[t]]<-uit1
```

```
          datfr<-as.data.frame(cbind(xmat[indekse,],yvec[indekse]))
          colnames(datfr)[p+1]<-"label"
          uit1<-rpart(label~., data=datfr , method = 'class')
                                #Fit decision tree
          binarymodels[[t]]<-uit1

          predictions<-predict(uit1,newdata=as.data.frame(xmat[indekse,]),
type="prob")
             #Decision tree predictions


          probs<-(predictions)
[,which(as.numeric(colnames(predictions))==1)]
             #Making sure correct posterior probability is used

          classes<-ifelse(probs>=thresh,1,0)
             #Classifying training cases

          ygroup1<-as.numeric(which(classes>0.5))
          ygroup0<-as.numeric(which(classes<0.5))
            #Split the x-values based on the classifier and threshold

          obsinnodes = list.append(obsinnodes,obsinnodes[[t]][ygroup1],
obsinnodes[[t]][ygroup0])
            #Observations in previous nodes plus resulting two

          keepprobs = list.append(keepprobs,probs[ygroup1],probs[ygroup0])
            #Posterior probabilities in previous nodes plus resulting two

          emptyyesno<-c(emptyyesno,ifelse(length(ygroup1)>0,1,-1),
ifelse(length(ygroup0)>0,1,-1))
          splityesno[t]<-1
                                                                   }


       if ( (length(indekse)<=minsize) || (var(yvec[indekse])==0) ) {
          binarymodels[[t]]<--1
          emptyyesno<-c(emptyyesno,-1,-1)
          obsinnodes = list.append(obsinnodes,-1,-1)
           #No observations in resulting nodes

          keepprobs = list.append(keepprobs,-1,-1)
          splityesno[t]<--1
                                                      }
                                            }
                                   }
uit = list(binarymodels,obsinnodes,keepprobs,emptyyesno,splityesno)
          #Storing the classifier and observations in each node

return(uit)
}
```

Fitcompletetree:

```
Fitcompletetree<-function(Xmat,Ymat,K,M,m,minsize,labelthr)
{
#########################################################################
# Function fits Random LDsplit ensemble of M tree-structures
# (with decision tree as base classifier).
# Inputs:
#     Xmat = the matrix of input values
#     Ymat = the matrix of 0-1 label vectors
#     K = total number of labels
#     M = number of permutations to use (size of ensemble)
#     m = number of levels
#     minsize = minimum node-size
#     labelthr = vector of size K containing threshold to apply
#                for each label
# Outputs:
#    keepmodels = list of all models used
#    keepnodeprobs = matrix where each row is node probabilities
#                    of a permutation
#    theperms = the matrix of permutations used
#    splityesno = matrix where each row gives code if node was split
#                 or not split (1 = split, -1 = no split)
########################################################################
p<-ncol(Xmat)

#Sample M permutations using Pallcombs
theperms<-matrix(0,nrow=M,ncol=m)
allperms<-Pallcombs(K,m)
numberofperms<-nrow(allperms)
permsused<-sample(c(1:numberofperms),M)


for(k in 1:M){
theperms[k,]<-allperms[permsused[k],]}


#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
        #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
        #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)

for(j in 1:M){

#Use Pfitmodeltree for each permutation
out<-Pfitmodeltree(Xmat,Ymat,theperms[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]
```

```
#Find node probabilities for a permutation
nodeprobsout<-Findnodeprob(Ymat,theperms[j,],out[[2]],m)

#Find terminal nodes of a permutation
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theperms[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
       #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
       #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theperms,splityesno))
}
```

Dropobsgeneraltree:

```
Dropobsgeneraltree<-function(Xmatnew,modelsused,m,yperm,labelthr,
splityesno)
{
########################################################################
# This function is used to fit the LDsplit model (with a decision tree
# base classifier) to new observations. More specifically, the function
# determines which of the (new) observations are in which nodes of a
# fitted tree-structure.
# Inputs:
#   Xmatnew = the matrix of new input values
#   modelsused = list of models used to split nodes of
#                fitted tree-structure
#   m = number of levels
#   yperm = vector giving the order in which the labels were used
#           when the tree-structure was fit
#   labelthr = vector of size K containing threshold to apply
#              for each label
#   splityesno = vector with 1 if node was split and -1 if node
#                was not split
# Outputs:
#    obsinnodes = a list containing the indices of the cases in all
#                 of the nodes
########################################################################
n<-nrow(Xmatnew)

### Root node ###
label<-yperm[1]
predictions<-predict(modelsused[[1]],
newdata=as.data.frame(Xmatnew),type="prob")
             #decision tree predictions
```

```
probs<-(predictions)[,which(as.numeric(colnames(predictions))==1)]
             #making sure correct posterior probability is used


classes<-ifelse(probs>=labelthr[label],1,0)  #Classifying test cases
ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
             #Split the x-values based on the classifier and threshold

obsinnodes = list(1:n, ygroup1, ygroup0)
             #Observations in root/node 1, node 2 and node 3

### Level k's split ###
for(k in 1:(m-1)){
beginn<-2^k                 #first node number of level k
eindig<-2^(k+1)-1           #last node number of level k
label<-yperm[k+1]

for(t in beginn:eindig){
indekse<-obsinnodes[[t]]

if( (splityesno[t]>0) && (sum(indekse)>0.5) ) {
          ###if node was split and there are currently
             #test observations in the node

if( length(indekse)>1 ){
predictions<-predict(modelsused[[t]],
newdata=as.data.frame(Xmatnew[indekse,]),type="prob")
             #decision tree predictions

probs<-(predictions)
[,which(as.numeric(colnames(predictions))==1)]
            #making sure correct posterior probability is used

classes<-ifelse(probs>=labelthr[label],1,0)
            #Classifying test cases


ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
           #Split the x-values based on the classifier and threshold

obsinnodes<-list.append(obsinnodes,obsinnodes[[t]]
[ygroup1],obsinnodes[[t]][ygroup0])
           #Observations in previous nodes plus resulting two
}
```

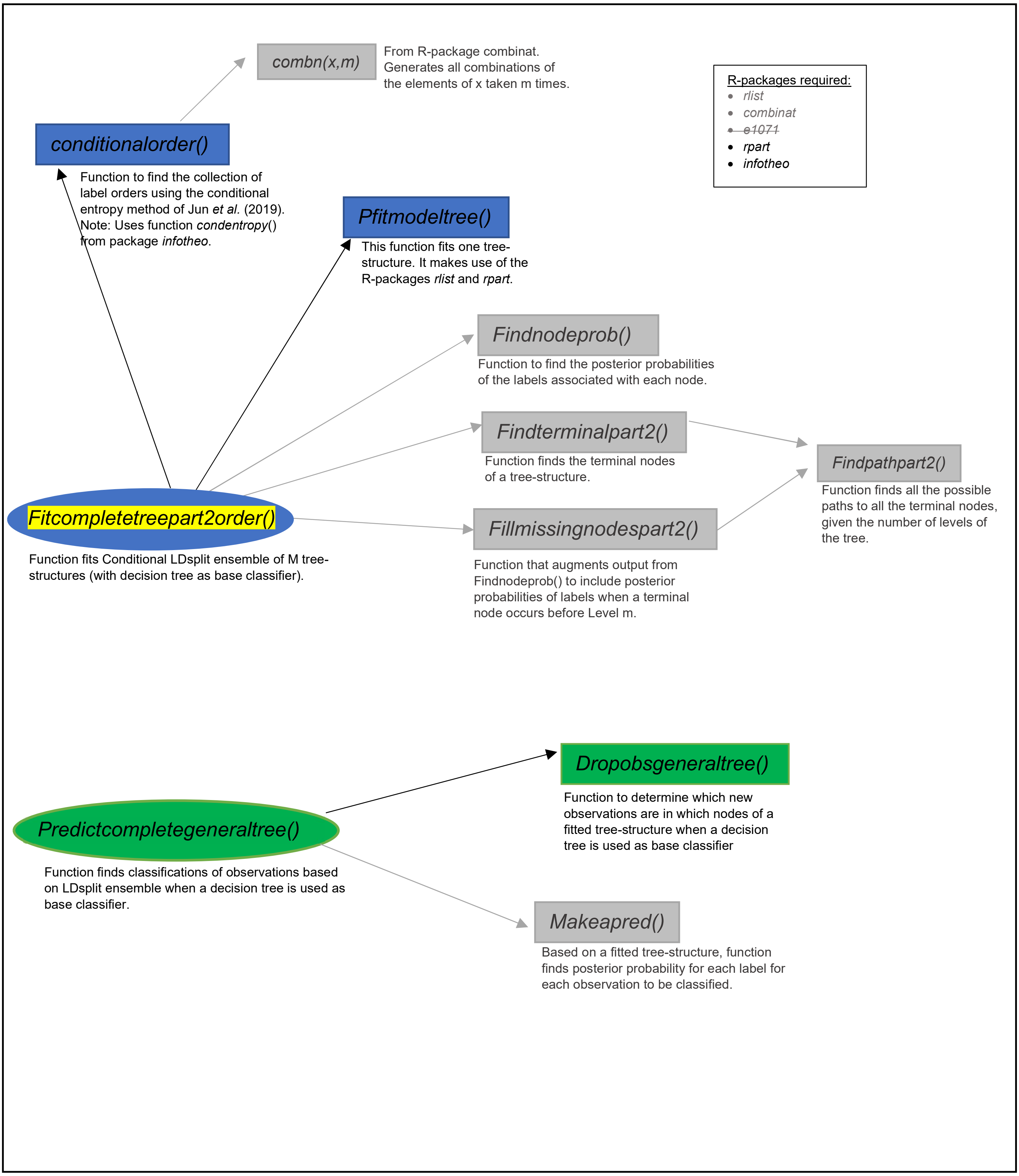
```
if( length(indekse)==1 ){
         ###only one observation in node
benodig<-matrix(c(Xmatnew[indekse,]),nrow=1,byrow=T)
colnames(benodig)<-names(Xmatnew[indekse,])
predictions<-predict(modelsused[[t]],newdata=as.data.frame(benodig),
type="prob")
           #decision tree predictions

probs<-(predictions)[,which(as.numeric(colnames(predictions))==1)]
           #making sure correct posterior probability is used

classes<-ifelse(probs>=labelthr[label],1,0)
           #Classifying test cases

ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
           #Split the x-values based on the classifier and threshold

obsinnodes<-list.append(obsinnodes,obsinnodes[[t]]
[ygroup1],obsinnodes[[t]][ygroup0])
           #Observations in previous nodes plus resulting two
}
}

if( (splityesno[t]>0) && (sum(indekse)<0.5) ) {
         ###if node was split but there are currently no test
           #observations in node
obsinnodes<-list.append(obsinnodes,-1,-1)}

if(splityesno[t]<0){
        ###if node did not split
obsinnodes<-list.append(obsinnodes,-1,-1)}
}}
return(obsinnodes)
}
```

Predictcompletegeneraltree:

```
Predictcompletegeneraltree<-function(Xmatnew,modelsused,nodeprobs,perms,
splityesno,M,K,m,labelthr)
{
#########################################################################
# Function finds classifications of observations based on LDsplit
# ensemble (when a tree is used as base classifier).
# Inputs:
#   Xmatnew = the matrix of new input values
#   modelsused = list of trained models used to split data
#   nodeprobs = matrix where each row is node probabilities
#               of a permutation
#   perms = matrix where each row gives permutation used
#   splityesno = matrix where each row gives code if node was split
#                or not split (1 = split, -1 = no split)
#   M = number of permutations (size of ensemble)
#   K = total number of labels
#   m = total number of levels
#   labelthr = vector of size K containing threshold to apply
#              for each label
# Output:
#   finalresults = Nnew*K matrix of posterior probabilities
#########################################################################

Nnew<-nrow(Xmatnew)

### Keep predictions ###
allpredictions<-array(-1,dim=c(Nnew,K,M))

for(j in 1:M){

### Drop observations in each tree-structure ###
obslist<-Dropobsgeneraltree(Xmatnew,modelsused[[j]],m,
perms[j,],labelthr,splityesno[j,])


### Find predictions of permutations ###
thepredictions<-Makeapred(obslist,nodeprobs[j,],K,Nnew,perms[j,],m)
allpredictions[,,j]<-thepredictions
}

finalresults<-matrix(0,nrow=Nnew,ncol=K)
```

```
for(n in 1:Nnew){
for(k in 1:K){
tel<-0
for(j in 1:M){
if(allpredictions[n,k,j]>=0){
tel<-tel+1
finalresults[n,k]<-finalresults[n,k]+allpredictions[n,k,j]}
}
finalresults[n,k]<-(finalresults[n,k])/tel
}}

finalresults[finalresults>=0.5]<-1
finalresults[finalresults<0.5]<-0
return(finalresults)
}
```

C.6 Functions to fit Conditional LDsplit with a decision tree base classifier

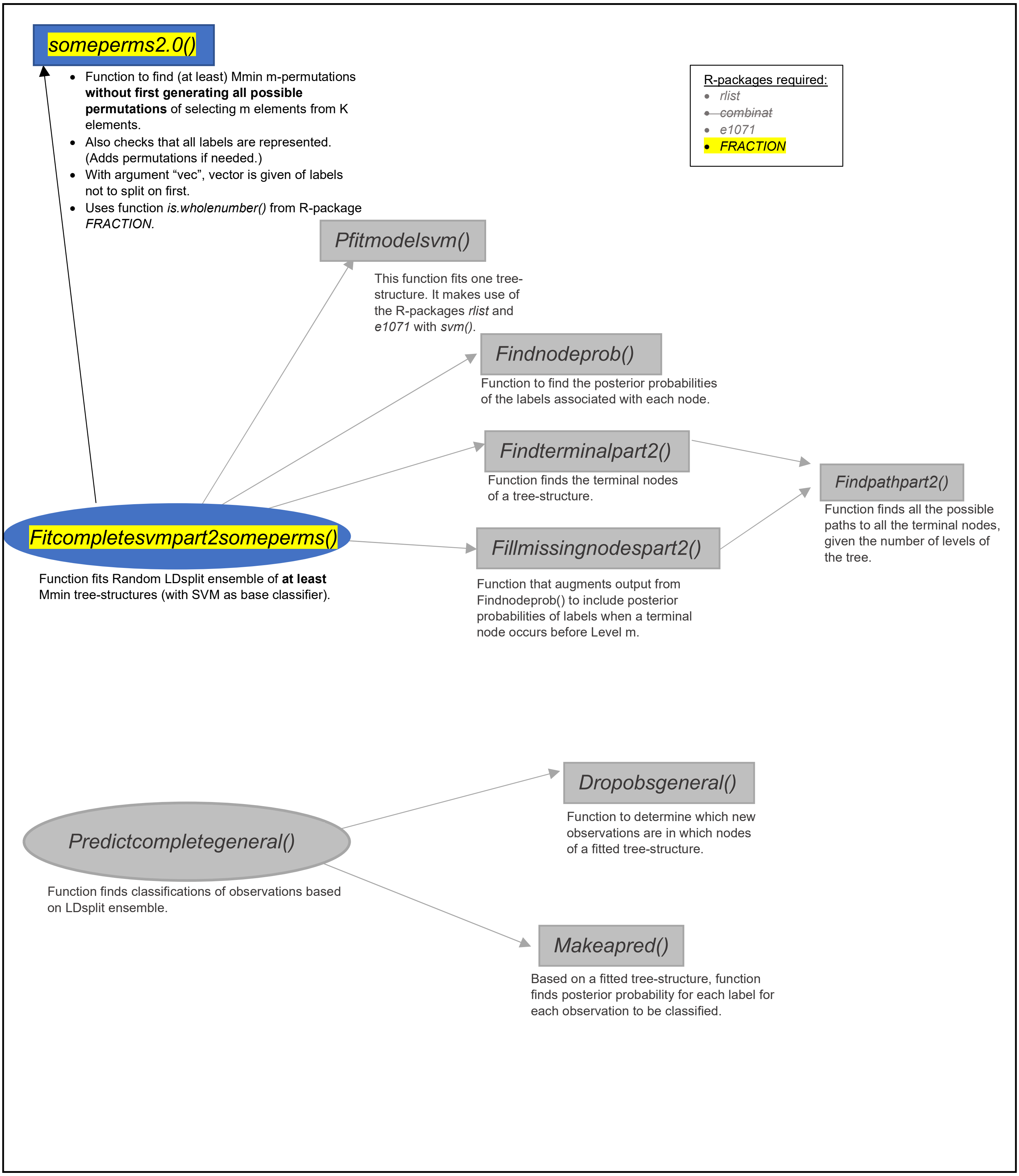


Fitcompletetreepart2order:

```
Fitcompletetreepart2order<-function(Xmat,Ymat,K,M,m,minsize,labelthr)
{
########################################################################
# Function fits Conditional LDsplit ensemble of M tree-structures
# (with decision tree as base classifier).
# Inputs
#     Xmat = the matrix of input values
#     Ymat = the matrix of 0-1 label vectors
#     K = total number of labels
#     M = number of label orders to use (ensemble size)
#     m = number of levels
#     minsize = minimum node-size
#     labelthr = vector of size K containing threshold to apply
#                for each label
# Outputs:
#    keepmodels = list of all models used
#    keepnodeprobs = matrix where each row is node probabilities
#                    of a label order
#    theperms = the matrix of label orders used
#    splityesno = matrix where each row gives code if node was split
#                 or not split (1 = split, -1 = no split)
#######################################################################
p<-ncol(Xmat)

#Sample M orders using CONDITIONAL ENTROPY
theorders<-matrix(0,nrow=M,ncol=m)
allorders<-conditionalorder(labelset=Ymat,m,K)
numberoforders<-choose(K,m)
ordersused<-sample(c(1:numberoforders),M)


for(k in 1:M){
theorders[k,]<-allorders[ordersused[k],]}


#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
        #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
        #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)
```

```
for(j in 1:M){

#Use Pfitmodel for each label order
out<-Pfitmodeltree(Xmat,Ymat,theorders[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]

#Find node probabilities for a label ordering
nodeprobsout<-Findnodeprob(Ymat,theorders[j,],out[[2]],m)

#Find terminal nodes of a label ordering
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theorders[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
       #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
       #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theorders,splityesno))
}
```

C.7 Functions to apply scaling techniques when fitting Random LDsplit with an SVM base classifier



someperms2.0:

```
someperms2.0<-function(K,m,limit,vec)
{
##########################################################################
# Function to find (at least) Mmin m-permutations without first generating
# all possible permuations of selecting m elements from K elements.
# Also checks that all labels are represented (adds permutations if
# needed) and allows for specification of labels not to be split on first.
# Function makes use of function "is.wholenumber" from package FRACTION.
# Inputs:
#     K = total number of labels
#     m = number of labels selected per permutation
#     limit = minimum ensemble size (Mmin)
#     vec = vector of labels that cannot be split on first
# Output:
#     outmat = matrix with each row forming a label ordering
########################################################################
alllabels<-c(1:K)
themat<-matrix(0,nrow=limit,ncol=m-1)
for (i in 1:limit){
themat[i,]<-sample(alllabels,m-1,replace=FALSE)}
          #initially excluding specification of first label in ordering

theset<-unique(sort(as.vector(themat)))
notincluded<-setdiff(c(1:K),theset)

#Missing labels
if(length(notincluded)>0){
included<-setdiff(c(1:K),notincluded)
howmany<-(length(notincluded))/(m-1)

if(is.wholenumber(howmany)){
extrarows<-howmany
extramat<-matrix(notincluded,nrow=extrarows,ncol=m-1,byrow=T)
finalmat<-rbind(themat,extramat)
}else{


extrarows<-ceiling(howmany)
total<-extrarows*(m-1)
need<-(total-(length(notincluded)))
get<-sample(included,need,replace=FALSE)
extramat<-matrix(c(notincluded,get),nrow=extrarows,ncol=m-1,byrow=T)
finalmat<-rbind(themat,extramat)
}
```

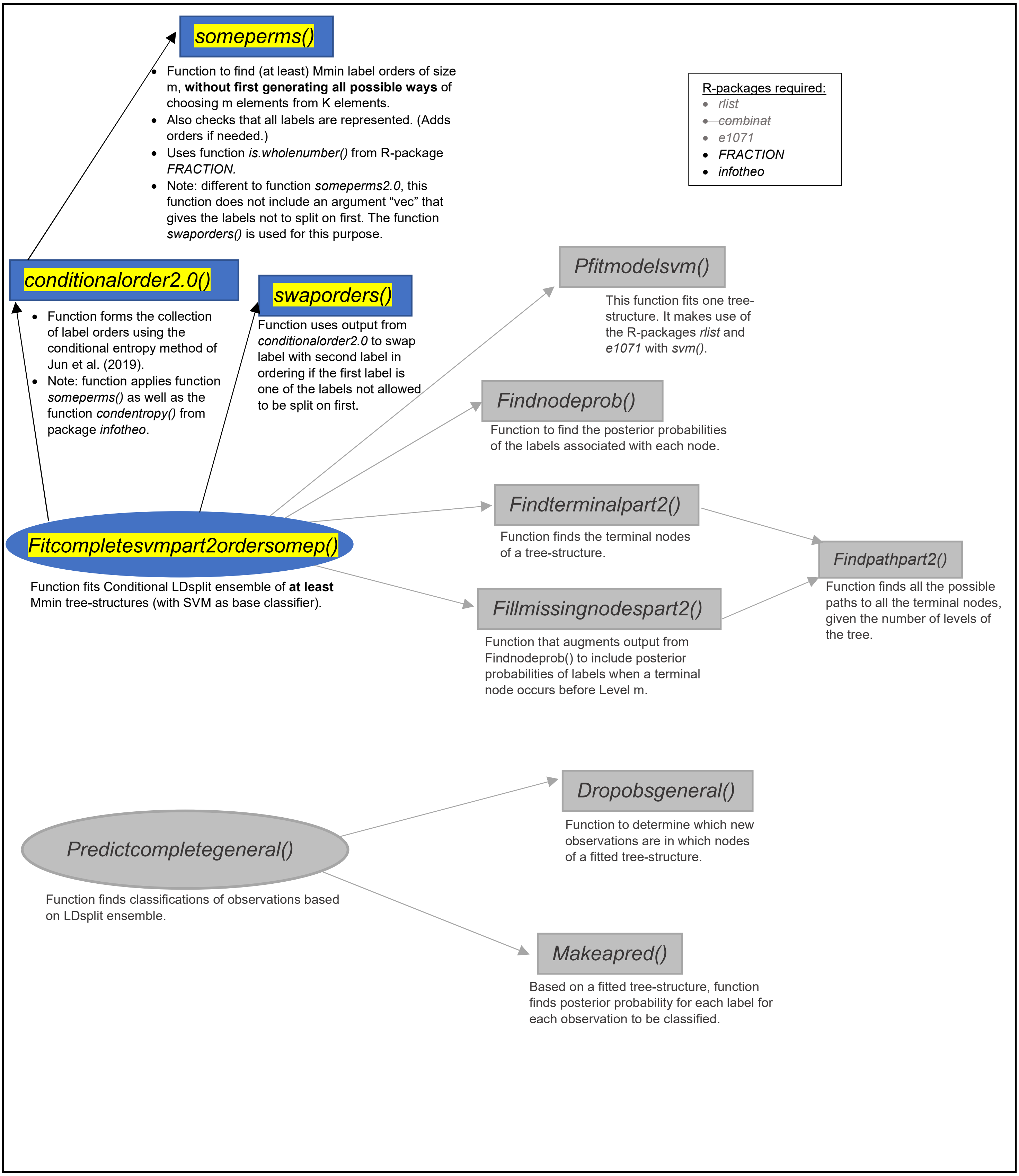
```
}else{
finalmat<-themat
}

### Adding first label
finaltotal<-nrow(finalmat)
firstcolumn<-matrix(0,nrow=finaltotal,ncol=1)
for(j in 1:finaltotal){
pool1<-setdiff(c(1:K),vec)
pool2<-setdiff(pool1,finalmat[j,])
firstcolumn[j,]<-sample(pool2,1)}

outmat<-cbind(firstcolumn,finalmat)
return(outmat)
}
```

Fitcompletesvmpart2someperms:

```
Fitcompletesvmpart2someperms<-function(Xmat,Ymat,K,M,m,minsize,
labelthr,vec)
{
########################################################################
# Function fits Random LDsplit ensemble of at least Mmin tree-structures
# (with SVM as base classifier).
# Inputs:
#      Xmat = the matrix of input values
#      Ymat = the matrix of 0-1 label vectors
#      K = total number of labels
#      M = minimum number of permutations to use (Mmin)
#      m = number of levels
#      minsize = minimum node-size
#      labelthr = vector of size K containing threshold to apply for
#                 each label
#      vec = vector giving the numbers of the labels not to split on first
# Outputs:
#     keepmodels = list of all models used
#     keepnodeprobs = matrix where each row is node probabilities
#                     of a permutation
#     theperms = the matrix of permutations used
#     splityesno = matrix where each row gives code if node was split
#                  or not split (1 = split, -1 = no split)
#     realM = true number of permutations used
#########################################################################
p<-ncol(Xmat)

#Sample at least Mmin permutations using someperms2.0
theperms<-someperms2.0(K,m,M,vec)
realM<-nrow(theperms)
```

```
#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",realM)
        #keep the binary models
keepnodeprobs<-matrix(0,nrow=realM,ncol=nnodes)
        #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=realM,ncol=nsplits)

for(j in 1:realM){

#Use Pfitmodelsvm for each permutation
out<-Pfitmodelsvm(Xmat,Ymat,theperms[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]

#Find node probabilities for a permutation
nodeprobsout<-Findnodeprob(Ymat,theperms[j,],out[[2]],m)

#Find terminal nodes of a permutation
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theperms[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
        #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
        #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theperms,splityesno,realM))
}
```

C.8 Functions to apply scaling techniques when fitting Conditional LDsplit with an SVM base classifier



someperms:

```
someperms<-function(K,m,limit)
{
##########################################################################
# Function to find (at least) Mmin label orders of size m, without first
# generating all possible ways of choosing m elements from K elements.
# Function also checks that all labels are represented (adds orders if
# needed). The function "is.wholenumber" is used from package FRACTION.
# Note: different to the function "someperms2.0", this function does
# not include an argument "vec" that gives the labels not to split on
# first, "swaporders" is used for that purpose.
# Inputs:
#   K = total number of labels
#   m = number of labels selected per label order
#   limit = minimum ensemble size (Mmin)
# Outputs:
#   finalmat = matrix with each row giving a label ordering
##########################################################################
alllabels<-c(1:K)
themat<-matrix(0,nrow=limit,ncol=m)
for (i in 1:limit){
themat[i,]<-sample(alllabels,m,replace=FALSE)}

theset<-unique(sort(as.vector(themat)))
notincluded<-setdiff(c(1:K),theset)

#Missing labels
if(length(notincluded)>0){
included<-setdiff(c(1:K),notincluded)
howmany<-(length(notincluded))/m

if(is.wholenumber(howmany)){
extrarows<-howmany
extramat<-matrix(notincluded,nrow=extrarows,ncol=m,byrow=T)
finalmat<-rbind(themat,extramat)
}else{

extrarows<-ceiling(howmany)
total<-extrarows*m
need<-(total-(length(notincluded)))
get<-sample(included,need,replace=FALSE)
extramat<-matrix(c(notincluded,get),nrow=extrarows,ncol=m,byrow=T)
finalmat<-rbind(themat,extramat)
}
}else{
finalmat<-themat
}
return(finalmat)
}
```

conditionalorder2.0:

```
conditionalorder2.0<-function(labelset,m,K,limit)
{
######################################################################
# Function forms the collection of label orders using the conditional
# entropy method of Jun et al. (2019).
# Note: function uses the written function someperms() as well as the
# function condentropy() from package infotheo.
# Inputs:
#     labelset = matrix of 0-1 label vectors
#     m = number of levels for LDsplit tree-structures
#     K = total number of labels
#     limit = minimum ensemble size (Mmin)
# Ouputs:
#     theordersmat = matrix with each row giving a label ordering
#####################################################################
allcomb<-someperms(K,m,limit)  #written function
howmany<-nrow(allcomb)

#Finding orders
theordersmat<-matrix(0,nrow=howmany,ncol=m)
if(m>2){
for(h in 1:howmany){

#Per order:
#Labels that are considered:
if(howmany==1){
thecurrentlabels<-allcomb}
if(howmany>1){
thecurrentlabels<-allcomb[h,]}
#Finding matrix of conditional entropies:
thecurrentlabelset<-labelset[,thecurrentlabels]
entropymat<-matrix(0,nrow=m,ncol=m)
for(i in 1:m){
for(j in 1:m){
entropymat[i,j]<-condentropy(thecurrentlabelset[,j],
thecurrentlabelset[,i])}}
tel<-0
usingmat<-entropymat
thelabels<-thecurrentlabels
for(k in 1:(m-2)){
thesums<-apply(usingmat,1,sum)
ith<-which(thesums==min(thesums))
if(length(ith)==1){
theordersmat[h,m-tel]<-thelabels[ith]
tel<-tel+1
usingmat<-usingmat[,-ith]
usingmat<-usingmat[-ith,]
thelabels<-thelabels[-ith]
}else{
```

```
ith<-ith[1]
theordersmat[h,m-tel]<-thelabels[ith]
tel<-tel+1
usingmat<-usingmat[,-ith]
usingmat<-usingmat[-ith,]
thelabels<-thelabels[-ith]}
}
#m-1
thesums<-apply(usingmat,1,sum)
ith<-which(thesums==min(thesums))
other<-which(thesums==max(thesums))
if(length(ith)==1){
theordersmat[h,m-tel]<-thelabels[ith]
theordersmat[h,m-tel-1]<-thelabels[other]
}else{
theordersmat[h,m-tel]<-thelabels[ith[1]]
theordersmat[h,m-tel-1]<-thelabels[other[2]]}
}                                        #howmany
}                                        #if
if(m<=2){
for(h in 1:howmany){
#Per order:
#Labels that are considered:
if(howmany==1){
thecurrentlabels<-allcomb}
if(howmany>1){
thecurrentlabels<-allcomb[h,]}
#Finding matrix of conditional entropies:
thecurrentlabelset<-labelset[,thecurrentlabels]
entropymat<-matrix(0,nrow=m,ncol=m)
for(i in 1:m){
for(j in 1:m){
entropymat[i,j]<-condentropy(thecurrentlabelset[,j],
thecurrentlabelset[,i])}}
tel<-0
usingmat<-entropymat
thelabels<-thecurrentlabels
#m-1
thesums<-apply(usingmat,1,sum)
ith<-which(thesums==min(thesums))
other<-which(thesums==max(thesums))
if(length(ith)==1){
theordersmat[h,m-tel]<-thelabels[ith]
theordersmat[h,m-tel-1]<-thelabels[other]
}else{
theordersmat[h,m-tel]<-thelabels[ith[1]]
theordersmat[h,m-tel-1]<-thelabels[other[2]]}
}                                        #howmany
}                                        #if
return((theordersmat))
}
```

swaporders:

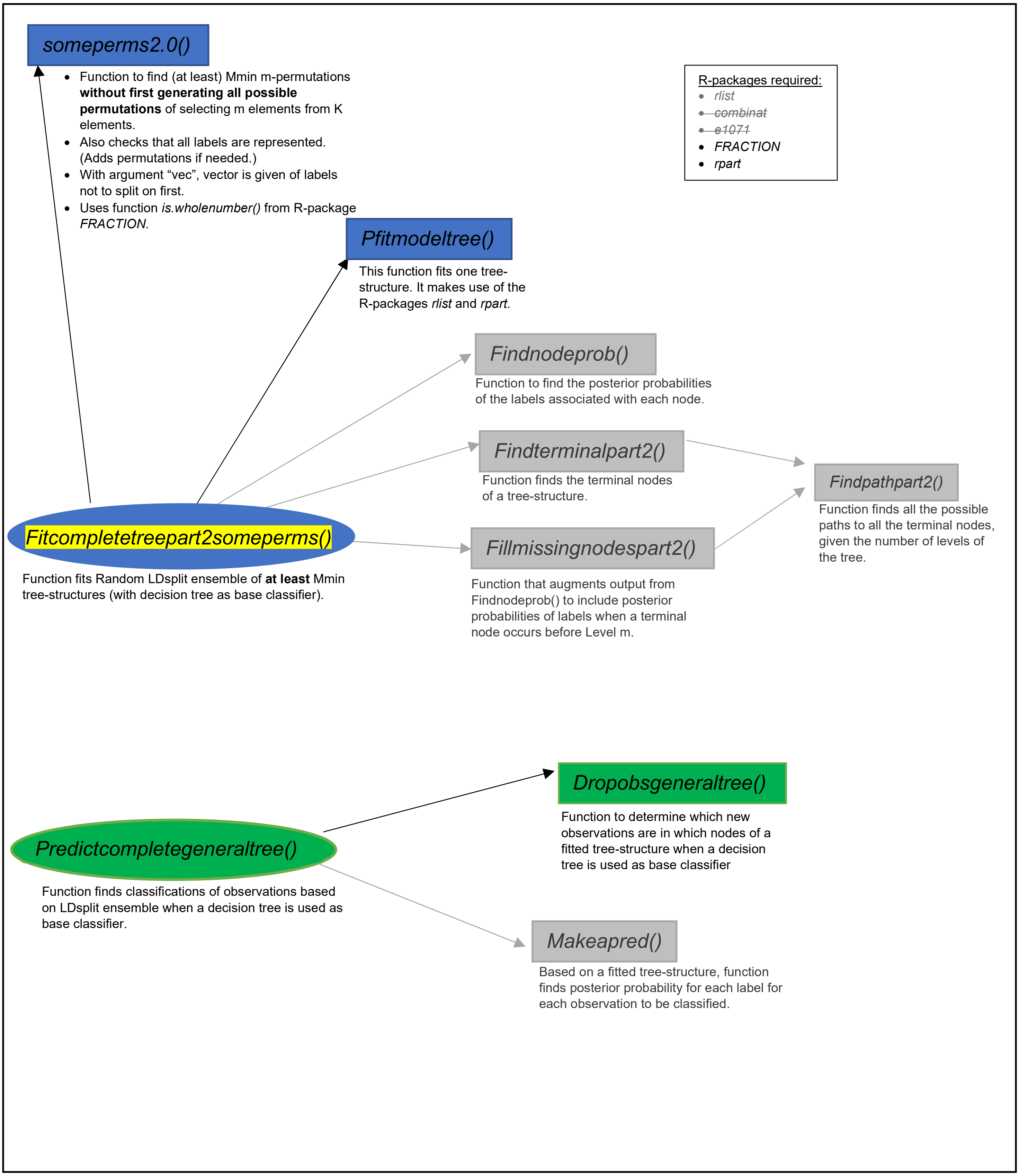
```
swaporders<-function(ordermat,vec)
{
##########################################################################
# Function uses output from conditionalorder2.0() to swap label with
# second label in ordering if the first label is one of the labels not
# allowed to be split on first.
# Input:
#   ordermat = matrix giving conditional label orders
#              (output from conditionalorder)
#   vec = vector giving labels not to split on first
# Output:
#   newordermat = matrix with final label orders given as rows
##########################################################################
howmany<-nrow(ordermat)
veclength<-length(vec)
newordermat<-ordermat

#which orders are problematic
longvec<-c(0)
for(k in 1:veclength){
longvec<-c(longvec,which(ordermat[,1]==vec[k]))
}
longvec<-longvec[-1]

#swapping them
swaplength<-length(longvec)
for(k in 1:swaplength){
rowindex<-longvec[k]
rowsorder<-ordermat[rowindex,]
original1<-rowsorder[1]
original2<-rowsorder[2]
neworder<-c(original2,original1,rowsorder[-c(1,2)])
newordermat[rowindex,]<-neworder}


#deleting if new orders still have wrong labels first
longvec2<-c(0)
for(k in 1:veclength){
longvec2<-c(longvec2,which(newordermat[,1]==vec[k]))
}
longvec2<-longvec2[-1]

if( (length(longvec2)) >=1){
newordermat<-newordermat[-longvec2,]
}else{
newordermat<-newordermat}

return(newordermat)
}
```

Fitcompletesvmpart2ordersomep:

```
Fitcompletesvmpart2ordersomep<-function(Xmat,Ymat,K,M,m,minsize,
labelthr,vec)
{
######################################################################
# Function fits Conditional LDsplit ensemble of at least Mmin
# tree-structures (with SVM as base classifier).
# Inputs:
#   Xmat = the matrix of input values
#   Ymat = the matrix of 0-1 label vectors
#   K = total number of labels
#   M = minimum number of permutations to use (Mmin)
#   m = number of levels
#   minsize = minimum node-size
#   labelthr = vector of size K containing threshold to apply
#                for each label
#   vec = vector giving the numbers of the labels not to split on first
# Outputs:
#   keepmodels = list of all models used
#   keepnodeprobs = matrix where each row is node probabilities
#                    of a permutation
#   theperms = the matrix of label orders used
#   splityesno = matrix where each row gives code if node was split
#                 or not split (1 = split, -1 = no split)
#   Mnew = true number of label orders used (true ensemble size)
###################################################################
p<-ncol(Xmat)

#Sample orders using CONDITIONAL ENTROPY
ordermat<-conditionalorder2.0(Ymat,m,K,M)
theorders<-swaporders(ordermat,vec)
Mnew<-nrow(theorders)

#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",Mnew)
        #keep the binary models
keepnodeprobs<-matrix(0,nrow=Mnew,ncol=nnodes)
        #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=Mnew,ncol=nsplits)

for(j in 1:Mnew){
#Use Pfitmodelsvm for each order
out<-Pfitmodelsvm(Xmat,Ymat,theorders[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]
```

```
nodeprobsout<-Findnodeprob(Ymat,theorders[j,],out[[2]],m)

#Find terminal nodes of a ordering
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theorders[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
        #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
        #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theorders,splityesno,Mnew))
}
```

C.9 Functions to apply scaling techniques when fitting Random LDsplit with a decision tree base classifier

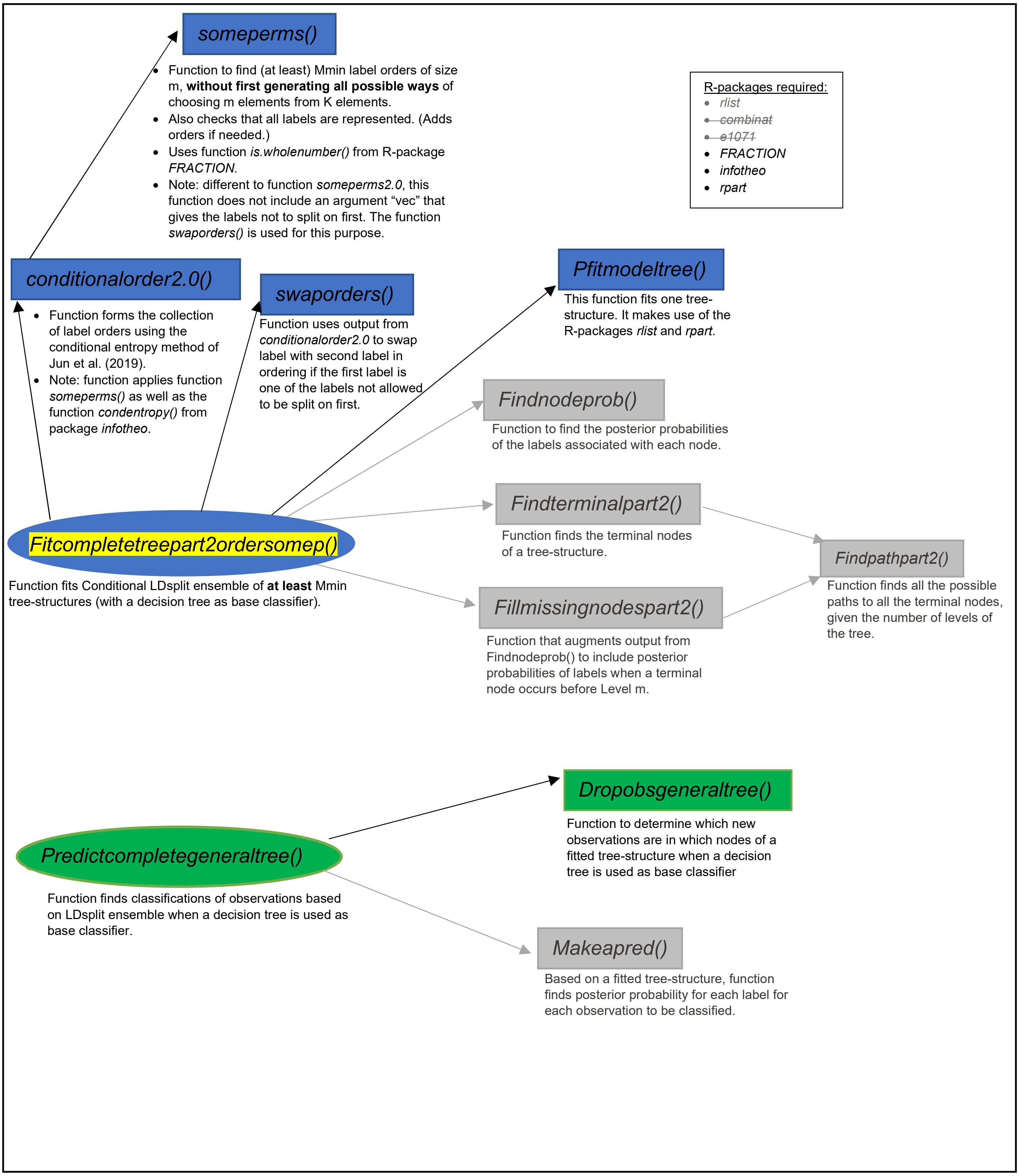


Fitcompletetreepart2someperms:

```
Fitcompletetreepart2someperms<-function(Xmat,Ymat,K,M,m,minsize,
labelthr,vec)
{
########################################################################
# Function fits Random LDsplit ensemble of at least Mmin tree-structures
# (with a decision tree as base classifier).
# Inputs:
#    Xmat = the matrix of input values
#    Ymat = the matrix of 0-1 label vectors
#    K = total number of labels
#    M = minimum number of permutations to use (Mmin)
#    m = number of levels
#    minsize = minimum node-size
#    labelthr = vector of size K containing threshold to apply
#                for each label
#    vec = vector giving the numbers of the labels not to split on first
# Outputs:
#    keepmodels = list of all models used
#    keepnodeprobs = matrix where each row is node probabilities
#                    of a permutation
#    theperms = the matrix of permutations used
#    splityesno = matrix where each row gives code if node was split
#                 or not split (1 = split, -1 = no split)
#    realM = true number of permutations used
#######################################################################
p<-ncol(Xmat)

#Sample (at least) Mmin permutations using someperms2.0
theperms<-someperms2.0(K,m,M,vec)
realM<-nrow(theperms)


#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",realM)
      #keep the binary models
keepnodeprobs<-matrix(0,nrow=realM,ncol=nnodes)
      #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=realM,ncol=nsplits)

for(j in 1:realM){

#Use Pfitmodeltree for each permutation
out<-Pfitmodeltree(Xmat,Ymat,theperms[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]

#Find node probabilities for a permutation
nodeprobsout<-Findnodeprob(Ymat,theperms[j,],out[[2]],m)
```

```
#Find terminal nodes of a permutation
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theperms[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
      #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
      #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theperms,splityesno,realM))
}
```

C.10 Functions to apply scaling techniques when fitting Conditional LDsplit with a decision tree base classifier



Fitcompletetreepart2ordersomep:

```
Fitcompletetreepart2ordersomep<-function(Xmat,Ymat,K,M,m,minsize,
labelthr,vec)
{
########################################################################
# Function fits Conditional LDsplit ensemble of at least Mmin
# tree-structures (with a decision tree as base classifier).
# Inputs:
#   Xmat = the matrix of input values
#   Ymat = the matrix of 0-1 label vectors
#   K = total number of labels
#   M = minimum number of label orders to use (Mmin)
#   m = number of levels
#   minsize = minimum node-size
#   labelthr = vector of size K containing threshold to apply
#              for each label
#   vec = vector giving the numbers of the labels not to split on first
# Outputs:
#   keepmodels = list of all models used
#   keepnodeprobs = matrix where each row is node probabilities
#                   of a label order
#   theperms = the matrix of label orders used
#   splityesno = matrix where each row gives code if node was split
#                or not split (1 = split, -1 = no split)
#######################################################################
p<-ncol(Xmat)

#Sample orders using CONDITIONAL ENTROPY
ordermat<-conditionalorder2.0(Ymat,m,K,M)
theorders<-swaporders(ordermat,vec)
Mnew<-nrow(theorders)


#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",Mnew)
       #keep the binary models
keepnodeprobs<-matrix(0,nrow=Mnew,ncol=nnodes)
       #keep the posterior probabilities of each node (excluding first)
splityesno<-matrix(0,nrow=Mnew,ncol=nsplits)

for(j in 1:Mnew){

#Use Pfitmodeltree for each order
out<-Pfitmodeltree(Xmat,Ymat,theorders[j,],minsize,m,labelthr)
splityesno[j,]<-out[[5]]

#Find node probabilities for a label ordering
nodeprobsout<-Findnodeprob(Ymat,theorders[j,],out[[2]],m)
```

```
#Find terminal nodes of a label ordering
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theorders[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
      #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
      #keep the posterior probabilities of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theorders,splityesno,Mnew))
}
```

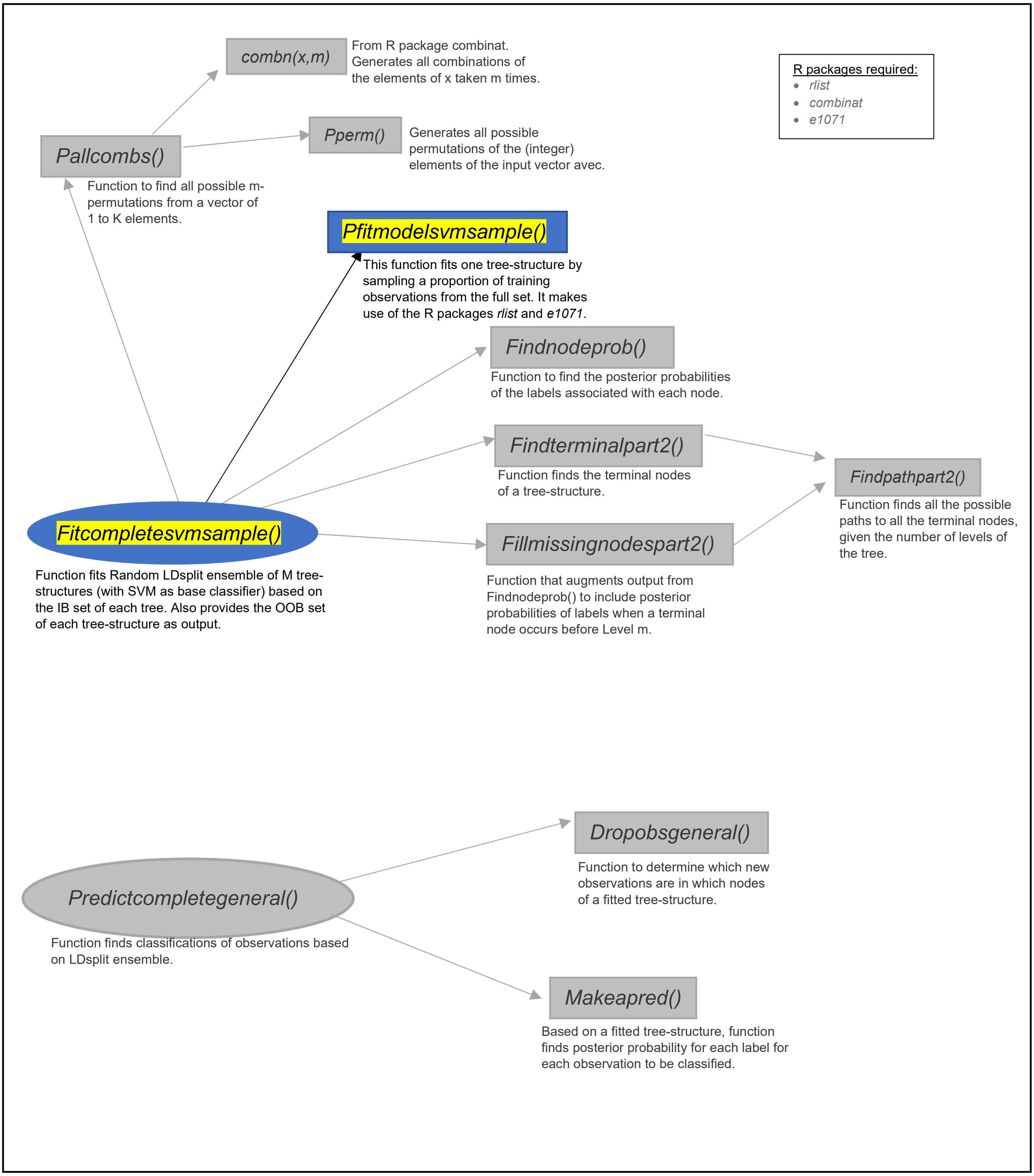
Appendix D: R-functions for LDsplit MDA introduced in Chapter 5

This appendix contains all written R-functions to fit LDsplit MDA as introduced in Chapter 5.

Appendix D.1 contains two functions required to apply the additional step of subsampling $prop_{IB}(N)$ training observations before fitting each of $T_j,\ j=1,\ldots,M$, tree-structures of the Random LDsplit ensemble (Section 5.5.1). To summarise the interaction between the functions, a diagram similar to those provided in Appendix C is given.

Appendix D.2 contains the four functions required to obtain the global and local LDsplit MDA input variable rankings as described in Section 5.5.1.

D.1 Functions to fit Random LDsplit with an SVM base classifier when the adaptation for LDsplit MDA is implemented



Pfitmodelsvmsample:

```
Pfitmodelsvmsample<-function(Xmat,Ymat,yperm,minsize,m,labelthr,propn)
{
########################################################################
# This function fits a Random LDsplit tree-structure to a given dataset
# by sampling propn observations from the full training dataset.
# Inputs:
#  Xmat = the matrix of input values
#  Ymat = the matrix of 0-1 label vectors
#  yperm = the vector defining the label order of the tree-structure
#  minsize = minimum node-size
#  m = number of levels of the tree-structure
#  labelthr = vector of size K containing threshold to apply
#             for each label
#  propn = proportion of observations to be sampled  (value between
#          0 and 1)  #NEW
#
# Outputs:
#  binarymodels = list of binary models used to perform the splitting
#  obsinnodesdummy = a list containing the indices of the cases in
#                    all of the nodes
#  keepprobs = posterior probabilities of training observations
#              in each node
#  emptyyesno = vector with -1 if node is empty, and 1 if node is
#               not empty
#  splityesno = vector with 1 if node was split, and -1 if node was
#               not split
#  oob = set of out-of-bag observations   #NEW
########################################################################
 n<-nrow(Xmat)
 p<-ncol(Xmat)
 K<-ncol(Ymat)

### NEW: sample observations ###
numn<-round(propn*n,0)
inbag<-sort(sample(c(1:n),numn))
oob<-setdiff(c(1:n),inbag)

### New "Xmat" and "Ymat" ###
n<-numn
Xmat<-Xmat[inbag,]
Ymat<-Ymat[inbag,]

binarymodels<-vector("list",(2^m)-1)
                     #(2^m)-1 splits/binary classifiers for a tree

splityesno<-rep(0,(2^m)-1)
```

```
Ymatperm<-matrix(0,nrow=n,ncol=m)
Ythr<-rep(0,m)
 for (k in 1:m) Ymatperm[,k] = Ymat[,yperm[k]]  #organise labels
 for (k in 1:m) Ythr[k] = labelthr[yperm[k]]    #organise thresholds

######################
## Root node split  ##
######################
xmat<-Xmat
yvec<-Ymatperm[,1]
thresh<-Ythr[1]

uit1<-svm(xmat,factor(yvec),probability=TRUE)  #fit a binary classifier
binarymodels[[1]]<-uit1
emptyyesno<-c(1)
splityesno[1]<-1

predictions<-predict(uit1,xmat,probability=TRUE)
probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
                     #making sure correct posterior probability is used

classes<-ifelse(probs>=thresh,1,0)          #Classifying training cases
ygroup1<-as.numeric(which(classes>0.5))
ygroup0<-as.numeric(which(classes<0.5))
              #Split the x-values based on the classifier and threshold

obsinnodes = list(1:n, ygroup1, ygroup0)
              #Observations in root/node 1, node 2 and node 3

obsinnodesdummy= list(inbag,inbag[ygroup1],inbag[ygroup0])    #NEW
keepprobs<-list(probs[ygroup1],probs[ygroup0])
emptyyesno<-c(emptyyesno,ifelse(length(ygroup1)>0,1,-1),
ifelse(length(ygroup0)>0,1,-1))


######################
## Level k's splits ##
######################
 for (k in 1:(m-1)) {
    yvec = Ymatperm[,k+1]
    thresh = Ythr[k+1]
    beginn = 2^k                #first node number of level k
    eindig = 2^(k+1) -1         #last node number of level k
    for (t in beginn:eindig) {
       indekse = obsinnodes[[t]]
       if ( (length(indekse)>minsize) && (var(yvec[indekse])>0) ){
          uit1 = svm(xmat[indekse,],factor(yvec[indekse]),
probability=TRUE)
              #fit a binary classifier

          binarymodels[[t]]<-uit1
```

```
          predictions<-predict(uit1,xmat[indekse,],probability=TRUE)
          probs<-(attr(predictions,"probabilities"))
[,which(as.numeric(colnames(attr(predictions,"probabilities")))==1)]
              #making sure correct posterior probability is used

          classes<-ifelse(probs>=thresh,1,0)
                                   #Classifying training cases

          ygroup1<-as.numeric(which(classes>0.5))
          ygroup0<-as.numeric(which(classes<0.5))
              #Split the x-values based on the classifier and threshold

          obsinnodes = list.append(obsinnodes,obsinnodes[[t]]
[ygroup1],obsinnodes[[t]][ygroup0])
              #Observations in previous nodes plus resulting two

          obsinnodesdummy = list.append(obsinnodesdummy,
obsinnodesdummy[[t]][ygroup1],obsinnodesdummy[[t]][ygroup0])
                                                            #NEW

          keepprobs = list.append(keepprobs,probs[ygroup1],probs[ygroup0])
            #Posterior probabilities in previous nodes plus resulting two

          emptyyesno<-c(emptyyesno,ifelse(length(ygroup1)>0,1,-1),
ifelse(length(ygroup0)>0,1,-1))
          splityesno[t]<-1
                                                                }
       if ( (length(indekse)<=minsize) || (var(yvec[indekse])==0) ) {
          binarymodels[[t]]<--1
          emptyyesno<-c(emptyyesno,-1,-1)
          obsinnodes = list.append(obsinnodes,-1,-1)
                  #No observations in resulting nodes

          obsinnodesdummy = list.append(obsinnodesdummy,-1,-1)
                                                             #NEW


          keepprobs = list.append(keepprobs,-1,-1)
          splityesno[t]<--1
                                                  }
                                       }
                             }
uit = list(binarymodels,obsinnodesdummy,keepprobs,emptyyesno,
splityesno,oob)
return(uit)
}
```

Fitcompletesvmsample:

```
Fitcompletesvmsample<-function(Xmat,Ymat,K,M,m,minsize,labelthr,propn)
{
########################################################################
# Inputs:
#      Xmat = the matrix of input values
#      Ymat = the matrix of 0-1 label vectors
#      K = total number of labels
#      M = number of permutations to use (size of ensemble)
#      m = number of levels
#      minsize = minimum node-size
#      labelthr = vector of size K containing threshold to apply for
#                 each label
#      propn = proportion of observations to be sampled (value between
#              0 and 1) #NEW
# Outputs:
#     keepmodels = list of all models used
#     keepnodeprobs = matrix where each row is node probabilities
#                     of a permutation
#     theperms = the matrix of permutations used
#     splityesno = matrix where each row gives code if node was split or
#                  not split(1 = split, -1 = no split)
#     OOBlist = list giving set of out-of-bag observations for each
#               tree-structure #NEW
########################################################################
p<-ncol(Xmat)

#Sample M permutations using Pallcombs
theperms<-matrix(0,nrow=M,ncol=m)
allperms<-Pallcombs(K,m)
numberofperms<-nrow(allperms)
permsused<-sample(c(1:numberofperms),M)

for(k in 1:M){
theperms[k,]<-allperms[permsused[k],]}


#Keep
nsplits<-(2^m)-1
nnodes<-(2^(m+1))-2
keepmodels<-vector("list",M)
              #keep the binary models
keepnodeprobs<-matrix(0,nrow=M,ncol=nnodes)
              #keep the posterior probs of each node (excluding first)
splityesno<-matrix(0,nrow=M,ncol=nsplits)
OOBlist<-vector("list",M)         #NEW

for(j in 1:M){
```

```
#Use Pfitmodelsvmsample for each permutation
out<-Pfitmodelsvmsample(Xmat,Ymat,theperms[j,],minsize,m,labelthr,propn)
splityesno[j,]<-out[[5]]
OOBlist[[j]]<-out[[6]]            #NEW

#Find node probabilities for a permutation
nodeprobsout<-Findnodeprob(Ymat,theperms[j,],out[[2]],m)

#Find terminal nodes of a permutation
terminalout<-Findterminalpart2(out[[4]],m)$terminals
pathindout<-Findterminalpart2(out[[4]],m)$pathindices
fillednodeprobsout<-Fillmissingnodespart2(pathindout,terminalout,
Ymat,theperms[j,],m,out[[2]],nodeprobsout)

keepmodels[[j]]<-out[[1]]
              #keep the binary models
keepnodeprobs[j,]<-fillednodeprobsout
              #keep the posterior probs of each node (excluding first)
}
return(list(keepmodels,keepnodeprobs,theperms,splityesno,OOBlist))
}
```

D.2 Functions to obtain global and local LDsplit MDA input variable rankings

permimportance:

```
permimportance<-function(Xmat,modelsused,nodeprobs,perms,splityesno,M,K,
m,labelthr,OOBlist,treenum)
{
##########################################################################
# This function is applied to one tree-structure. It permutes each input
# variable in turn and finds OOB classifications for the permuted set.
# Inputs:
#   Xmat = original matrix of input values
#   modelsused = list of trained models used to split data
#   nodeprobs = matrix where each row is node probabilities
#               of a permutation
#   perms = matrix where each row gives permutation used
#   splityesno = matrix where each row gives code if node was split or
#                not split (1 = split, -1 = no split)
#   M = number of permutations (size of ensemble)
#   K = total number of labels
#   m = total number of levels
#   labelthr = vector of size K containing threshold to apply for
#              each label
#   OOBlist = list with each element the OOB-set for each tree
#   treenum = number of tree in the ensemble
# Output:
#   newpredictions = Nnew*m*p array of OOB classifications
#########################################################################
P<-ncol(Xmat)
Nnew<-length(OOBlist[[treenum]])
OOB<-OOBlist[[treenum]]
Xmatnew<-Xmat[OOB,]

#Keep predictions
newpredictions<-array(-1,dim=c(Nnew,m,P))


#Permute a variable
for(p in 1:P){
Xmatpermuted<-Xmatnew
permutings<-sample(c(1:Nnew),Nnew,replace=FALSE)
Xmatpermuted[,p]<-Xmatpermuted[permutings,p]

#Drop observations in tree-structure
obslist<-Dropobsgeneral(Xmatpermuted,modelsused[[treenum]],m,
perms[treenum,],labelthr,splityesno[treenum,])
```

```
#Find predictions of permutations
thepredictions<-Makeapred(obslist,nodeprobs[treenum,],K,Nnew,
perms[treenum,],m)

thepredictions<-thepredictions[,sort(perms[treenum,])]
thepredictions<-ifelse(thepredictions>=0.5,1,0)
newpredictions[,,p]<-thepredictions
}
return(newpredictions)
}
```

Pmeasures:

```
Pmeasures<-function(ylabels,zlabels)
{
##########################################################################
# This program computes various multi-label evaluation measures
# Inputs:
#  ylabels = an indicator matrix containing the true labels for a set
#            of Nnew new cases
#  zlabels = an indicator matrix containing the predicted labels for
#            the Nnew new cases
##########################################################################
# The following measures are computed:
#     1. Hamming loss (Hloss)
#     2. Accuracy (accuracy)
#     3. Precision (precision)
#     4. Recall (recall)
#     5. Fscore  (2*precision*recall)/(precision+recall)
#     6. Subset accuracy / classification accuracy (subsetacc)
##########################################################################
Nnew<-nrow(ylabels)
K<-ncol(ylabels)

#Hamming loss
yminz<-ylabels-zlabels
ydeltaz<-apply(yminz,1,function(x) sum(abs(x)))
Hloss<-(sum(ydeltaz))/(Nnew*K)

#Accuracy
ymaalz<-ylabels*zlabels
ysnydingz<-apply(ymaalz,1,sum)
yplusz<-ylabels+zlabels
yverenigz<-apply(ifelse(yplusz>=1,1,0),1,sum)
accuracy<-mean((ysnydingz/yverenigz))
```

```
#Precision
zabs<-apply(zlabels,1,sum)
#precision<-mean(ysnydingz/zabs)
precision<-mean((ysnydingz/zabs)[!is.nan(ysnydingz/zabs)])

#Recall
yabs<-apply(ylabels,1,sum)
#recall<-mean(ysnydingz/yabs)
recall<-mean((ysnydingz/yabs)[!is.nan(ysnydingz/yabs)])

#Fscore
Fscore<-(2*precision*recall)/(precision+recall)

#Subset accuracy /Classification accuracy
subsetacc<-sum(ydeltaz==0)/Nnew

afvoer<-list(Hloss=Hloss,accuracy=accuracy,precision=precision,
recall=recall,Fscore=Fscore,subset.accuracy=subsetacc)

return(afvoer)
}
```

calculateGLOBALimportance:

```
calculateGLOBALimportance<-function(Xmat,Ymat,modelsused,nodeprobs,perms,
splityesno,M,K,m,labelthr,OOBlist)
{
#########################################################################
# Function used to calculate global LDsplit MDA variable importance
# (as outlined in Section 5.5.1)
# Inputs:
#   Xmat = original matrix of input values
#   Ymat = original matrix of 0-1 label vectors
#   modelsused = list of trained models used to split the observations
#   nodeprobs = matrix where each row is node probabilities of
#               a label order
#   perms = matrix where each row gives label order used
#   splityesno = matrix where each row gives code if node was split or
#                not split (1 = split, -1 = no split)
#   M = number of label orders (size of ensemble)
#   K = total number of labels
#   m = total number of levels
#   labelthr = vector of size K containing threshold to apply
#              for each label

#   OOBlist = list where each element is the OOB-set for each tree

# Output:
#   GlobalimportanceHL = vector of global importance per variable based
#                        on Hamming loss
#   GlobalimportanceFscore = vector of global importance per variable
#                            based on F-score
#########################################################################
P<-ncol(Xmat)

#Global Hamming loss and F-score calculation
OOBHL<-rep(0,M)
OOBFscore<-rep(0,M)
permutedresultsHL<-matrix(0,nrow=P,ncol=M)
permutedresultsFscore<-matrix(0,nrow=P,ncol=M)
GlobaldiffHL<-matrix(0,nrow=P,ncol=M)
GlobaldiffFscore<-matrix(0,nrow=P,ncol=M)

#Per tree
for(j in 1:M){

### True OOB classification of tree ###
OOB<-OOBlist[[j]]
ms<-sort(perms[j,])
Ytrue<-Ymat[OOB,ms]

### The OOB classification ###
obslistOOB<-Dropobsgeneral(Xmat[OOB,],modelsused[[j]],m,perms[j,],
labelthr,splityesno[j,])
```

```
OOBpredictions<-Makeapred(obslistOOB,nodeprobs[j,],K,length(OOB),
perms[j,],m)

OOBclassifications<-OOBpredictions[,ms]
OOBclassifications<-ifelse(OOBclassifications>=0.5,1,0)

allmeasures<-Pmeasures(Ytrue,OOBclassifications)
OOBHL[j]<-allmeasures[[1]]                   #calculate Hamming loss
OOBFscore[j]<-allmeasures[[5]]               #calculate F-score

### The OOB classifications over all permuted variables ###
thearray<-permimportance(Xmat,modelsused,nodeprobs,perms,splityesno,
M,K,m,labelthr,OOBlist,j)

for(p in 1:P){
allmeasures<-Pmeasures(Ytrue,thearray[,,p])
permutedresultsHL[p,j]<-allmeasures[[1]]      #calculate Hamming loss
permutedresultsFscore[p,j]<-allmeasures[[5]]} #calculate F-score

}  #j

######################################################################
### Finding differences ###
for(p in 1:P){
GlobaldiffHL[p,]<-(as.vector(permutedresultsHL[p,]))-OOBHL
GlobaldiffFscore[p,]<-OOBFscore-(as.vector(permutedresultsFscore[p,]))}

### Computing mean of differences ###
GlobalimportanceHL<-apply(GlobaldiffHL,1,mean)
GlobalimportanceFscore<-apply(GlobaldiffFscore,1,mean)

return(list(GlobalimportanceHL,GlobalimportanceFscore))
}
```

calculateLOCALimportance:

```
calculateLOCALimportance<-function(Xmat,Ymat,modelsused,nodeprobs,perms,
splityesno,M,K,m,labelthr,OOBlist)
{
#########################################################################
# Function used to calculate local LDsplit MDA variable importance
# (as outlined in Section 5.5.1)
# Inputs:
#   Xmat = original matrix of input values
#   Ymat = original matrix of 0-1 label vectors
#   modelsused = list of trained models used to split observations
#   nodeprobs = matrix where each row is node probabilities of
#               a label order
#   perms = matrix where each row gives the label order used
#   splityesno = matrix where each row gives code if node was split or
#                not split (1 = split, -1 = no split)
#   M = number of label orders (size of ensemble)
#   K = total number of labels
#   m = total number of levels
#   labelthr = vector of size K containing threshold to apply
#              for each label
#   OOBlist = list where each element is the OOB-set for each tree
# Output:
#  averages = pxK matrix of local importance of variables
#########################################################################

#label identifier matrix
labelidentify<-matrix(0,nrow=K,ncol=M)
for(j in 1:M){
labelidentify[perms[j,],j]<-1}
howmany<-apply(labelidentify,1,sum)

P<-ncol(Xmat)


#Local calculation
storepropcorrect<-matrix(-1,nrow=M,ncol=m)
storepropcorrectp<-array(-1,dim=c(P,m,M))

########################################
#Per tree
for(j in 1:M){

### True OOB classification of tree ###
OOB<-OOBlist[[j]]
ms<-sort(perms[j,])
Ytrue<-Ymat[OOB,ms]
```

```
### The OOB classification ###
obslistOOB<-Dropobsgeneral(Xmat[OOB,],modelsused[[j]],m,perms[j,],
labelthr,splityesno[j,])

OOBpredictions<-Makeapred(obslistOOB,nodeprobs[j,],K,length(OOB),
perms[j,],m)

OOBclassifications<-OOBpredictions[,ms]
OOBclassifications<-ifelse(OOBclassifications>=0.5,1,0)

### Calculating proportion of correct classifications per label ###
wrong<-abs(OOBclassifications-Ytrue)
propwrong<-(apply(wrong,2,sum))/(length(OOB))
propcorrect<-(rep(1,m))-propwrong
storepropcorrect[j,]<-propcorrect

### The OOB classifications over all permuted variables ###
thearray<-permimportance(Xmat,modelsused,nodeprobs,perms,splityesno,
M,K,m,labelthr,OOBlist,j)

for(p in 1:P){
wrongp<-abs(thearray[,,p]-Ytrue)
propwrongp<-(apply(wrongp,2,sum))/(length(OOB))
propcorrectp<-(rep(1,m))-propwrongp
storepropcorrectp[p,,j]<-propcorrectp}
}  #j

### Finding differences ###
alldifferences<-matrix(0,nrow=p,ncol=(m*M))
for(j in 1:M){
for(p in 1:P){
eindig<-(j*m)
begin<-(eindig-(m-1))
thedifference<-(storepropcorrect[j,])-(storepropcorrectp[p,,j])
alldifferences[p,begin:eindig]<-thedifference}}
### Calculating mean of differences ###
columncodes<-rep(0,m*M)
for(j in 1:M){
thelabels<-sort(perms[j,])
eindig<-(j*m)
begin<-(eindig-(m-1))
columncodes[begin:eindig]<-thelabels}

averages<-matrix(0,nrow=p,ncol=K)
for(p in 1:P){
for(k in 1:K){
thecolumns<-which(columncodes==k)
averages[p,k]<-mean(alldifferences[p,thecolumns])}}

return(averages)
}
```