L-classifier chains classification and variable selection for multi-label datasets

Monika du Toit

Thesis presented in partial fulfillment of the requirements for the degree of MCom (Mathematical Statistics) at the University of Stellenbosch

December 2016

Supervisor: Professor SJ Steel

Stellenbosch University https://scholar.sun.ac.za

Plagiarism declaration

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.

2. I agree that plagiarism is a punishable offence because it constitutes theft.

3. I also understand that direct translations are plagiarism.

4. Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5. I declare that the work contained in this assignment, except otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

	Signature
Initials and surname M du Toit	**Date December 2016**

Copyright © 2016 Stellenbosch University
All rights reserved

Stellenbosch University https://scholar.sun.ac.za

Abstract

Multi-label classification extends binary and multi-class classification to scenarios where every data case is assigned several labels simultaneously. Applications include labelling images with tags, identifying instruments that are playing in a musical piece and classifying text according to two or more labels. Variable selection is an important part of multi-label data analysis, but it has received little attention in the literature. Multi-label variable selection is more complex than for binary classification, mainly due to the presence of more than one response as well as label dependence.

In this thesis, a multi-label classification approach called L-classifier chains (LCC) is proposed. This method implements a compromise between simple classifier chains and the ensemble of classifier chains procedures. The LCC approach uses an ensemble of classifier chains with a semi-random chain structure and random forests as base learners to perform variable selection. The specific structural assumptions of the LCC method allow for variable importance inference based on the output from the random forests. The results from LCC include multi-label predictions and a matrix of variable importance values.

This thesis illustrates the application of the LCC clasifier by conducting empirical work using multi-label benchmark datasets, simulated datasets and a practical dataset obtained from a South African credit bureau. Throughout the practical applications, it compares the performance of LCC relative to three other classifiers, namely binary relevance, classifier chains and ensemble of classifier chains.

Key words:

classification, multi-label, random forests, variable importance

Stellenbosch University https://scholar.sun.ac.za

Table of contents

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of figures

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of tables

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of appendices

Stellenbosch University https://scholar.sun.ac.za

List of abbreviations and/or acronyms

AA	Algorithm adaptation
AC	Accuracy
BR	Binary relevance
CC	Classifier chains
CL	Classification accuracy
ECC	Ensemble classifier chains
F2	F2 score
HL	Hamming loss
LCC	L-classifier chains
LP	Label powerset
ML	Multi-label
PPT	Pruned problem transformation
PR	Precision
PT	Problem transformation
RE	Recall
RF	Random forests
VS	Variable selection

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 1: Introduction

1.1 BACKGROUND

Supervised statistical learning refers to a set of approaches for estimating a functional relationship $f(\boldsymbol{x})$ between a set of input variables $\boldsymbol{X}$ and an output Y, in order to find a prediction rule for new input values $\boldsymbol{x}_0$:

$$f: \boldsymbol{X} \to Y(\boldsymbol{X}), \quad \hat{y} = \hat{f}(\boldsymbol{x}_0).$$

For quantitative response variables, regression analysis techniques are applied. Alternatively, qualitative responses require the use of classification analysis. In regression, the functional relationship $f(\boldsymbol{x}) = E(Y|\boldsymbol{X} = \boldsymbol{x})$ is the expected value of the conditional distribution. Since the value of $f(\boldsymbol{x})$ is typically unknown, $E(Y|\boldsymbol{X} = \boldsymbol{x})$ needs to be estimated. In classification with two classes, it is of interest to compute the predictions $\hat{G}(\boldsymbol{x}) = argmax_g \hat{f}(g|\boldsymbol{x}), g \in \{0,1\}$. The functional relationship in this case determines the threshold value for prediction rules. Depending on the model, it may refer to the estimated Bayes posterior probability or the majority class in a given cluster, for example. Classification will be the main focus of this thesis.

There exists a trade-off between the prediction accuracy and interpretability of a model. In classification, less flexible approaches such as logistic regression are restricted in terms of estimating $f(\boldsymbol{x})$. These methods tend to have lower variance but potentially higher bias than more complex models. Interpretability of such models is often good. More flexible approaches, such as classification trees, model a wider variety of shapes of $f(\boldsymbol{x})$. They often have lower bias, potentially higher variance and can provide limited inference. Deciding on the form of $f(\boldsymbol{x})$ becomes more difficult as the dimensionality and complexity of datasets increase.

The objective of this thesis is to consider a specific type of classification, multi-label (ML) classification, where each observation is associated simultaneously with more than one binary label. The thesis will focus on introducing an ML classification

Stellenbosch University https://scholar.sun.ac.za

method, as well as exploring variable selection (VS) and variable importance within this context.

1.2 NOTATION

The general notation in the ML context is focused on two sets of variables. Consider an ML training dataset of size *N*, consisting of *P* predictors and *L* binary labels. Let a set of responses or labels in this dataset be denoted by $\boldsymbol{Y} = (Y_1, Y_2, \dots, Y_L)$. Similarly, a set of input features, which refer to the variables conveying information about the labels, is $\boldsymbol{X} = (X_1, X_2, \dots, X_P)$. A training ML dataset is denoted by $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{N}$, where $\boldsymbol{x}_i: 1\times P$ is a feature vector of *P* predictors and $\boldsymbol{y}_i: 1\times L$ is a response vector of *L* binary labels, corresponding to observation i. The matrix representation of the ML dataset is $\boldsymbol{X}: N\times P$ and $\boldsymbol{Y}: N\times L$. The columns of an observed data matrix $\boldsymbol{X}$ and $\boldsymbol{Y}$ are denoted by *N*-dimensional vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_P$ and $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_L$ respectively.

1.3 OVERVIEW

This thesis is divided into eight chapters, out of which the first and the last chapters consist of the introductory and conclusion chapters. The body of the thesis is split into theoretical Chapters 2, 3 and 4, and Chapters 5, 6 and 7, which describe the empirical work done.

Chapter 2 describes ML classification in general, starting with the distinction between the main classification types. It zooms in on ML classification and provides a description of some of the key characteristics of this problem. Furthermore, the objective of this thesis is also provided in this chapter. Finally, some of the most significant ML classification approaches are summarised and described.

The focus of Chapter 3 is on one of the ML classification methods known as classifier chains. Various aspects of this classifier are discussed, including its modifications and limitations, supported with some experimental work. In this chapter, the LCC algorithm is proposed and its most important aspects are explained in detail. Lastly, the implementation of VS within the LCC classifier is described.

Stellenbosch University https://scholar.sun.ac.za

Variable selection is the primary focus of Chapter 4. In this chapter, some of the problems associated with high-dimensional problems are discussed, followed by ways of overcoming them. A brief overview of existing VS methods is provided, followed by a summary of VS methods in ML classification. Special attention is paid to embedded VS methods, namely decision trees, bagging and random forests (RF). Furthermore, the concept of variable importance is explored within the RF model. Lastly, RF application in the proposed LCC classifier is described.

In Chapter 5, the benchmark datasets analysis is presented. It is the first part of the experimental work done in this thesis. In this analysis, three ML benchmark datasets are analysed using four ML classifiers, including the LCC classifier. The main idea of using these datasets is to apply some of the ML techniques to datasets available online. The objective of this analysis is to compare the accuracy of predictions of these classifiers. The comparison is done using various exploratory and confirmatory analyses of the results. Furthermore, the variable importance values from the LCC classifier are analysed.

The second part of the experimental work is summarised in Chapter 6. In this chapter, a simulation study was done in order to compare the ML classifiers in a controlled environment. This study consists of 32 simulation designs, each of which has varying values of the five parameters of interest. Exploratory, as well as confirmatory analyses of the ML classification results are provided. Lastly, the variable importance values from the LCC classifier's output are explored within the different simulation designs.

In Chapter 7, the final part of the experimental work is presented. In this chapter a practical ML classification dataset obtained from a South African credit bureau is analysed. Of interest is to compare the performance of the four classifiers in an applied setting. Lastly, the importances of the predictors are analysed using variable importance values obtained from the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 2: Multi-label classification

Classification in the context of multi-label data problems has stirred up interest in the last decade. Compared to simple binary classification, it offers larger scope for analysis due to the complexity of ML datasets. The concept of conditional label dependence is one of the main factors underlying the dataset structure. Applications of ML classification include the areas of image and text annotation, bioacoustics and bioinformatics.

2.1 CLASSIFICATION HIERARCHY

Classification is a statistical method supervised by a qualitative output. Its categories include binary, multi-class, multi-label and multi-output classification. The main distinction between these categories is in the format of the output.

As the name suggests, binary classification is associated with a single two-class response variable. A binary dataset is of the form:

$$(\boldsymbol{x}_i,\ y_i)_{i=1}^{N}, \qquad \boldsymbol{x}_i \in \mathbb{R}^P, \qquad y_i \in \{0,1\}.$$

Extensive research has been done in this context. Generally, a method tries to identify a classification rule and determine which of the two classes is more likely to be associated with a given input. This decision rule can be linear (for example, linear discriminant analysis), non-linear (for example, RF) or linear in an extended feature space (for example, support vector machines). Many of these classifiers estimate the posterior probabilities and classify observations to the class with the highest estimated posterior probability:

$$\hat{P}(Y_j|\boldsymbol{x}) = \frac{\hat{f}(\boldsymbol{x}, Y_j)}{\hat{f}(\boldsymbol{x})}, \qquad j \in \{0,1\}, \qquad \hat{Y} = argmax_{j\in\{0,1\}}[\hat{P}(Y_j|\boldsymbol{x})].$$

Fraud detection, medical testing and quality control are examples of binary classification.

Stellenbosch University https://scholar.sun.ac.za

In multi-class classification every data case belongs to one of K mutually exclusive classes. Examples of multi-class datasets include vowel recognition and multiple-type cancer classification. A multi-class dataset can be represented in the following way:

$$(\boldsymbol{x}_i,\ y_i)_{i=1}^{N}, \qquad \boldsymbol{x}_i \in \mathbb{R}^P, \qquad y_i \in \{1,2,\ldots,K\},\ \ K \geq 3.$$

Multi-label classification can be identified by a set of L binary class labels that are non-mutually exclusive. Therefore, each observation can be associated with more than one label. Some of the well known applications include image annotation, text categorisation and music instrument recognition. An ML dataset has the following notation:

$$(\boldsymbol{x}_i,\ \boldsymbol{y}_i)_{i=1}^{N}, \qquad \boldsymbol{x}_i \in \mathbb{R}^P, \qquad \boldsymbol{y}_i \in \{0,1\}^L,\ \ L \geq 2.$$

Lastly, a combination of L labels each consisting of K label classes is referred to as multi-class multi-label or multi-output classification. It is the most complex type of classification dataset. Examples of such a problem include Wikipedia page label tags and Flickr photo label tags (Dekel, 2010). The classification hierarchy is summarized in Table 2.1.

Table 2.1: Summary of classification types

		Number of labels per data case	
		$L = 1$	$L \geq 2$
Number of label classes	$K = 2$	Binary	Multi-label
	$K \geq 3$	Multi-class	Multi-class multi-label

Stellenbosch University https://scholar.sun.ac.za

2.2 COMPLEXITY OF MULTI-LABEL DATASETS

In the last decade research on ML classification has experienced significant growth. However, compared to well-researched binary classification, ML classification is still more of an unchartered type of problem. Some approaches are better known than others, but in general a standard procedure for analysing ML datasets has not been established. The growing research as well as the lack of set methods can be attributed to the complexity of ML datasets. Problems involving ML datasets require more extensive learning than the binary case. The general structure of an ML dataset is provided in Table 2.2.

The complexity of ML classification can be described using the music instrument classification example. Each piece of music (observation) is characterised by sound frequencies (features) and the instruments playing (labels). It is an ML dataset, since each piece of music can be associated with more than one label, for example violin, piano and trumpets.

Table 2.2: Multi-label dataset structure.

	Predictors				Labels			
Data instances	$\boldsymbol{X}_1$	$\boldsymbol{X}_2$	…	$\boldsymbol{X}_P$	$\boldsymbol{Y}_1$	$\boldsymbol{Y}_2$	…	$\boldsymbol{Y}_L$
1	x_{11}	x_{12}	…	x_{1P}	1	0	…	1
2	x_{21}	x_{22}	…	x_{2P}	0	0	…	1
…	…	…	…	…	…	…	…	…
N	x_{N1}	x_{N2}	…	x_{NP}	0	1	…	1

Working with an ML dataset requires consideration of various aspects. Firstly, there may be dependencies between features, resulting in multi-collinearity with its implied disadvantages. Secondly, there can exist dependencies between labels. For example, violin and piano are more likely to play simultaneously than trumpet and harp. Exploiting such dependencies is a major challenge in ML research. Furthermore, as in any classification problem, the relationship between the sound frequencies and the instruments playing, the input and output variables, needs to be modelled. This

Stellenbosch University https://scholar.sun.ac.za

relationship can become highly complex, especially in high dimensions and/or with many labels. Lastly, the high-dimensional input data often leads to additional challenges, should the predictors include irrelevant and/or redundant variables.

Given these complexities, determining a good ML classifier is likely to be more challenging than in the binary case. Furthermore, evaluating the goodness of a classifier is more complex in the ML case. This is due to the various ML evaluation measures that have been proposed, which tend to favour different methods for the same dataset.

2.3 OBJECTIVES WHEN ANALYSING MULTI-LABEL DATASETS

In an ML setting, the joint conditional distribution of $\boldsymbol{Y}$ given $\boldsymbol{X} = \boldsymbol{x}$ is to be maximised in order to find the most probable class labels (Hong, 2014). Therefore, a multivariate posterior probability needs to be maximised:

$$f(\boldsymbol{x}) = argmax_{\boldsymbol{y} \in \{0,1\}^L}[P(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{X} = \boldsymbol{x})].$$

Considering an ML classification problem in a probabilistic setting, it can be seen that estimating such a function accurately is not straightforward. This task requires taking into account the joint distributions of both the input and output variables, while at the same time approximating a conditional relationship between the two sets of variables.

The statistical analysis of an ML dataset includes the following objectives. Firstly, it aims at accurately classifying previously unseen observations $\boldsymbol{x}_0$. This is known as ML classification. Secondly, the objective of ML ranking involves assigning an ordered sequence of labels to $\boldsymbol{x}_0$, from most to least relevant labels (Tsoumakas *et al.*, 2010).

Apart from these two main objectives of ML analysis, there are often other objectives of interest. Inference involving interpretation of the relationship between predictors and labels is often important. Moreover, obtaining a lower-dimensional representation of the ML dataset can be useful in certain applications. Finally, in a high-dimensional setting, VS or shrinkage can significantly improve ML classification and inference. The objective of this thesis is to introduce an ML

Stellenbosch University https://scholar.sun.ac.za

classification method that takes label dependencies into consideration. Variable selection when this method is applied is also investigated. Furthermore, quantifying variable importance in this setting is integrated into the classifier.

2.4 LABEL DEPENDENCE

Label dependence is a central theme in ML learning. The complexity of the label dependencies potentially grows with the number of labels. The rich nature of ML datasets has led to many research investigations of label dependence. It is essential that model complexity as well as computational practicality of a classifier need to be considered simultaneously, especially in high-dimensional problems.

It is fair to question the importance of taking label dependence into account in general. If label dependencies exist, should a classifier take them into account when estimating $P(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \boldsymbol{x})$? Assuming that the conditional label dependence could be accurately estimated, does this information lead to better classifiers or does it only add complexity and noise? The answer to this question depends on whether the label dependence is accurately estimated. Inaccurate estimation could likely worsen the accuracy of classifiers while capturing the label dependence well could improve the fit.

Determining whether label dependencies are present in a given dataset is difficult. The following definitions are aimed at understanding the concepts better. If the joint distribution of labels is not a product of their marginal distributions, there exists a so-called unconditional dependence, given as (Read, 2013):

$$P(Y_j, Y_k) \neq P(Y_j) \times P(Y_k).$$

Conditional dependence of labels given the inputs $\boldsymbol{x}$ exists if the following result holds:

$$P(Y_j, Y_k|\boldsymbol{X} = \boldsymbol{x}) \neq P(Y_j|\boldsymbol{x}) \times P(Y_k|\boldsymbol{x}).$$

Unconditional label dependence can for example be quantified by using the observed label frequencies to estimate the mutual information (Read, 2013). Measuring conditional label dependence is however more challenging, due to the conditioning

Stellenbosch University https://scholar.sun.ac.za

on possibly a large number of predictors. In an ML setting, how can the conditional label dependence structure be accurately estimated? A direct method for estimating the conditional label dependence is a difficult topic, which will not be attended to in this thesis. However, classifiers that use the information from label dependence are described in the following section.

Multi-label classifiers that use the label dependence information for data classification can be characterised to be of first-, second- or higher-order (Zhang & Zhang, 2010). These terms correspond to the degree to which models take label dependencies into account. First-order classification approaches consider labels independently of each other. They do not take other labels into account when predicting a label. Second-order approaches make use of pairs of labels and consider the relationship between two labels at a time. Higher-order approaches consider more than two labels at a time in ML learning. For example, classifiers can use the interaction between a label and all other labels. Examples of these three approaches are provided in Section 2.6.

As with any statistical learning problem, the task of approximating the true underlying distribution is associated with uncertainty and a single best method for ML classification does not exist. Methods that model label dependence to a large extent tend to be computationally complex and often inefficient for large datasets. On the other hand, simpler methods are faster but can possibly ignore some information from the data.

These factors make the ML classification task challenging. The final word on the best way to approach ML datasets has thus not been spoken. In general, in order to decide whether a specific ML classifier is better than another classifier, the performances of the classifiers need to be evaluated. The following section summarises some of the evaluation measures for ML classification.

Stellenbosch University https://scholar.sun.ac.za

2.5 MULTI-LABEL EVALUATION MEASURES

Unlike in binary classification, the presence of multiple labels makes the evaluation of ML classifiers more complex. In binary classification, it is easy to determine which of the predicted labels were classified correctly, and which were misclassified. Taking the same approach in ML classification would involve evaluating the labels independently. It may however be useful to treat the labels dependently in some way, since the labels belong to a specific data case. This would allow for quantifying how successful the classifier was at predicting the joint set of labels.

Some of the most commonly reported evaluation measures for ML classification are described below (Tsoumakas *et al.*, 2010:13). These measures consider an ML test dataset of N_{test} observations, $(\boldsymbol{x}_i, \boldsymbol{Y}_i)_{i=1}^{N_{test}}$. Let $\hat{f}(\boldsymbol{x}_i) = \widehat{\boldsymbol{Y}}_i$ be the predicted set of labels based on an ML classifier.

Hamming loss (HL) measures the average proportion of misclassified labels per observation. It resembles the misclassification error of binary classification, since it considers every label separately, and it is given by:

$$HL = \frac{1}{LN_{test}} \sum_{i=1}^{N_{test}} \sum_{j=1}^{L} I\left(Y_{ij} \neq \hat{Y}_{ij}\right).$$

Since HL considers scalar values and not vectors of label combinations, it does not evaluate how well a classifier captured the multi-labelled nature of the problem. To illustrate this concept, consider classifier A that resulted in a lower value of HL compared to classifier B. While on average classifier A predicted individual labels more accurately, every case contained one misclassified label. On the other hand, classifier B predicted on average fewer correct labels according to HL, but only a few cases were completely misclassified. Therefore, the majority of cases were predicted accurately for every label. In this example, classifier A successfully predicted more individual labels accurately, while classifier B correctly predicted the set of labels for the majority of observations.

In contrast to HL, classification accuracy (CL) measures the average exact label match per observation. While it provides a useful way of evaluating ML classifiers, it

Stellenbosch University https://scholar.sun.ac.za

leaves no room for errors, not even for a single label. Considering that some large ML datasets may contain hundreds of labels, CL is a very strict evaluation measure. However, should exact matching be of importance, this measure can be useful. The CL can be computed in the following way:

$$CL = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I(\boldsymbol{Y}_i = \widehat{\boldsymbol{Y}}_i).$$

Besides HL and CL, other measures were introduced to evaluate ML classifiers. Precision (PR) and recall (RE) measure the proportion of correctly positively predicted labels per case, out of all positively predicted labels (PR), or out of all true positive labels (RE):

$$PR = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{|\boldsymbol{Y}_i \cap \widehat{\boldsymbol{Y}}_i|}{|\widehat{\boldsymbol{Y}}_i|}, \qquad RE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{|\boldsymbol{Y}_i \cap \widehat{\boldsymbol{Y}}_i|}{|\boldsymbol{Y}_i|},$$

where $|A|$ denotes the number of elements in the set A that equal to 1. While PR estimates how likely the predicted labels are truly present, the RE estimates how likely true labels are actually detected by the classifier. Since there is a trade-off between PR and RE, they should be considered simultaneously. The F2 measure uses their harmonic mean to capture the combined information from PR and RE, and it is defined as:

$$F2 = \frac{1}{\frac{1}{2}\left(\frac{1}{PR} + \frac{1}{RE}\right)}.$$

Lastly, the accuracy (AC) measure computes the mean proportion of correctly positively classified labels out of the union of predicted and true labels present per case. The AC measure thus considers both the true and predicted label sets simultaneously. It is another useful measure for evaluating ML classifiers, and it is defined in the following way:

$$AC = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{|\boldsymbol{Y}_i \cap \widehat{\boldsymbol{Y}}_i|}{|\boldsymbol{Y}_i \cup \widehat{\boldsymbol{Y}}_i|}.$$

Stellenbosch University https://scholar.sun.ac.za

To illustrate the abovementioned measures, consider the vectors of true and predicted labels based on using three different classifiers, as given in Table 2.3. All measures except for HL are to be maximised, and the bold values correspond to the best classifier for a given measure.

Table 2.3: Illustration of six ML evaluation measures for three scenarios.

Classifier	$\boldsymbol{y_i}$	$\boldsymbol{\hat{y}_i}$	**HL**	**CL**	**PR**	**RE**	**F2**	**AC**
1	{1, 1, 0, 1, 0}	{1, 0, 0, 0, 0}	0.4	**1**	**1**	0.33	0.5	0.33
2	{1, 1, 0, 1, 0}	{1, 1, 1, 1, 1}	0.4	**1**	0.6	**1**	0.75	0.6
3	{1, 1, 0, 1, 0}	{1, 0, 0, 1, 0}	**0.2**	**1**	**1**	0.67	**0.86**	**0.67**

Due to the presence of multiple measures, determining whether a classifier is better than others depends on which evaluation measure is considered. In Table 2.3, the inverse relationship between PR and RE can be observed by comparing classifiers 1 and 2. Classifier 1 resulted in the highest PR but lowest RE. Out of the two classifiers, classifier 2 had the higher values of RE, F2 and AC, and therefore could be considered better than classifier 1. Comparing these results with the measures for classifier 3, it is clear that the third classifier has the best values for the HL, PR, F2 and AC measures. Therefore classifier 3 performed overall the best, based on this simple example.

There is some room for interpretation when it comes to ML classifier evaluation, depending on the problem. For example, if the exact match of labels is of importance for a specific application, CL could be given a larger weight. For the purposes of this thesis, a combination of equally weighted measures was considered when comparing classifiers in experimental work. The following section describes some of the specific approaches to ML classification in more detail.

Stellenbosch University https://scholar.sun.ac.za

2.6 DIFFERENT APPROACHES TO MULTI-LABEL CLASSIFICATION

Classification in an ML context involves predicting more than one label for each observation. There are three main approaches to ML classification, namely the algorithm adaptation, problem transformation and ensemble approaches.

Algorithm adaptation (AA) approaches take a binary classification algorithm and adapt it for ML data to directly output multiple label predictions. Examples of this approach include the ML-kNN approach by Zhang and Zhou (2007), application of decision trees in the ML-C4.5 algorithm by Clare and King (2001), as well as AdaBoost.MH and AdaBoost.MR boosting algorithms by Schapire and Singer (2000). These approaches are algorithm-specific and can result in effective models. However, for some classifiers adapting the algorithm can be a complex task.

Problem transformation (PT) approaches involve adapting the data to the algorithm. These approaches transform the ML dataset into one or more binary datasets. The PT approaches use binary classification methods and transform predictions back to ML representations. One of the main advantages of this approach lies in using existing binary classification algorithms. Methods of transforming ML datasets include binary relevance, label powerset and pairwise methods (Madjarov *et al.*, 2012). The PT approaches will be the primary focus of this thesis.

Ensemble approaches are extensions of the AA and PT methods. They usually consist of repeating a specific method numerous times in order to improve it. Examples include random k-labelsets (RAkEL) by Tsoumakas and Vlahavas (2007), ensembles of classifier chains by Read *et al.* (2011), and RF of predictive clustering trees (RF-PCT) by Kocev *et al.* (2007). These methods are computationally intensive and require further tuning of parameters, but can provide accurate models.

Apart from the traditional approaches to ML learning, research done in various fields contributed with alternative approaches to ML classification. For example, ML output coding uses the system of encoding labels into a lower-dimensional space, predicting encoded labels in this space and then decoding predictions back to the original label space. This scaling allows for simpler predictions and graphical

Stellenbosch University https://scholar.sun.ac.za

representations. ML output coding methods include principal label space transformations (Tai & Lin, 2012) and output coding with canonical correlation analysis (Zhang & Schneider, 2011). The following figure summarises some of the most significant approaches to ML classification. The PT approaches are described in more detail in the following section.

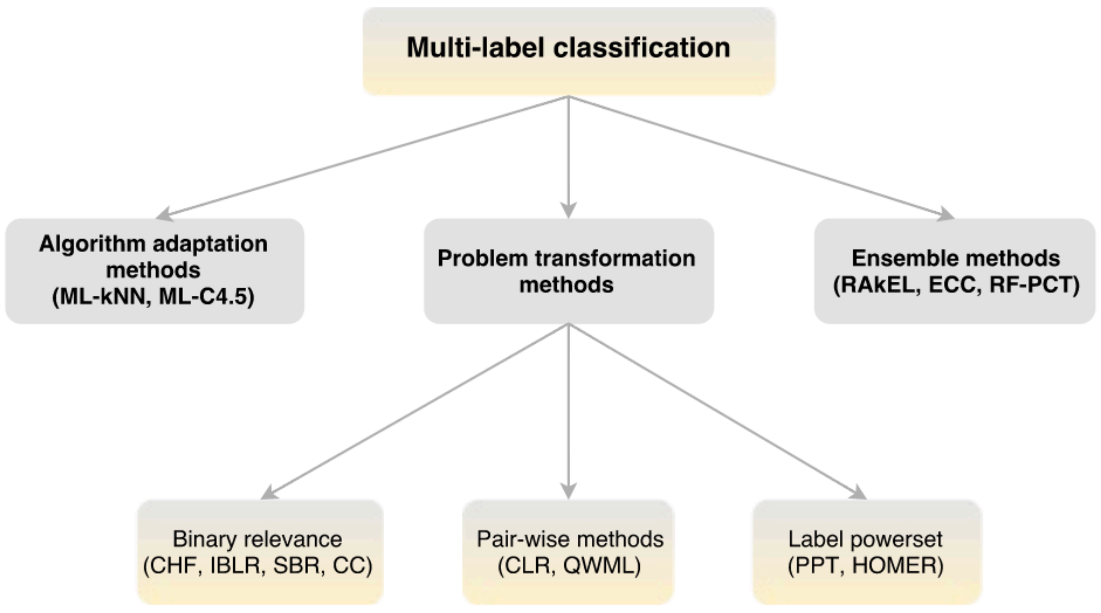

Multi-label classification
Algorithm adaptation methods (ML-kNN, ML-C4.5)
Problem transformation methods
Ensemble methods (RAkEL, ECC, RF-PCT)
Binary relevance (CHF, IBLR, SBR, CC)
Pair-wise methods (CLR, QWML)
Label powerset (PPT, HOMER)

Figure 2.1: Summary of multi-label classification methods.

2.7 PROBEM TRANSFORMATION METHODS

2.7.1 BINARY RELEVANCE

One of the best-known problem transformation approaches is known as binary relevance (BR). This method simply transforms the ML problem into L independent single-label binary problems. A binary classifier is then constructed for each of the L problems. These classifiers are responsible for predicting labels for the corresponding response variables. Table 2.4 summarises this approach.

Often the assumption of label independence is somewhat unrealistic. Consider for example music frequencies that can be associated with correlated music instruments. Ignoring this information may lead to inaccurate predictions of instruments that usually do not play together in practice, such as trumpet and harp. The BR classifier

Stellenbosch University https://scholar.sun.ac.za

focuses on marginal label distributions instead of joint distributions, thus ignoring label dependence altogether. On the other side, the strong independence assumption makes BR computationally efficient, simple and intuitive. Moreover, since BR treats the labels independently, it is not affected by a change in label dependence structure.

Table 2.4: Binary relevance method.

	$\hat{f}_1$		$\hat{f}_2$			$\hat{f}_L$	
Data instances	$\boldsymbol{X}$	$\boldsymbol{Y}_1$	$\boldsymbol{X}$	$\boldsymbol{Y}_2$	…	$\boldsymbol{X}$	$\boldsymbol{Y}_L$
1	$\boldsymbol{x}_1$	1	$\boldsymbol{x}_1$	0	…	$\boldsymbol{x}_1$	1
2	$\boldsymbol{x}_2$	0	$\boldsymbol{x}_2$	0	…	$\boldsymbol{x}_2$	1
…	…	…	…	…	…	…	…
N	$\boldsymbol{x}_N$	0	$\boldsymbol{x}_N$	1	…	$\boldsymbol{x}_N$	1

$$\hat{f}_l: \boldsymbol{x} \rightarrow y_l, \qquad y_l \in \{0,1\}, \qquad l = 1, \dots, L.$$

The BR approach is referred to in almost every piece of literature on ML classification, and many methods use it as the foundation of their proposal. Examples include classification with heterogeneous features (Godbole & Sarawagi, 2004), instance-based logistic regression (Cheng & Hüllermeier, 2009), and classifier chains Read *et al*. (2011). In its original form, the BR method considers label dependency of first-order. The BR classifier ignores the label correlations, considering every label independently without regard for other labels.

2.7.2 LABEL POWERSET

In terms of modelling label dependence, label powerset (LP) (Tsoumakas & Vlahavas, 2007) lies on the opposite side compared to BR. Instead of splitting the label set according to the L separate labels, it joins all labels corresponding to a single case into one label class. Observations with the same label combination are labelled with the same label class. The new single label classes are therefore no longer binary but multi-class. The ML classification problem is thus transformed into a multi-class problem. Following the transformation, classification is performed using appropriate multi-class classifiers. The LP transformation is represented in the following table:

Stellenbosch University https://scholar.sun.ac.za

Table 2.5: Label powerset method.

	Predictors				Labels
Data instances	$\boldsymbol{X}_1$	$\boldsymbol{X}_2$	…	$\boldsymbol{X}_P$	$\boldsymbol{Y}$
1	x_{11}	x_{12}	…	x_{1P}	1
2	x_{21}	x_{22}	…	x_{2P}	1
…	…	…	…	…	…
N	x_{N1}	x_{N2}	…	x_{NP}	3

$$\hat{f}\colon \boldsymbol{x} \rightarrow y, \qquad y \in \{1,2,\dots,K\}, \qquad K \geq 2.$$

This intuitive and easy transformation leads to a few desirable properties. Whereas the BR approach ignores the label dependence completely, the LP approach considers the full conditional joint distribution of the labels. This is as a result of the label structure left untouched during the transformation phase.

The LP approach also has some disadvantages. Maintaining the structure comes at a price of having a large number of label classes as the number of label combinations increases. A large number of classes make multi-class classifiers computationally inefficient. Furthermore, there is typically an imbalanced distribution of label classes. Label combinations with low frequencies contribute to this imbalance. The predictions based on an imbalanced distribution are more challenging, as there are few data points for the infrequent label combinations to use. Lastly, any label combination that is not modelled at the training phase will not be considered for prediction, resulting in model bias. These limitations of LP can make multi-class classification very difficult.

Pruned problem transformation (PPT) (Read, 2008) is an extension of the LP method that initially transforms the ML dataset in the same way as LP. Most of the issues identified for the LP method are caused by having a large number of class labels. PPT aims at reducing the number of classes by eliminating the most infrequent ones. Furthermore, PPT provides simpler models than LP and focuses on the most important label combinations. It is a computationally more efficient process, potentially without class imbalance issues. However, bias increases as fewer label

Stellenbosch University https://scholar.sun.ac.za

combinations are considered for prediction. On the other hand, PPT requires specifying a threshold that determines which label combinations are considered infrequent. It is also important to note that by removing the infrequent observations the training dataset becomes smaller, with less information entering the model. Despite these issues, a PPT model can be useful for situations where only the most frequent label combinations are of interest.

Other modifications of the LP method include an ensemble approach, RAkEL, and hierarchy of ML classifiers (HOMER) (Tsoumakas *et al.*, 2008), where labels are first clustered and a classifier is constructed for each cluster of labels. In general, LP approaches consider label dependency of higher-order. Specifically, the transformation implemented in LP approaches allows for consideration of the influences between all labels.

2.7.3 PAIRWISE METHODS

Pairwise methods are another PT approach to ML classification (Hüllermeier *et al.*, 2008). These methods try to capture a second-order label dependency by considering every label pair in a dataset. Pairwise methods consider the interaction between all pairs of labels. The idea is to transform the dataset of L labels into $\frac{L(L-1)}{2}$ datasets, one for each label pair. Each of these datasets contains a set of binary labels (y_j , y_k, $j \neq k$) with the corresponding observations. The task is thus simplified to using a binary classification technique. The pairwise method is summarised in the following table:

Table 2.6: Pairwise method.

	$\hat{f}_1$		$\hat{f}_2$			$\hat{f}_{\frac{L(L-1)}{2}}$	
Data instances	$\boldsymbol{X}$	$\boldsymbol{Y_1\ vs.Y_2}$	$\boldsymbol{X}$	$\boldsymbol{Y_1\ vs.Y_3}$	**…**	$\boldsymbol{X}$	$\boldsymbol{Y_{L-1}\ vs.Y_L}$
1	$\boldsymbol{x}_1$	1	$\boldsymbol{x}_1$	0	…	$\boldsymbol{x}_1$	1
2	$\boldsymbol{x}_2$	0	$\boldsymbol{x}_2$	0	…	$\boldsymbol{x}_2$	1
…	…	…	…	…	…	…	…
N	$\boldsymbol{x}_N$	0	$\boldsymbol{x}_N$	1	…	$\boldsymbol{x}_N$	1

Stellenbosch University https://scholar.sun.ac.za

At prediction time each of the $\frac{L(L-1)}{2}$ binary classifiers predicts a label for a new case and the predicted label receives a vote. These votes are aggregated over all classifiers and ranked. The final set of predicted labels are the M_{pw} labels with the most votes. The parameter M_{pw} can be specified or determined from the data. Alternatively, a method called calibrated label ranking (Fürnkranz *et al.*, 2008) makes use of an artificially created label that acts as a threshold. Other adaptations of the pairwise method include the QWeighted method proposed by Mencía *et al.* (2010).

Some of the aspects of pairwise learning include computational considerations and label dependency. The number of classifiers that need to be constructed can be significantly more than that of BR, for $\frac{L(L-1)}{2} > L$. However, the number of observations for each classifier is not constant and can be lower than in BR. Only the cases that are labelled with either of the pair of labels considered are included in each of the classifiers. Some of the classifiers will therefore have more data cases to learn from. Lastly, pairwise methods consider only a second-order dependency. The question arises whether it is a sufficient representation of the truelabel relationship.

Some of the most significant methods for ML classification were summarised in this section. Which of these methods performs the best and which should be used in practical applications? It is important to note that just as with binary classification, the best method for ML datasets in general does not exist. However, it could be claimed that RF and boosting are very popular and accurate binary classifiers. Similarly in an ML setting, decision tree-based models are efficient and fast, PT methods that are BR-based are flexible and ensemble methods have strong predictive performance (Read, 2013). According to an extensive study done by Madjarov *et al.* (2012), overall the best methods were RF-PCT, HOMER and then BR and CC.

It is however very important to note that these results, as well as any other results reported in the literature are based on numerous evaluation measures, which tend to favour different approaches. It is therefore not as straightforward to evaluate an ML classifier, as it is to evaluate a binary classifier, as discussed in Section 2.5. The following chapter focuses on a specific PT approach, classifier chains, as well as its limitations and modifications. The proposed ML classifier is also described in this chapter.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 3: Classifier chains in multi-label classification

3.1 CLASSIFIER CHAINS

Classifier chains (CC) were introduced by Read *et al.* (2011). This approach is derived from the BR method. It assumes that conditional label dependence exists and uses the idea of chaining to exploit the label dependence information. The key idea is to capture some of these dependencies, while keeping the process computationally efficient. CC aim at finding a compromise between ignoring the label dependence completely and estimating the complex joint conditional distribution.

In its original formulation, CC construct a single chain of L classifiers, each predicting a corresponding label. The chaining is illustrated in Figure 3.1. In this figure, the first classifier involves predicting Y_1 using only information from $\boldsymbol{X}$. The second label is predicted from both $\boldsymbol{X}$ and Y_1 via the second classifier. This process is repeated L times, resulting in L classifiers. What makes this process different from BR is an expanding feature space for each link in the sequence. At each consecutive chain link, the feature space gets appended with the true binary label from the previous link.

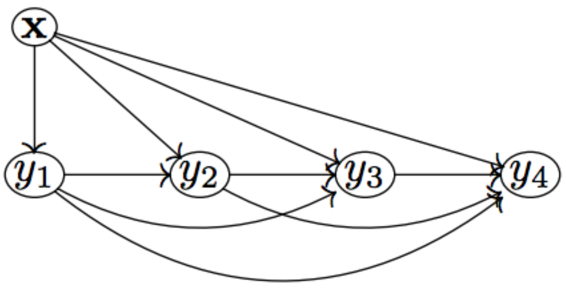

$$\begin{tikzcd}
 & \mathbf{x} \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
y_1 \arrow[r] \arrow[rr, bend right] \arrow[rrr, bend right] & y_2 \arrow[r] \arrow[rr, bend right] & y_3 \arrow[r] & y_4
\end{tikzcd}$$

Figure 3.1: Illustration of the CC training algorithm with four chain links (Read, 2013).

The original CC algorithm is as follows:

For $i = 1, 2, \dots, L$:

1. Fit a base classifier $\hat{f}_i(.)$ using $X_1, X_2, \dots, X_{P+i-1}$ and Y_i as training data.
2. Extend the training data by a new variable X_{P+i}, containing the true labels Y_i.

Stellenbosch University https://scholar.sun.ac.za

Consider a new case with input vector $\boldsymbol{x}_0.: P\times1$. In order to classify new cases, the following steps are required:

1. Compute $\hat{Y}_1(\boldsymbol{x}_0)$ using only the predictors in $\boldsymbol{x}_0$.
2. Append $\hat{Y}_1(\boldsymbol{x}_0)$ to $\boldsymbol{x}_0$ and use this augmented vector to compute $\hat{Y}_2(\boldsymbol{x}_0)$.
3. Repeat Step 2 for every label until all labels are predicted. The last label is predicted using the feature vector [$\boldsymbol{x}_0$, $\hat{Y}_1(\boldsymbol{x}_0)$, ..., $\hat{Y}_{L-1}(\boldsymbol{x}_0)$].

The method of classifier chains enjoys some of the advantages of the BR method, including low memory requirements and computational efficiency. Similar to BR, it requires construction of L classifiers but the feature space is larger along the chain. Moreover, CC makes use of the chaining method and can take label correlations into account. However, it is important to note that it does not manage to model the full joint conditional distribution $P(\boldsymbol{Y}|\boldsymbol{X})$, since it only considers marginal label distributions in a chain-wise manner. The first label in the chain is modelled and predicted without taking any other labels into account, whereas the last label in the chain gets information from all the other labels. Despite this simplification of the underlying relationship, the CC method managed to outperform BR in various studies, including a paper by Read *et al*. (2011).

Interesting questions regarding the original idea of classifier chains arise. Firstly, during the training phase the feature vector is appended with true labels, whereas in the test phase it is expanded with predicted labels. This adds an additional layer of complexity as the classifier relies on accurate predictions of previous labels at the test phase. Should the feature space be expanded with incorrectly classified labels, propagation of error down the chain results. Ideally, such predictors should not be included in the subsequent models. The original CC does not account for a possibility of such error.

Secondly, the conditional label dependence is estimated using the sequence in which the labels enter the chain. In a CC classifier the order is random, starting with the first and ending with the last label. Therefore, the last label in the chain can make use of information from all the other labels. A question arises whether a random chain sequence should be used. It is generally expected that some labels would be more significant in explaining the remaining labels than others. The following experiment

Stellenbosch University https://scholar.sun.ac.za

was done in order to see whether the label sequence order has an effect on the accuracy of the classifier.

The analysis was conducted using the emotions dataset, consisting of six labels and 72 predictors. A more detailed description of the emotions dataset is provided in Section 5.2. The dataset was split into training and a test set using 75% and 25% of the data respectively. There were in total 6! = 720 label permutations of interest. For every label permutation, the CC classifier was fitted to the training dataset, using RF of 200 trees as the base classifier, and label sequence $\boldsymbol{j}_i, i = 1, \dots, 720$. For example, $\boldsymbol{j}_i = (1, 2, 3, 4, 5, 6)$ would indicate that the original CC classifier was implemented. The RF classifier is described in more detail in Section 4.5.3.

After the fitting process, the test set was predicted using the CC model, and a HL measure was computed for each repetition. This measure was chosen randomly, and its purpose was to be an indication of possible differences in performance for the different label sequences. Figure 3.2 shows distribution of HL based on the 720 models using label sequences $\boldsymbol{j}_i, i = 1, \dots, 720$. The R code of this analysis is provided in Appendix A.1.

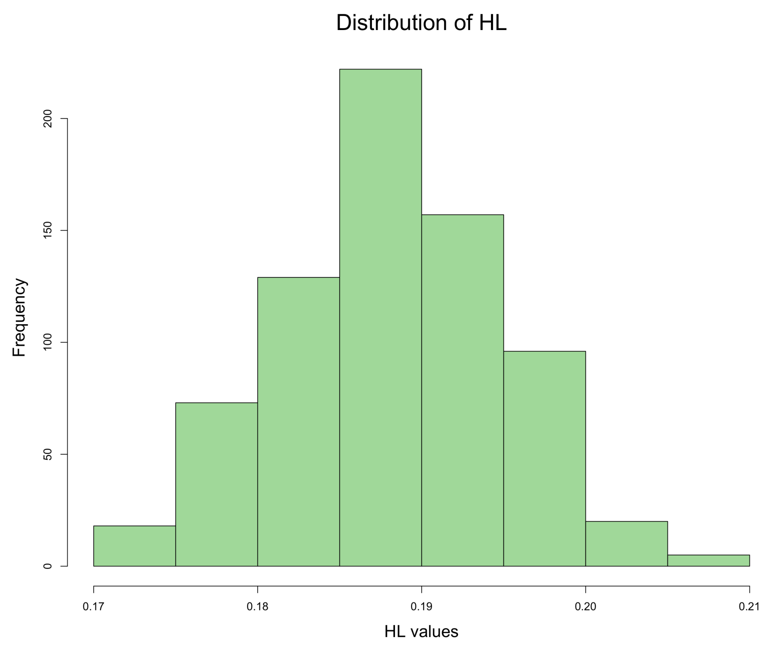


Figure 3.2: CC classification of the emotions dataset using all label permutations.

Stellenbosch University https://scholar.sun.ac.za

The HL distribution is fairly symmetric and slightly skewed to the right. These results are indicative of the effect of the different label sequences. The figure shows that some label orderings performed better than others in terms of HL. However, it is not correct to say that label sequence that provided the lowest value of HL is the best sequence for the emotions dataset in general. Due to data variability, a different training/test split is likely going to provide another 'best' label sequence. Furthermore, HL is just one approach for evaluating ML classifiers. It is always good practice to consider other evaluation measures, in order to obtain a more robust evaluation.

The idea of having a non-random label sequence order has also been explored in Cheng *et al.* (2010). In practice, it is not plausible to fit a classifier to a large dataset using all label permutations in order to choose the best label sequence. However, there have been numerous modifications of CC, which will be discussed in the next section.

3.2 MODIFICATIONS OF CLASSIFIER CHAINS

Since its introduction, CC received a lot of attention in statistical research, including various modifications and implementations. The number of published papers that at least marginally deal with CC is surprising. Most of these methods are inspired by the original chaining idea and address some of its characteristics, shortcomings or propose a new approach altogether. Research is not limited to statistical literature, but includes areas of machine learning, computer science and genetics, with various interesting practical applications. The majority of papers explore the idea of estimating conditional label dependence and selecting the label sequence in the chaining process.

To introduce some of the modifications of the CC algorithm, the following notation is used. Let ξ denote the set of all possible permutations of the integers $1, 2, \dots, L$, and write $\boldsymbol{j}$ for a permutation from this set. The elements of $\boldsymbol{j}$ are denoted by $\{j_1, j_2, \dots, j_L\}$. The original CC procedure implements L binary classifiers using the permutation $\boldsymbol{j} = \{1, 2, \dots, L\}$. The l^{th} of these classifiers is trained on the input data

Stellenbosch University https://scholar.sun.ac.za

$[\boldsymbol{x}, \boldsymbol{y}_1, \dots, \boldsymbol{y}_{l-1}]$, using $\boldsymbol{y}_l$ as the response, for $l = 1, 2, \dots, L$. Here, for $l = 1$, the input data is simply $\boldsymbol{x}$.

3.2.1 ENSEMBLE OF CLASSIFIER CHAINS

The CC authors (Read *et al.*, 2011) proposed an alternative algorithm called ensemble classifier chains (ECC). In ECC, *M* permutations $\boldsymbol{j}_1, \boldsymbol{j}_2, \dots, \boldsymbol{j}_M$ are randomly selected from ξ and ordinary CC are trained using each of these permutations to identify a label sequence. Only a random subset of the training data is used for every classifier. In the empirical work done in Read *et al.* (2011), a subset of 67% of the data was used as the training data for every classifier. Each of the *M* CC classifiers assigns a label vector, $\widehat{\boldsymbol{y}}_m(\boldsymbol{x})$, to a new data case with input vector $\boldsymbol{x}$, $m = 1, \dots, M$. The final assignment of labels to $\boldsymbol{x}$ requires specification of a threshold, say *t*, where $1 \le t \le M$. According to the ECC approach, $\hat{y}_l(\boldsymbol{x}) = 1$ if and only if the l^{th} component of $\sum_{m=1}^{M} \widehat{\boldsymbol{y}}_m(\boldsymbol{x})$ is greater than or equal to *t*. The ECC thus implements a thresholded voting system, where each of the *M* classifiers votes for the labels it predicts.

Although ECC are intuitively sound, perform well (Read *et al.*, 2011) and use a more robust way of working with the chain sequencing, they are computationally expensive. Not only do they require more classifiers to be computed, but values for the number of models *M,* as well as the threshold *t* need to be specified or determined from the data.

3.2.2 1-CLASSIFIER CHAINS

Both CC and ECC use random label sequences and therefore do not explicitly use the existing label dependence structure. This issue gave rise to another way of seeing CC. What if the label sequence could be selected non-randomly to capture the label dependencies more accurately? This sequence could be determined data dependently. Using data to define the best sequence would require a definition of what ‘best’ means: best in terms of one of the ML evaluation measures, or a combination of all of the measures? Approaches to identifying a single label sequence result in classifiers

Stellenbosch University https://scholar.sun.ac.za

that are based on a non-random label sequence. Let us refer to these as '1-CC' classifiers.

In the 1-CC approach a single permutation $\boldsymbol{j}$ from ξ is used to identify the sequence in which label vectors are appended to the input matrix. However, now $\boldsymbol{j}$ is not selected randomly, but determined in some way from the training data. A first possibility in this regard considers only the label data. For example, let r_{ij} denote the correlation between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j, i, j = 1, \dots, L; i \neq j$. Compute $\bar{r}_l = \frac{1}{L-1}\sum_{j \neq l}|r_{lj}|$ for $l = 1, \dots, L$, the average absolute correlation between $\boldsymbol{y}_l$ and all the other label vectors. Suppose $\bar{r}_{l_1} < \bar{r}_{l_2} < \cdots < \bar{r}_{l_L}$. Then the permutation used according to this approach could be defined by $\boldsymbol{j} = [l_1, l_2, \dots, l_L]$. The label that corresponds to the highest average absolute correlation with all other labels is considered last for prediction. This way the most correlated label can use the information from the other labels and learn from them at the training stage.

More generally, the selection of permutation $\boldsymbol{j}$ for the 1-CC classifiers can be done by:

1. *Using only the label variables:*

1.1. *Absolute correlation*

The permutation $\boldsymbol{j}$ is computed as described above.

1.2. *RF importance values*

Let I_{ij} denote the Gini importance value between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j, i, j = 1, \dots, L; i \neq j$, obtained from fitting a RF model to $\boldsymbol{y}_i$ using all other labels as predictors. The Gini measure is described in more detail in Section 4.5.3. Compute $\bar{I}_l = \frac{1}{L-1}\sum_{j \neq l} I_{lj}$ for $l = 1, \dots, L$, the average contribution of all other labels for label l. Suppose $\bar{I}_{l_1} < \bar{I}_{l_2} < \cdots < \bar{I}_{l_L}$, then the permutation is $\boldsymbol{j} = [l_1, l_2, \dots, l_L]$.

Stellenbosch University https://scholar.sun.ac.za

1.3. ReliefF

Let rF_{ij} denote the ReliefF value between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j, i, j = 1, \dots, L; i \neq j$. ReliefF is a multivariate measure of relationship between predictors and a label that takes into account the interaction between predictors. A predictor receives a high value of rF if its values differ for examples from different classes, and gets penalised with a low rF value if its values are different for examples from the same classes (Spolaôr *et al.*, 2013). Compute $\overline{rF_l} = \frac{1}{L-1}\sum_{j \neq l} rF_{lj}$ for $l = 1, \dots, L,$ the average contribution of all other labels for label l. Suppose $\overline{rF_{l_1}} < \overline{rF_{l_2}} < \cdots < \overline{rF_{l_L}}$, then the permutation is $\boldsymbol{j} = [l_1, l_2, \dots, l_L]$.

2. ***Using both input and label variables:***

2.1. *Ordered CC* (Keikha & Hashemi, 2016)

Consider an ML dataset split into training and test sets. Using the BR approach, fit $\boldsymbol{y}_l$ using the training data with RF as the base classifier, for $l = 1, \dots, L$. Predict the corresponding labels for the test dataset, using the fitted BR model. Let $\hat{y}_{il}$ be the predicted value for the i^{th}observation and l^{th} label. Compute the accuracy measure of each of the label predictions, defined as $AC_l = \frac{1}{N}\sum_{i=1}^{N} I(y_{il} = \hat{y}_{il}), l = 1, \dots, L.$ Suppose $AC_{l_1} > AC_{l_2} > \cdots > AC_{l_L}$, then the 1-CC permutation is $\boldsymbol{j} = [l_1, l_2, \dots, l_L]$. The idea is to place the labels that are most likely to be correctly predicted first, so that the subsequent labels can make use of correctly predicted labels.

2.2. *Canonical correlation analysis (CCA)*

Consider a CCA between $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_P]$ and $\boldsymbol{y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_L]$ resulting in the vectors of canonical coefficients $\boldsymbol{a}, \boldsymbol{b}$ that maximise the canonical correlation $\rho = corr(\boldsymbol{a}^T\boldsymbol{x},\ \boldsymbol{b}^T\boldsymbol{y})$. The coefficient b_l represents the strength of label *l* in relation to $\boldsymbol{x}$. Suppose $b_{l_1} > b_{l_2} > \cdots > b_{l_L}$, then the permutation is $\boldsymbol{j} = [l_1, l_2, \dots, l_L]$, so that the label that has the strongest relationship with the features would be first in the sequence and likely be predicted most accurately by the CC.

Stellenbosch University https://scholar.sun.ac.za

In Section 3.1 it was stated that the order of the labels in the label sequence of CC does have an impact on the ML evaluation measures. In the following analysis, the use of a 1-CC classifier with a non-random label sequence was compared to the BR and CC classifiers, in order to see whether the 1-CC approach resulted in better performance. An experiment was conducted, in which the BR, CC as well as the five 1-CC methods were used for ML classification of the emotions dataset. All of the classifiers used RF of 200 trees as the base classifier. The analysis was repeated using 30 random dataset training and test splits. All of the classifiers were fitted on the same data splits. The means and standard deviations of six ML evaluation measures, defined in Section 2.5, were computed for each of the classifiers. The results are summarized in Table 3.1, and the corresponding R code is provided in *Appendix A.2.* All measures except for HL were to be maximised, and the bold value indicates the best classifier for a specific ML measure.

Some of the 1-CC methods performed slightly better than others in terms of the individual evaluation measures. The original CC performed better compared to BR for HL, CL, RE, F2 and AC. The 1-CC methods performed better than CC in terms of CL, RE, F2 and AC, but not for HL and PR. The 1-CC classifier that implemented the CCA method to identify the label sequence did overall consistently well compared to the other 1-CC methods. However, the performance differences based on all measures were relatively small, and further analysis would be necessary to determine whether the differences were significant.

These findings make intuitive sense, as both CC and 1-CC methods make use of the chain structure, whereas BR treats the labels independently. However, using only a single label sequence may not be sufficient for capturing information from the joint conditional distribution of labels. This could explain the relatively small differences amongst the CC and 1-CC methods. The five 1-CC methods summarised above are just a few examples of how a single label sequence method could be applied. All of them identify the best label sequence based on the characteristics of the dataset, prior to fitting an ML classifier.

Stellenbosch University https://scholar.sun.ac.za

Table 3.1: ML classification of the emotions dataset using the BR, CC and 1-CC algorithms.

Mean	HL	CL	PR	RE	F2	AC
$\boldsymbol{BR}$	0.182	0.313	**0.758**	0.619	0.681	0.541
$\boldsymbol{CC}$	**0.181**	0.322	0.746	0.642	0.690	0.559
$\boldsymbol{CC_{cor}}$	0.182	0.324	0.743	0.642	0.689	0.560
$\boldsymbol{CC_{rfimp}}$	0.182	0.320	0.743	0.641	0.688	0.558
$\boldsymbol{CC_{relieff}}$	0.182	0.329	0.735	**0.650**	0.690	0.562
$\boldsymbol{CC_{occ}}$	0.183	0.330	0.730	0.646	0.686	0.561
$\boldsymbol{CC_{cancor}}$	**0.181**	**0.334**	0.738	**0.650**	**0.691**	**0.565**

Std. Dev.	HL	CL	PR	RE	F2	AC
$\boldsymbol{BR}$	0.014	0.034	0.029	0.027	0.026	0.027
$\boldsymbol{CC}$	0.015	0.038	0.031	0.030	0.029	0.030
$\boldsymbol{CC_{cor}}$	0.015	0.032	0.030	0.033	0.030	0.031
$\boldsymbol{CC_{rfimp}}$	0.016	0.042	0.031	0.032	0.030	0.033
$\boldsymbol{CC_{relieff}}$	0.015	0.035	0.029	0.029	0.027	0.030
$\boldsymbol{CC_{occ}}$	0.016	0.036	0.033	0.034	0.031	0.034
$\boldsymbol{CC_{cancor}}$	0.015	0.034	0.032	0.030	0.029	0.030

Another way of determining the best label sequence is by fitting the 1-CC classifier for all permutations of the label sequence, and testing which sequence performs best. In general, one could use *K*-fold cross validation (*K*-CV) to determine the best label sequence from the data in the following way. Let the tuning parameter θ denote a label sequence, and let θ_i be all the permutations of interest, $i = 1, \dots, L!$. The dataset is split into *K* random non-overlapping partitions. Using the k^{th} partition as the test set, the 1-CC classifier using θ_1 is fitted to the other *K*-1 partitions. The classifier's

Stellenbosch University https://scholar.sun.ac.za

performance is evaluated for the test set, using some ML evaluation measure. This process is repeated K times, until all of the folds have been used as the test set. The performance of θ_1 is then quantified as the average performance measure over all K folds. The analysis is repeated for all possible label permutations θ_i, $i = 1, \dots, L!$. The permutation θ that results in the best average performance over all partitions is selected for model fitting.

Using K-CV to determine the best label sequence can be challenging in the following ways. It is clear that this process would be computationally infeasible for a large number of labels, since the increased number of permutations would involve fitting too many CC. Furthermore, evaluating the performance of a classifier using a specific value of θ involves determining which of the ML evaluation measures should be used. As discussed previously, it may be important to consider more than one measure simultaneously. Evaluating the performance by using a few evaluation measures would lead to more computational difficulties. For more information on the topic of finding the best label sequence from the data, methods proposed in the literature that attempt to approximate this process include the double-Monte Carlo scheme (Read *et al.*, 2013) and the probabilistic CC with beam search (Kumar *et al.*, 2012).

3.2.3 LIMITATIONS OF THE CLASSIFIER CHAINS-BASED METHODS

It is essential to mention some of the limitations of the CC-based methods, in order to gain a deeper understanding of ML classification. The limitations of the CC, 1-CC and ECC classifiers are summarised in Table 3.2, and a more detailed description of the individual aspects is provided next.

The CC classifiers use a single random sequence of labels for ML classification. They constrain the label dependencies to follow a single chain-like structure. The first label in the sequence can only use the $\boldsymbol{X}$ predictors, and the last label in the sequence can make use of all the other label information. It is however likely that the label dependence structure is more complex than that of the single sequence. Therefore, using the CC could potentially result in a single sequence bias.

Stellenbosch University https://scholar.sun.ac.za

Table 3.2: Limitations of CC-based methods.

Limitation	CC	1-CC	ECC
Single sequence bias	Bias of a single sequence	Bias of a single sequence	N/A
Random sequence bias	Bias of random sequence	N/A	Bias of random sequences
Other limitations	N/A	Problem of estimation of the best sequence	Computationally demanding

Furthermore, CC classifiers suffer from a random sequence bias, as they do not make use of the label dependence information for the selection of label sequence. Omitting this potentially essential information could lead to sub-optimal predictive performance of the CC classifiers.

Fitting the single best sequence in the 1-CC classifiers results in the same single sequence bias as in the case of CC. The 1-CC classifiers do not choose the label sequence randomly, so the bias of random sequence is not a limiting factor. However, the 1-CC classifiers involve tuning parameter estimation, where methods such as cross-validation are used in order to choose the most suitable parameter estimate. The 1-CC sequence is not a straightforward quantity to determine from the data, and this process becomes computationally challenging for a large number of labels, as discussed previously.

Lastly, the ECC classifiers are not associated with the bias of a single sequence, since they make use of an ensemble of M classifiers. However, each of the classifiers uses a random label sequence and a bias of random sequence is present. The ECC classifier tends to have better performance than a single sequence CC classifier, but it can be computationally demanding in the following ways. The larger the number of CC classifiers in the ECC algorithm, the more computationally intensive the model. Furthermore, the values of M and threshold t are extra parameters that need to be specified or determined from the data, which adds extra computations to the classifier.

Stellenbosch University https://scholar.sun.ac.za

Having considered some of the aspects of the CC-based methods, it appears that all of them suffer from some limitations. An alternative approach that could overcome some of these issues would be preferable. The following section describes the proposed algorithm that combines some of the ideas of the abovementioned CC-based approaches.

3.3 L- CLASSIFIER CHAINS

In this section a new CC-based approach is introduced. The following sub-sections introduce the concept, algorithm and strengths and limitations of this approach, as well as its implementation within a variable importance context.

3.3.1 L-CLASSIFIER CHAINS ALGORITHM

In this section a method that implements a compromise between the simple 1-CC (including the original CC) and the ECC is proposed, called the L-classifier chains (LCC). This method uses an ensemble approach consisting of L semi-random classifier chain sequences to classify ML datasets. Figure 3.3 provides an overview of the adaptations of CC-based methods, including the LCC classifier.

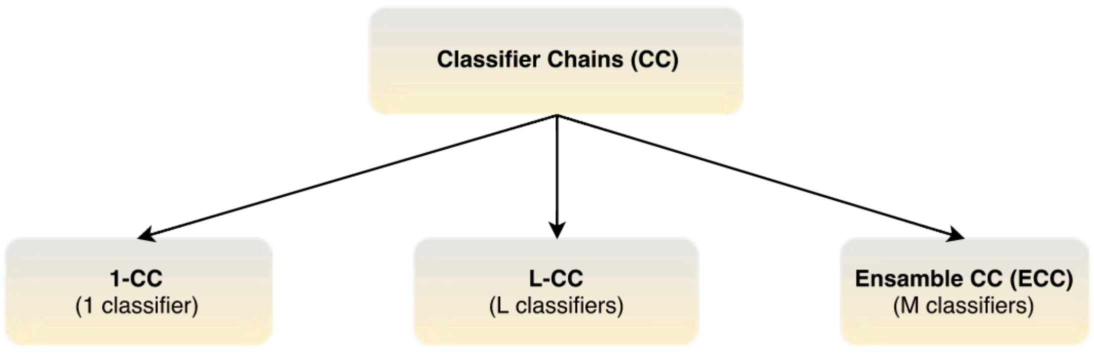


Figure 3.3: Summary of classifier chains modifications.

The key idea of LCC is to introduce some structure into the sequencing process, allowing each of the labels to make use of all other labels. The first classifier in LCC puts the first label in the last place of the chain link, while the other label positions are chosen randomly. Similarly, the second classifier puts the second label at the end of the sequence, choosing the other positions randomly. This is performed L times,

Stellenbosch University https://scholar.sun.ac.za

until all L labels have had the opportunity to be last in the label sequence. The semi-random chaining structure is represented in Table 3.3.

Table 3.3: Training phase of the LCC method.

CC	Chain sequence		Model
$i = 1$	4, 5, 3, 2, 6, …, **1**	$\rightarrow$	$\hat{f}_1$
$i = 2$	5, 3, 4, 6, 1, …, **2**	$\rightarrow$	$\hat{f}_2$
:	:	:	:
$i = L$	4, 1, 2, 5, 3, …, $\boldsymbol{L}$	$\rightarrow$	$\hat{f}_L$

Prediction using the LCC classifier is done in the same was as for the ECC classifier. Each of the L CC classifiers assigns a label vector, $\hat{\boldsymbol{y}}_l(\boldsymbol{x})$, to a new data case with input vector $\boldsymbol{x}$, $l = 1, \dots, L$. The final label prediction requires specification of a threshold t, where $1 \leq t \leq L$. Let $\hat{y}_l(\boldsymbol{x}) = 1$ if and only if the l^{th} component of $\sum_{l=1}^{L} \hat{\boldsymbol{y}}_l(\boldsymbol{x})$ is greater than or equal to t. For example, if $t = 1$, then all labels that were predicted to be present in the test data case by at least one of the L classifiers will be chosen for the final prediction. Similarly, if $t = L$, then all of the classifiers would need to agree and predict the label to be present, in order to include it in the prediction. Therefore, higher values of threshold result in fewer predicted labels. Specification of the threshold can be done in the following way.

The threshold value t for the LCC classifier is determined by using an approach adapted from Read *et al.* (2011). Consider a training and a test dataset $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{N}$ and $(\boldsymbol{x}_i, \boldsymbol{Y}_i)_{i=1}^{N_{test}}$, respectively. Let label cardinality (*LCard*) be the average number of labels per observation, defined as:

$$LCard = \frac{1}{N}\sum_{i=1}^{N} |\boldsymbol{y}_i|.$$

Higher values of *LCard* correspond to a more densely populated label space. Let $LCard_{train}$ be the label cardinality computed for the training dataset. It is of interest to choose t to minimise the difference between the training *LCard* and *LCard*

Stellenbosch University https://scholar.sun.ac.za

computed on the predicted labels. This way, the structure of the predicted label set best resembles that of the training set. It can be implemented within the LCC classifier in the following way.

Fit L CC to the training set, and predict a set of labels for each classifier for all $\boldsymbol{x}_i, i = 1, \dots, N_{test}$, $\widehat{\boldsymbol{y}}_l(\boldsymbol{x}_i),\ l = 1, \dots, L$. For a test case $\boldsymbol{x}_i$, aggregate the labels predicted by the L classifiers into a vector $\boldsymbol{W}_i = (W_{i1}, \dots, W_{iL}),\ \boldsymbol{W}_i = \sum_{l=1}^{L} \widehat{\boldsymbol{y}}_l(\boldsymbol{x}_i),\ l = 1, \dots, L$, $i = 1, \dots, N_{test}$. Let $\boldsymbol{W} = \left(\boldsymbol{W}_1, \dots, \boldsymbol{W}_{N_{test}}\right)$ be a matrix of the aggregated votes for all test observations.

Once the matrix $\boldsymbol{W}$ has been obtained, the $LCard_{test_t}$ for all values of the threshold t are computed in the following way:

$$LCard_{test_t} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} |\widehat{\boldsymbol{y}}_i|,$$

where

$$\text{if } W_{il} \geq t, \qquad \hat{y}_l(\boldsymbol{x}_i) = 1, \qquad \text{else } \hat{y}_l(\boldsymbol{x}_i) = 0,$$

$$for\ i = 1, \dots, N_{test}\ \ and\ \ l = 1, \dots, L.$$

Lastly, the final value of the threshold is selected in the following way:

$$t^* = argmin_t\{|LCard_t - LCard_{train}|\}.$$

Note that the abovementioned way of selecting the threshold assumed a single value of threshold for all labels. Other selection methods that consider a separate threshold for the different labels could be used as well. An example of such a method is selecting the predicted label set, so that the density of the predicted label vector resembles the density of the training label vector the most. These alternative approaches are not explored for the purpose of this thesis.

Stellenbosch University https://scholar.sun.ac.za

The following summary provides a brief description of the LCC training algorithm.

For $i = 1,2,\dots,L$:

1. Let seq be a semi-random sequence. Sample $seq_{[1:(L-1)]}$ randomly and set $seq_{[L]} = i$.
2. Fit a CC classifier $\hat{f}_i$ using label sequence seq.

At test phase, the LCC works on the principle of majority voting, as in ECC. Consider a test dataset $\boldsymbol{X}_{test}$, where $\boldsymbol{x}_i$ is an element of the test dataset. The following steps are required in order to classify $\boldsymbol{X}_{test}$:

---

1. For $l = 1,2,\dots,L$:

   Compute the set of labels $\hat{\boldsymbol{y}}_l(\boldsymbol{x}_i)$, using $\hat{f}_l$ for $\boldsymbol{x}_i$.

2. For case $\boldsymbol{x}_i$ and label l, aggregate the labels predicted by the L classifiers into a vector:

   $\boldsymbol{W}_i = (W_{i1},\dots,W_{iL}),\ \boldsymbol{W}_i = \sum_{l=1}^{L} \hat{\boldsymbol{y}}_l(\boldsymbol{x}_i),\ l = 1,\dots,L,\ i = 1,\dots,N_{test}.$

3. Compute $t^* = argmin_{t\in\{1,2,\dots,L\}}\{|LCard_{test_t} - LCard_{train}|\}$,

   with $LCard_{test_t} = \frac{1}{N_{test}}\sum_{i=1}^{N_{test}}|\hat{\boldsymbol{y}}_i|$,

   where for $i = 1,\dots,N_{test}$ and $l = 1,\dots,L$:

   if $W_{il} \geq t$: $\hat{y}_l(\boldsymbol{x}_i) = 1,$

   else: $\hat{y}_l(\boldsymbol{x}_i) = 0.$

4. The final prediction is:

   if $W_{il} \geq t^*$: $\hat{y}_l(\boldsymbol{x}_i) = 1,$

   else: $\hat{y}_l(\boldsymbol{x}_i) = 0,$

   for $i = 1,\dots,N_{test}$ and $l = 1,\dots,L$:

Stellenbosch University https://scholar.sun.ac.za

3.3.2 IMPORTANT L-CLASSIFIER CHAINS ASPECTS

Theoretical considerations

One of the main disadvantages of CC is the single random label sequence used. Only the label in the last chain link is given the opportunity to use information from all other labels. The ECC classifier overcomes the potential bias of having a single sequence, but uses random selection of sequences instead of making use of the label dependence in the data.

The LCC classifier tries to overcome both of these issues. Firstly, it incorporates an ensemble of CC, like the ECC, in order to prevent the single chain bias. The potential random order bias in ECC is addressed in the semi-random choice of sequences. Therefore, it attempts to avoid the challenging task of estimating the conditional label dependence; instead it imposes a simple structure to the sequencing. The bias might still be present, as the joint conditional distribution is not explicitly modelled. However, due to the structure of the semi-random label sequence, each of the labels can make use of information from all other labels. This structure of the LCC classifier was designed not only to take care of all possible label relationships, but also to allow for an output of variable importance values, as described later in the chapter.

Computational considerations

The LCC classifier aims at finding a balance between CC and ECC. Since it computes L classifiers, it is computationally more intensive than CC. The underlying idea is to make use of the ensemble method of modelling in order to improve the accuracy of CC, while keeping the complexity limited to fitting L classifiers. The semi-random structure of the sequences should provide an improvement in accuracy compared to CC. It is also expected that LCC will perform not worse, if not better than ECC.

Another important aspect in statistical modelling is the ease of fitting a model in an off-the-shelf way. Determining fewer parameters leads to lower probability of incorrect model specification, as well as decreased computational complexity. In

Stellenbosch University https://scholar.sun.ac.za

ECC, two threshold parameters are required: the number of classifiers M and the final label set prediction threshold t. The LCC method only requires one threshold value to be optimised, since the number of classifiers is fixed at a constant value L. The LCC method thus presents a simpler way of using ensembles of CC for ML classification.

LCC limitations

The LCC classifier is only partially non-random and has a strong random component. Moreover, the LCC approach will not be efficient for a very large number of labels, and will be slower than ECC when the same number of classifiers is considered. This is due to the fitting of ECC classifiers on only a subset of the training data, whereas LCC uses the full training set. Compared to ECC, LCC however can effectively output variable importance values, which are relevant for ML inference, as discussed in the next section.

3.3.3 L-CLASSIFIER CHAINS VARIABLE SELECTION AND VARIABLE IMPORTANCE

Variable selection has become one of the most important research areas in statistics, primarily due to increasingly large datasets becoming available. One of the main advantages of VS is reducing the dimensionality of a dataset and thereby making interpretation easier. For example, in an ML context, knowing which of the 5 000 credit data predictors are significant in predicting opened accounts (labels) of clients, can improve the understanding of the credit risk problem. Variable selection also reduces the computational complexity of algorithms that are applied to the data. In some cases it is found that the reduced model based on only the selected variables performs better than the full model in terms of prediction or classification accuracy. In this section an important advantage of LCC when VS is envisaged, is explained. Chapter 4 contains a more detailed discussion of VS.

There are many ways of performing VS. A useful distinction is between selection methods applied as part of pre-processing of the data, and so-called embedded methods. The pre-processing methods provide as output the variables deemed to be important, and these variables are subsequently used in further analyses. An embedded method of VS forms part of a larger algorithm, and variable importance

Stellenbosch University https://scholar.sun.ac.za

values are only a part of the output from this algorithm. A well-known example of the embedded approach is the VS implicitly performed at each of the nodes of a decision tree.

The discussion proceeds with a description of the embedded selection approach when CC are used in an ML context. The output from such an approach can be summarised in an $L \times (P+L)$ matrix, which will be denoted by $\boldsymbol{A}$. The rows of this matrix correspond to the L labels (dependent variables) $Y_1, Y_2, \ldots, Y_L$, while the columns correspond to the P input variables $X_1, X_2, \ldots, X_P$, together with $Y_1, Y_2, \ldots, Y_L$. The entry A_{ij} in the i^{th} row and j^{th} column of $\boldsymbol{A}$ is interpreted as an indication of the importance of X_j $(1 \le j \le P)$ or Y_{j-P} $(P < j \le P+L)$ for Y_i $(1 \le i \le L)$. Note that the last L columns of $\boldsymbol{A}$ are present because of the fact that CC use the label variables $Y_1, Y_2, \ldots, Y_L$ (in some order) as input variables as well.

It seems desirable to treat the L labels symmetrically when variable importance values are determined. This is impossible by using an embedded selection method based on a single classifier chain. Consider for example the 1-CC approach with ordering $(1, 2, \ldots, L)$ for the labels. The model with Y_1 as response is fitted to the training data for Y_1 and only $X_1, X_2, \ldots, X_P$, resulting in importance values of these variables for Y_1 in the first row of $\boldsymbol{A}$. If Y_2 is the response, the model is fitted to the training data for Y_2 and $Y_1, X_1, X_2, \ldots, X_P$, leading to importance values of these variables for Y_2 in the second row of $\boldsymbol{A}$. Finally, if Y_P is the response, importance values for Y_P are obtained corresponding to $X_1, X_2, \ldots, X_P$ and $Y_1, Y_2, \ldots, Y_{L-1}$. An example of the structure of the matrix $\boldsymbol{A}$ is given in Table 3.4.

The need to treat $Y_1, Y_2, \ldots, Y_L$ symmetrically when quantifying importance values is one of the main motivations for introducing the LCC algorithm. Let $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_L$ be $L \times (P+L)$ matrices, corresponding to the different chains in the LCC procedure. Denote the L CC by CC_m, $m = 1, 2, \ldots, L$. According to the definition of LCC, CC_1

Stellenbosch University https://scholar.sun.ac.za

is fit to the data with Y_1 as the last label in the first semi-random permutation. Suppose this semi-random permutation is $\left[l_1, l_2, \ldots, l_{L-1}, 1\right]$. The importance values obtained from this fit are stored in $\boldsymbol{A}_1$, a matrix with structure as described in the previous paragraph. This step gives the following importance values. The last row of $\boldsymbol{A}_1$ will contain importance values for Y_1 from $X_1, X_2, \ldots, X_P$ and all of $Y_2, Y_3, \ldots, Y_L$. The penultimate row of $\boldsymbol{A}_1$ will contain importance values for $Y_{l_{L-1}}$ from $X_1, X_2, \ldots, X_P$ and $Y_{l_j}, j = 1,2,\ldots, L-2$, *i.e.* excluding Y_1 and of course $Y_{l_{L-1}}$ itself. The entries in the other rows of $\boldsymbol{A}_1$ can be interpreted similarly, up to the first row, which will contain importance values for Y_{l_1} from only $X_1, X_2, \ldots, X_P$. A similar interpretation holds for the other matrices, $\boldsymbol{A}_2, \boldsymbol{A}_3, \ldots, \boldsymbol{A}_L$, produced by the LCC sequence with, for example, the last row of $\boldsymbol{A}_L$ containing importance values for Y_L from $X_1, X_2, \ldots, X_P$ and all of $Y_1, Y_2, \ldots, Y_{L-1}$.

It is clear from the above explanation that each of the matrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_L$ will contain importance values for each of $Y_1, Y_2, \ldots, Y_L$ from all of $X_1, X_2, \ldots, X_P$. The final importance value of X_j for Y_l is therefore computed as the average $\frac{1}{L}\sum_{l=1}^{L} A_{lj}$, $l = 1,2,\ldots,L;\ j = 1,2,\ldots,P$. These values can be entered into the L rows and the first P columns of a final $L \times (P+L)$ importance values matrix, denoted by $\boldsymbol{A}_{final}$. The remaining L columns of this matrix are filled by the entries in the last L columns of the last rows of the matrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_L$, *i.e.* the lj^{th} element of $\boldsymbol{A}_{final}$ is given by the Lj^{th} element in the matrix $\boldsymbol{A}_l$, $l = 1,2,\ldots,L;\ j = P+1, P+2, \ldots, P+L$. The structure of the matrix $\boldsymbol{A}_{final}$ is shown in Table 3.5.

Stellenbosch University https://scholar.sun.ac.za

Table 3.4: Matrix of variable importance values for the CC classifier, $\boldsymbol{A}$.

	X_1	X_2	…	X_P	$Y_{seq[1]}$	$Y_{seq[2]}$	…	$Y_{seq[L-1]}$	$Y_{seq[L]}$
$Y_{seq[1]}$	I	I	…	I	-	-	…	-	-
$Y_{seq[2]}$	I	I	…	I	I	-	…	-	-
…	…	…	…	…	…	…	…	…	…
$Y_{seq[L]}$	I	I	…	I	I	I	…	I	-

Table 3.5: Matrix of variable importance values for the LCC classifier, $\boldsymbol{A}_{final}$.

	X_1	X_2	…	X_P	Y_1	Y_2	…	Y_{L-1}	Y_L
Y_1	I	I	…	I	-	I	…	I	I
Y_2	I	I	…	I	I	-	…	I	I
…	…	…	…	…	…	…	…	…	…
Y_L	I	I	…	I	I	I	…	I	-

The following algorithm summarises the LCC approach. To our best knowledge, this way of adapting the original CC has not been proposed in the literature within the context of ML classification and VS.

LCC algorithm:

<u>Input parameters:</u>

- P = number of input variables
- L = number of labels
- train_data = $\boldsymbol{XY}_{train}$: $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N_{train}}$, $\boldsymbol{x}^{(i)}: P\times 1, \boldsymbol{y}^{(i)}: L\times 1$,
- test_data = $\boldsymbol{XY}_{test}$: $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N_{test}}$, $\boldsymbol{x}^{(i)}: P\times 1, \boldsymbol{y}^{(i)}: L\times 1$.
- *notree* = number of RF trees

```
1.    Compute $LCard_{train} = \frac{\sum_{i=1}^{N} |\boldsymbol{Y}^{(i)}|}{N}$.
2.    Initiate matrix $\boldsymbol{W}: N_{test}\times L$ and $\boldsymbol{A}_{final}: L\times(L+P)$.
3.    For $cc = 1, \ldots, L$:
    3.1.  Sample randomly $seq_{[1:(L-1)]}$ and set $seq_{[L]} = cc$.
    3.2.  Initiate matrix $\boldsymbol{A}: L\times(L+P)$.
```

Stellenbosch University https://scholar.sun.ac.za

```
        3.3. For $l = 1, \dots, L$:
            3.3.1. Fit $RF_{seq[l]}$ to $\boldsymbol{y}_{seq[l]}$ using $\boldsymbol{X}_{train}: N_{train} \times (P + i - 1)$ and $notree$ trees.
            3.3.2. Store importance values from $RF_{seq[l]}$ to matrix $\boldsymbol{A}$ (row $l$).
            3.3.3. Compute $\hat{\boldsymbol{y}}_{seq[l]}$ using $RF_{seq[l]}$ and $\boldsymbol{X}_{test}: N_{test} \times (P + i - 1)$.
            3.3.4. Append $\hat{\boldsymbol{y}}_{seq[l]}$ to $\boldsymbol{X}_{test}$.
            3.3.5. Append $\boldsymbol{y}_{seq[l]}$ to $\boldsymbol{X}_{train}$.
        3.4. Update $\boldsymbol{W} = \boldsymbol{W} + \hat{\boldsymbol{y}}_{\text{seq[l]}}$.
        3.5. For $k = 1, \dots, L$, update $\boldsymbol{A}_{final}$ with all values from $\boldsymbol{A}$ that correspond to label $k$. For other labels, using only the last row of importance values of $\boldsymbol{A}$.
    4. Standardise $\boldsymbol{A}_{final_{[,1:\text{P}]}} = \boldsymbol{A}_{final_{[,1:\text{P}]}} / L$.
    5. For $t = 1, \dots, L$:
        5.1. If $\boldsymbol{W} \geq t: \boldsymbol{W}_t = 1$, else $\boldsymbol{W}_t = 0$.
        5.2. Compute $LCard_t = \frac{\sum_{i=1}^{N} |Y_{W_t}^{(i)}|}{N}$.
    6. $t^* = argmin_t\{|LCard_t - LCard_{train}|\}$.
    7. Use $t^*$ to compute the final ML prediction $\hat{\boldsymbol{y}} = I(\boldsymbol{W} \geq t)$
    8. Use $\boldsymbol{y}_{test}$ and $\hat{\boldsymbol{y}}$ to compute the ML evaluation measures.
```

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 4: Variable selection in multi-label classification

4.1 INTRODUCTION

In this chapter, the difficulties presented by high-dimensional problems are identified, including the presence of irrelevant predictors and the curse of dimensionality. Ways of overcoming these problems are described in Section 4.3. Particular attention is paid to one of these methods, namely variable selection. A brief overview of general VS methods is provided, followed by a summary of VS approaches in ML classification. Variable selection, as well as obtaining variable importance values in ML classification is of interest. These goals are explored in detail, and a way of achieving them within the LCC classifier is provided as well.

4.2 HIGH-DIMENSIONAL PROBLEMS

The number of predictors in a dataset can be very large, sometimes up to a few hundred, or even a few thousand. Many of these predictors may not be important for explaining the response variables. In this regard, it is essential to distinguish between irrelevant and redundant variables. Irrelevant or noise variables do not convey any information that could be used to explain the outputs. Including irrelevant features into the model can result in adding noise to the model. If this is the case, the predictive performance of the model deteriorates. Redundant variables may be relevant to explaining the output, but they are not necessarily important for the model. Some of the other variables contain similar information with respect to the outputs. Since redundant features do not add additional information, it is preferable to exclude them from the model. Including noise variables and/or redundant variables in a model can lead to overfitting, which refers to fitting an overly complex model with poor generalisation performance.

Irrespective of whether the predictors are relevant or irrelevant, predictions in high dimensions can be worsened by a phenomenon called the curse of dimensionality (COD). The COD refers to diminishing predictive performance of certain learning

Stellenbosch University https://scholar.sun.ac.za

algorithms in high-dimensional problems. As the dimensionality of a dataset increases, certain algorithms become less effective compared to the same algorithms in two or three-dimensional problems. This is especially true for local models that suffer from some of the manifestations of this curse, as described in the following section.

Firstly, as dimensionality increases, the data observations are no longer local with respect to a target point of interest. The target point is usually a new data case $\boldsymbol{x}_0$ that is to be predicted. Data cases are dispersed more sparsely in higher dimensions compared to low dimensions, if the sample size is kept constant. Due to the increased sparseness of the data, capturing a constant fraction of the data in high dimensions requires covering a wider range of the input variables. Consider an example of classifying a target point, using uniformly distributed inputs from a P-dimensional hypercube. Capturing 10% of the input data in a single dimensional problem corresponds to covering 10% of the input range. In ten-dimensional space, 80% of the input space needs to be covered in order to keep the fraction constant (Hastie *et al.*, 2009:22). While considering data points further away from the target point ensures that a constant fraction of the inputs is taken into account, these points are no longer close to the target point. Therefore, bias of predictions occurs as a result of the COD.

The second way in which the COD manifests itself refers to the data points moving further away from the target point and towards the edge of the sample space. For example, consider N data points that are uniformly distributed in a P-dimensional hyper-sphere. Let the target point be at the origin. The median distance from the target point to the closest data case is $d_{med}(P,N) = \left[1 - \left(\frac{1}{2}\right)^{\frac{1}{N}}\right]^{\frac{1}{P}}$ (Hastie *et al.*, 2009:23). For 100 data points, $d_{med}(1,100)$ is 0.007 in one-dimensional space, and 0.608 in ten-dimensional input space. The closer the data points move towards the edge of the input range, the more difficult it becomes to predict accurately.

The last difficulty resulting from the COD concerns the sampling density in high-dimensional problems. The sampling density d is proportional to $N^{\frac{1}{P}}$ (Hastie *et al.*, 2009:23). In order to maintain a constant d as the sparseness of the data points

Stellenbosch University https://scholar.sun.ac.za

increases, a larger sample size needs to be taken. For example, if 100 observations resulted in a specific sampling density in one-dimensional space, 100^{10} observations would be required in ten-dimensional space to keep the density constant. Therefore, if the number of samples available is limited, the COD occurs and the prediction accuracy deteriorates.

Irrelevant variables and the COD are closely related concepts. The COD may occur even if all of the predictors are relevant, but often both the COD and noise features occur simultaneously. In this case, the high dimensionality of the dataset causes the COD to manifest and the presence of irrelevant predictors adds to the deterioration of prediction accuracy. In order to improve the accuracy of the predictions, the irrelevant variables as well as the COD need to be considered carefully. Variable selection is an effective way of removing noisy input variables, and its application in ML classification is discussed in greater detail in this chapter. Variable selection is also one of the ways of overcoming the COD. However, there are other ways that can deal with this curse. These are discussed in the next section.

4.3 OVERCOMING THE CURSE OF DIMENSIONALITY

Overcoming the COD has become a challenging task in times of increasingly large datasets. It is impossible to ignore this curse, since most practical datasets involve a fairly large number of predictors. For example, the music instrument classification problem can involve hundreds of sound frequencies. In general, some way of reducing the input complexity is essential to overcome the COD. Some of the benefits accompanying this reduction include improved computational time, better interpretability of a model and potentially more accurate predictions.

It is useful to see why local methods suffer from the COD in high dimensions, in order to understand why other methods are less severely affected by the COD. Local methods use non-parametric models, since there is no implied structure of the function $f(\boldsymbol{X})$. These methods are completely data-dependent, which makes them very flexible. On the other hand, they tend to suffer from high variance, due to the lack of restrictions on the structure of $f(\boldsymbol{X})$. One way to overcome the COD is to make structural assumptions about the form of $f(\boldsymbol{X})$. Some of the approaches include

Stellenbosch University https://scholar.sun.ac.za

restriction, regularisation, extraction and selection methods, which are summarised in the next section (Hastie *et al*., 2009:140-141).

The first way of overcoming the COD is by using a restriction method. This refers to using restricted classes of functions $f(\boldsymbol{X})$, for example using a linear discriminant analysis. Imposing a structure on the form of the model results in a reduction in variance of predictions. On the other hand, the fit is not as flexible as with local methods, and a misspecification of the structure can lead to model bias. Therefore, choosing the appropriate structure for a given dataset is essential. Specifying this functional form can become difficult as the dimensionality increases. Furthermore, using the restriction method to overcome the COD in ML classification would require taking into account the complex relationship between P predictors and L labels. Due to the complexity of the ML data structure, it would be challenging to correctly determine $f(\boldsymbol{X})$. Often a simplification of this process is provided via transforming the ML dataset, using one of the PT approaches. Once the data is transformed and simplified, the restricted model is easier to determine.

The COD can also be addressed by applying regularisation methods. These methods restrict the size of the model coefficients. Examples in regression applications include ridge regression, the lasso and splines. By restricting the sizes of the coefficients, the variability of the predictions can be reduced and an overall improved accuracy can be achieved. The amount of shrinkage of the coefficients is controlled by a regularisation parameter λ. The parameter λ can be specified beforehand, or determined from the data using methods such as cross-validation.

Extraction methods aim at overcoming the COD by performing dimensionality reduction in the following way. Consider a dataset of P input variables. Let these variables be replaced by M new variables, where $M < P$. The new predictors are usually extracted by using a linear combination of the original variables, in a way that the loss of information is minimised. The extraction method thus extracts the most important information from the original inputs. Reducing dimensionality of data in this way comes at a cost of information loss, which is often outweighed by overcoming the COD. Once the new predictors have been extracted, a model is fitted and predictions are made in the reduced input space.

Stellenbosch University https://scholar.sun.ac.za

When dependent variables are used to determine the new predictors, the extraction methods are called supervised. Linear discriminant analysis is an example of a supervised extraction method. Unsupervised extraction methods involve the computation of variables based only on the original inputs, disregarding the dependent variables. Principal component analysis is an example of such a method. Other advantages of using extraction methods include faster computations, without much loss of information. Furthermore, plotting the dataset in the reduced space allows for low-dimensional representations of a high-dimensional dataset.

Finally, VS methods are dimensionality reduction techniques of the input space to overcome the COD. Unlike the extraction methods, where all of the predictors are combined into a new set of inputs, the selection methods select the relevant predictors and completely disregard the irrelevant ones. A more detailed description of selection methods, as well their implementation within ML classification is provided in the following sections.

4.4 OVERVIEW OF VARIABLE SELECTION

Variable selection methods aim to identify a subset of relevant features $\boldsymbol{X}' \subseteq \boldsymbol{X}$, from the set of all predictors. Some of the advantages of performing VS include overcoming the COD, removing irrelevant and/or redundant variables, improving computational speed and inference, and improving model accuracy.

Selection methods can successfully remove irrelevant predictors. While irrelevant features do not contribute to the model fit, redundant predictors can still be relevant. For example, two correlated variables may both be relevant, but when considered jointly, one may be redundant. Selection methods can also deal with redundant variables. Redundant variables are often present in high-dimensional data with highly correlated features.

There are many different approaches to VS. Some of the traditional approaches include best subset selection, where all 2^P combinations of inputs are considered in the following way. Firstly, the best set of predictors is identified for each subset size, using training error measures such as multiple R^2. Secondly, the best model out of the $P + 1$ models from the previous step are evaluated in terms of an estimated test

Stellenbosch University https://scholar.sun.ac.za

error measure. This error takes into account the goodness as well as the complexity of fit. An example of such measure is the C_p statistic, $C_p = \frac{1}{N}[\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + 2d\hat{\sigma}_\varepsilon^2]$, where $\hat{\sigma}_\varepsilon^2$ is an estimate of the noise variance (James *et al.*, 2013:211).

The best subset approach is very thorough but inefficient, since the number of fitted models (2^P) becomes very large in high-dimensional problems. Other traditional selection methods include the forward, backward and stepwise selection approaches. Forward selection starts with the null model and successively adds one predictor at a time to the model. At every step, the chosen predictor is the one that is best at improving the model, according to some goodness-of-fit measure. Cross-validation or other information criteria are used to determine how many predictors need to be included in the model. The backward selection method works in a similar way, but starts with the full model, sequentially removing predictors that contribute least to the model. A combination of the forward and backward selection approaches is implemented in the stepwise selection method. In this method, a test is performed at every step, identifying whether any of the variables should enter or exit the model.

Other methods of VS include forward-stagewise selection and the lasso. Forward-stagewise selection uses a slow learning approach to selection. At every step, the predictor that is most correlated with the current residuals is identified. A fraction of the coefficient that results from fitting the chosen predictor to the residuals is then added to the model, keeping the other predictor coefficients unadjusted. This process is repeated until the correlation between the predictors and the residuals is no longer significant. By taking a lot of small steps, the stability and predictions of the model are improved. The concept of slow learning was implemented in boosting, which is one of the most popular methods for regression as well as classification.

The lasso selection method uses both regularisation and selection. It uses the regularisation method to shrink the parameter coefficients towards zero, by minimising $\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{P}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{P}\left|\beta_j\right|$ (James *et al.*, 2013:219). The tuning parameter λ determines the amount of shrinkage applied. If λ is zero, no shrinkage is used, and for $\lambda \to \infty$, coefficients go towards zero until only the null model is reached. In cases where the coefficients reach the zero value, VS occurs.

Stellenbosch University https://scholar.sun.ac.za

Therefore, the fit largely depends on the value of λ, which needs to be pre-specified or determined from the data.

The overview of VS methods provided above is a brief discussion of a very broad topic. For more information on VS methods, see for example the paper by Fan and Lv (2010).

When considering VS in an ML setting, it is important to distinguish between so-called global and local variable importance (Sandrock & Steel, 2016). An explanation of these concepts is provided in the next section, followed by a summary of VS approaches in ML classification.

4.4.1 LOCAL AND GLOBAL IMPORTANCE

The difference between local and global relevance of a predictor is an important concept in ML classification, as explained by Sandrock and Steel (2016). Local relevance refers to the relationship between a given predictor and a single label. A variable is said to be locally relevant for a given label if it can explain the label and contribute to its fit. There are many ways of determining the local relevance of a variable. One option is to compute some measure of the strength of the relationship between the variable and the label. Alternatively, the measure can be obtained from a fitted model.

On the other hand, global relevance of a predictor is a reflection of the relationship between a single predictor and all the labels. A variable is said to be globally relevant for all labels if it can explain the labels effectively. A measure of global relevance may be computed from the individual local relevance values. The experiments reported in the literature most frequently refer to global relevance measures (Spolaôr *et al*., 2013). By using global importance values, it is assumed that each of the labels can be explained by the same set of globally relevant predictors.

The global importance values may therefore be based on an overly simplified assumption about the importance structure. For example, consider a direct marketing ML application. Customers (cases) with spending habits and accounts information (predictors) have opened various debit accounts (labels). For a specific label, say the Woolworths store account, variables such as the number of payments made at

Stellenbosch University https://scholar.sun.ac.za

Woolworths and other similar stores will be more relevant to predicting the opening of a Woolworths account than variables based on spending in other categories. Similarly, a bank account label will be better explained by using bank related variables, whereas the Woolworths-related variables may be completely irrelevant. It may therefore be an over-simplification to identify a single set of globally relevant variables, and force them to be locally relevant for every label at the same time. In this thesis, obtaining local importance values of predictors is of interest. The following section briefly summarises VS methods in ML classification, as well as a method of obtaining the local importance values.

4.4.2 VARIABLE SELECTION IN MULTI-LABEL CLASSIFICATION

The complexity of ML datasets makes the VS process more difficult compared to the binary case. The literature on VS for ML classification is thus not nearly as extensive as for binary classification. Whether the objective of ML VS is to select globally or locally relevant variables, the dataset structure is too complex to perform this selection without any prior data transformation. Therefore, the problem transformation approach to simplify the ML dataset is usually applied prior to the VS step. The ML VS approaches can be summarised in terms of three approaches: filter, wrapper and embedded Spolaôr *et al*. (2013).

Filter methods attempt to remove irrelevant variables by using general characteristics of the dataset. These methods work in the following way. The ML dataset is first transformed into L sets of binary datasets, as in the BR approach. Secondly, the relevance of the first predictor is quantified for every label, according to measures such as ReliefF, Gini index, information gain, etc. This process is repeated for all predictors, so that an importance value is obtained for every predictor with regard to every label. Finally, for every variable, an average of the relevance measures for the L labels is computed, so that each variable is associated with a single value. This average value is one way of obtaining the global relevance measure for that variable. For VS the global relevance values are ordered and only the predictors that correspond to the measures that exceed a threshold value are selected. The threshold can be specified beforehand or determined from the data. Performing VS according to the filter method is not dependent on any specific model fitted to the data. The input

Stellenbosch University https://scholar.sun.ac.za

space is reduced prior to performing ML classification. Applications of the filter approach with information gain include a paper written by Li *et al.* (2014). Another example of these methods is a paper by Spolaôr *et al.* (2013) who compared ML VS methods using the PT approach. Some of the advantages of the filter methods include their simplicity and efficiency. Due to these properties, most of the papers on VS implement filter methods and global relevance.

The second approach to ML VS is the wrapper method. This method selects variables based on a specific algorithm. It resembles the best subset selection approach, since it considers every set of feature variables. The wrapper method considers all subsets of the predictor space and finds the best subset, based on optimisation of an algorithm-specific loss function. There is therefore no pre-processing step as in the filter approach. Similar to best subset selection, the wrapper approach may perform well, but the process of using all subsets of variables becomes computationally intensive in high dimensions. Examples of applications of this approach include papers by Shao *et al.* (2013) and Zhang *et al.* (2009).

Lastly, the embedded methods consider yet another way of performing VS in ML datasets. The embedded methods for VS are built-in within the training phase of a specific learning algorithm. An example of this approach is decision trees classification. Decision tree models are characterised by recursively splitting the predictor space into binary partitions, based on the value of a specific predictor. This predictor is chosen so that splitting on this variable improves the model fit the most compared to the other inputs. Therefore, at the training stage, the decision trees not only fit the model, but also perform VS at the same time. Examples of the embedded approach VS methods include applications of decision trees (Clare & King, 2001) and boosting (Esuli *et al.*, 2008).

In contrast to the filter and wrapper methods, the embedded methods do not explicitly select a subset of variables prior to ML classification. On the one hand, the process can therefore be fast and efficient. On the other hand, the set of globally relevant predictors is not readily available as an output. Having such output is useful for understanding the relationships present in the ML dataset. In decision trees, this problem can however be solved by extracting the importance values from the model after the classification was done. These importance values are the local relevance

Stellenbosch University https://scholar.sun.ac.za

values of interest. The way in which they can be extracted from the decision tree analysis is summarised in the following section, after which the implementation of this approach within the LCC algorithm is provided.

4.5 VARIABLE IMPORTANCE IN MULTI-LABEL CLASSIFICATION

The following section describes decision trees as a way of performing classification. Bagging and RF are introduced thereafter, and the local variable importance values are described as well. Finally, the implementation of RF classification, as well as the variable importance analysis within the LCC classifier, is summarised.

4.5.1 DECISION TREES

Decision trees are designed to fit a highly non-linear function to the training data, while keeping the model simple, intuitive and interpretable. The main idea of this approach is to recursively split the feature space based on $X_1,\, X_2, \dots,\, X_P$ into T unique and non-overlapping regions, $R_1, R_2, \dots, R_T$, and to fit a simple function in these terminal regions or nodes. Observations that fall into region i are assigned a response value that corresponds to that region. The following section provides a more detailed description of the decision trees process for binary classification.

Consider fitting a decision tree model to the training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, followed by predicting response values for a test case $\boldsymbol{x}_0$. Let the response variable assume values $K \in \{1, 2\}$. In general, it is of interest to split the training predictor space into T regions, and to compute the proportion of observations belonging to class 1 and class 2 for every region. The class 1 and class 2 proportions in the m^{th} region are denoted by p_m and $p_m{}'$ respectively, for $m = 1, 2, \dots, T$:

$$p_m = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = 1)\,, \qquad p_m{}' = 1 - p_m,\ \ \forall\, m,$$

where N_m is the number of data cases in region m and R_m consists of data cases that fall into region m.

Stellenbosch University https://scholar.sun.ac.za

Decision trees obtained their name due to the result of the splitting process resembling an upside-down tree, as displayed in Figure 4.1.

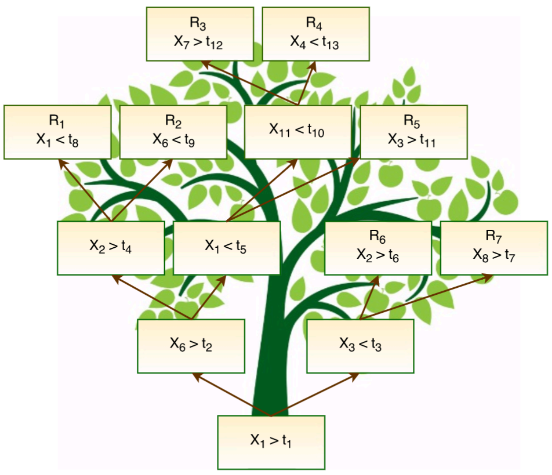


Figure 4.1: Classifier tree structure.

The tree splitting is done in the following way. Initially, all training data is grouped into a so-called root node. The root node is split into two nodes or regions R_1 and R_2, based on the predictor and split-point that results in the best fit. The predictor that can separate the data the best ensures the highest node purity. The purer the nodes are, the more successful the trees become at differentiating between the corresponding groups. The criterion for determining the best predictor and split-point is the measure of node impurity. There are various definitions of this measure. The most well-known measures are the misclassification error, Gini index and cross-entropy (Hastie *et al.*, 2009:309), defined by

$$Misclassification\ error = 1 - max(p_m, p_m'),$$

$$Gini\ index = 2\, p_m\, p_m',$$

$$Cross - entropy = -p_m\, log(p_m) - p_m' log(p_m').$$

The splitting process loops through all predictors and all split-points for each of the predictors, and for every combination the impurity measure is computed. Usually, either the Gini index or cross-entropy are preferred, since they are differentiable and more sensitive to node impurity than the misclassification error (James *et al.*, 2013:312). The predictor and split point that results in the smallest total impurity is

Stellenbosch University https://scholar.sun.ac.za

chosen for the split. The splitting process continues in this way, as long as the number of observations in any of the resulting regions does not become smaller than n_{min}. Therefore, n_{min} determines the size of the tree, and it can be pre-specified or determined from the data by using cross-validation.

Once the decision tree has been grown, the majority class of the training data within region m is computed in the following way, for $m = 1, 2, \dots, T$:

$$k(m) = 1, \qquad if\ p_m > p_m', \qquad else\ k(m) = 2.$$

At prediction time, the test case $\boldsymbol{x}_0$ is allocated a region into which it falls, according to the splitting process of the predictors. The fitted value $k(m)$ of that region is then the predicted response value for $\boldsymbol{x}_0$.

Once the full decision tree has been grown, the number of interactions or splits can be fairly large. These complex and flexible trees likely result in low bias but potentially high variance of predictions. Even a small change in the training data can result in different node splits. Overall, the decision trees are known to be unstable. One way of overcoming this problem is to construct simpler trees by pruning the full trees. Cost-compexity pruning within the cross-validation framework is typically used to find the best tree size. These procedures will not be discussed in greater detail, as the experimental work in this thesis makes use of full-grown trees.

Trees are generally considered to be one of the best off-the-shelf learning methods. They are fast, interpretable and can perform VS based on the splitting process. However, trees suffer from high variance and sub-optimal predictive performance. Various techniques for overcoming this downside have been proposed, including bagging, random forests and boosting. These methods typically combine multiple trees and lead to improved performance at the cost of diminished interpretability. The approach of bagging is described in the following section.

4.5.2 BAGGING

The method of bagging was designed to improve the performance of a learning algorithm. Within the framework of decision trees, bagging uses the bootstrap approach to reduce variance and improve predictive performance of trees. The

Stellenbosch University https://scholar.sun.ac.za

method of bagging can be explained in the following way. Let the training dataset be denoted as $\boldsymbol{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_N)$, $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, and consider a set of bootstrap samples, $\boldsymbol{Z}^{*b} = (\boldsymbol{z}_1^{*b}, \boldsymbol{z}_2^{*b}, \dots, \boldsymbol{z}_N^{*b})$, $b = 1, \dots, B$, randomly drawn with replacement from the training data $\boldsymbol{Z}$. A decision tree is fitted to each of the B samples. Let $\hat{\boldsymbol{f}}^{*b}(\boldsymbol{x}_0)$ be a vector denoting the predicted class for a test case $\boldsymbol{x}_0$, based on the b^{th} bootstrap sample. More specifically, $\hat{\boldsymbol{f}}^{*b}(\boldsymbol{x}_0) = [0, 1]$ if $\boldsymbol{x}_0$ is assigned to class 2 and $\hat{\boldsymbol{f}}^{*b}(\boldsymbol{x}_0) = [1, 0]$ otherwise.

At prediction time, the prediction for $\boldsymbol{x}_0$ is computed in the following way. Let the proportions of the bootstrap models that predicted the response value as class 1 and class 2 be denoted as $p(\boldsymbol{x}_0)$ and $p'(\boldsymbol{x}_0)$ respectively. The final bagging proportions are based on a committee of trees. The proportions of the bootstrap models that assign the new case $\boldsymbol{x}_0$ to classes 1 and 2 can be computed as:

$$\hat{\boldsymbol{f}}_{bag}(\boldsymbol{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{\boldsymbol{f}}^{*b}(\boldsymbol{x}_0) = [p(\boldsymbol{x}_0), p'(\boldsymbol{x}_0)].$$

Note that the bagging estimate $\hat{\boldsymbol{f}}_{bag}(\boldsymbol{x}_0)$ is a Monte Carlo approximation of the true bagging estimate, which is based on all N^N possible samples. The final bagging prediction for $\boldsymbol{x}_0$ is given as:

$$\hat{G}_{bag}(\boldsymbol{x}_0) = argmax_{k\in\{0,1\}} \hat{\boldsymbol{f}}_{bag}(\boldsymbol{x}_0)[k], \qquad \text{(Hastie } et\ al., 2009{:}\,283).$$

The averaging process of bagging results in a variance reduction of decision trees. The bias of the aggregated predictions is constant compared to a single tree, due to the identical distributions of the bootstrap samples. It is important to note that bagging works only if non-linear or adaptive functions of the data are fitted. These functions typically have high variance and low bias, making them ideal candidates for bagging. On the downside, if the individual classifiers are weak, bagging might make the final classifier worse.

In classification, bagging can be viewed as a consensus decision based on a set of independent weak learners. However, bagged trees cannot be considered independent, since the bootstrap sampling is done with replacement and the samples

Stellenbosch University https://scholar.sun.ac.za

partially overlap. Random forests have been introduced as a way of de-correlating the trees, and they are summarised in the next section.

4.5.3 RANDOM FORESTS

Random forests were introduced by Breiman in 2001 and have since become widely popular. The main contribution of RF lies in the de-correlation process of bagged trees. Recall that the bagged trees were not independently distributed, due to the structure of bootstrap sampling. Consider a set of identically distributed random variables $X_1, X_2, \dots, X_B$ with variance $Var(X_i) = \sigma^2$ and covariance $Cov(X_i, X_j) = \rho\sigma^2 \; \forall \; i, j; \; i \neq j$. The averaging process in bagging corresponds to the averaging of $X_1, X_2, \dots, X_B$. The variance of an average of B such random variables can be computed as:

$$Var(\bar{X}_{id}) = \rho\sigma^2 + \tfrac{1-\rho}{B}\sigma^2.$$

Note that for $Var(\bar{X}_{id}) > 0$, the correlation ρ must be exceed $-\frac{1}{B-1}$, which goes to 0 if $B \to \infty$. This makes intuitive sense, since the bagged trees are partially overlapped. For $\rho \geq 0$, the variance of the mean of correlated variables is larger than that of uncorrelated or independent variables:

$$Var(\bar{X}_{id}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 > Var(\bar{X}_{iid}) = \frac{1}{B}\sigma^2.$$

The main objective of bagging is to use the averaging process to reduce variance. However, bagging only manages to achieve $Var(\bar{X}_{id})$ and not $Var(\bar{X}_{iid})$, since the samples are not independent. By considering $Var(\bar{X}_{id})$ it can be seen that increasing the number of samples B can reduce this quantity, but it cannot reduce the correlation ρ. It is therefore essential to try to change the individual trees slightly, in order to decrease the correlation between them, without increasing the variance. One method of achieving this is implemented in the RF approach (Breiman, 2001), as described in the following section.

Random forests de-correlate bagged trees by restricting the information available at each split of a tree. Instead of choosing the best split based on all P predictors, only

Stellenbosch University https://scholar.sun.ac.za

Mtry random variables are considered at a time, *Mtry* < *P*. In this way RF essentially force the trees to be less similar and less correlated. The training RF algorithm can be summarised by the following steps according to Hastie *et al*. (2009:588):

```
For $b = 1,2,\ldots,B$:

1.  Draw a bootstrap sample $\boldsymbol{Z}^{*b}$ from $\boldsymbol{Z}$.
2.  Fit a decision tree to $\boldsymbol{Z}^{*b}$ by repeating:
    2.1.  Randomly select $Mtry$ variables out of all $P$ variables.
    2.2.  Determine the best variable and split-point from the $Mtry$ candidates.
    2.3.  Split node into two sub-nodes.
3.  Output the tree based on the bootstrap sample.
```

Predicting a new input case $\boldsymbol{x}_0$ is done in the following way:

$$\hat{\boldsymbol{f}}_{rf}(\boldsymbol{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{\boldsymbol{f}}^{*b}(\boldsymbol{x}_0) = [p(\boldsymbol{x}_0),\, p'(\boldsymbol{x}_0)],$$

$$\hat{G}_{rf}(\boldsymbol{x}_0) = argmax_{k\in\{0,1\}}\hat{\boldsymbol{f}}_{rf}(\boldsymbol{x}_0)[k],$$

where $p(\boldsymbol{x}_0)$ and $p'(\boldsymbol{x}_0)$ are the proportions of the RF models that predicted class 1 and class 2 for $\boldsymbol{x}_0$ respectively. The values of the parameters n_{min} and *Mtry* can be pre-specified or determined from the data. In classification, the default values for n_{min} and *Mtry* are 1 and $\sqrt{P}$ respectively.

Random forests are successful at fitting complex models, while implicitly performing VS at the same time. Therefore, they can be viewed as an embedded VS method. In order to implement RF within the LCC framework, the base learner for every CC is set to be a random forest. This way of performing ML classification accounts for VS at the same time, resulting in all of the advantageous properties of selection methods mentioned previously. However, in order to determine which predictors are locally relevant for the labels, the following quantities need to be computed Hastie *et al*. (2009:368):

Stellenbosch University https://scholar.sun.ac.za

For $b = 1, 2, \dots, B$:

1. Fit a decision tree to $\boldsymbol{Z}^{*b}$.
2. Compute the importance measure for $X_p,\ p = 1, \dots, P$:

$$I_p^2(b) = \sum_{i=1}^{T-1} \hat{\imath}_i^2 I(X(i) = p).$$

The RF importance values based on all B trees are:

$$I_p^2 = \frac{1}{B} \sum_{b=1}^{B} I_p^2(b).$$

The individual tree importance measures $I_p^2(b)$ consider the splitting variables at each of the $T - 1$ internal nodes of the tree. In instances where the variable of interest X_p corresponds to the splitting variable $X(i)$, the importance measure for X_p is increased by the improvement $\hat{\imath}_i^2$ of the fitted model. This improvement in the model fit results from using variable $X(i)$ for the split. In classification, the improvement measure can be the misclassification error, Gini index or cross-entropy. Therefore, the importance of X_p is measured as the sum of improvement values over all nodes for which X_p was chosen at the split. This makes intuitive sense, since the variable chosen at each split corresponds to the largest improvement in fit compared to all other variables. Finally, the resulting importance measure for X_p is computed by averaging the importance values over all B trees, to give I_p^2.

4.5.4 RANDOM FORESTS APPLICATION IN L-CLASSIFIER CHAINS

Implementing RF variable selection within the LCC framework offers a few attractive properties. Using RF as base classifier in each of the L classifier chains ensures that the labels added to the feature space are only selected if they seem relevant. Additionally, even if a label is predicted correctly, but the dependence between the previous labels and the current label is not strong, the RF does not select it for splitting. Therefore, if no previous labels are chosen for prediction, the procedure essentially follows the BR classification approach.

Stellenbosch University https://scholar.sun.ac.za

It is important to make a distinction between VS and variable importance. Variable selection assigns a binary value to a predictor, identifying it as either relevant or irrelevant. Random forests perform VS during the classification process by selecting variables at every tree split. Variable importance assigns an importance value to every predictor, indicating how important a given variable was, based on the classification model fitted. Therefore, the importance values obtained from the RF models can be analysed after the model was fitted.

While classifiers such as BR, CC and ECC can implement RF as a base classifier to perform VS, there is no explicit way to do variable importance analysis based on the model. The importance values could be obtained for all of the classifiers, but only LCC is designed in a symmetric way to account for all the relationships between predictors and labels, as explained in Section 3.3.3. Note that ECC could also be used to construct the $\boldsymbol{A}_{final}$ matrix, but the analysis would not be as efficient as for LCC. The importance analysis requires every label to be predicted last for at least one classifier chain. The ECC classifier chooses the label sequences randomly, so the number of classifiers needed for this purpose can become large, especially for datasets with many labels. Due to the structure of the semi-random chain sequencing of LCC, the matrix of importance values $\boldsymbol{A}_{final}$ can be constructed for all local importance values using only L classifiers. Once the importance values are obtained from the RF base classifier, the construction of $\boldsymbol{A}_{final}$ follows the methodology summarised in Section 3.3.3. In the following chapters implementation of the BR, CC, ECC and LCC classifiers is presented. Benchmark datasets analysis, simulation analysis as well as practical dataset analysis are done in order to develop a deeper understanding of the ML classification methods and the variable importance values based on the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 5: Benchmark datasets analysis

5.1 INTRODUCTION

The benchmark datasets analysis is the first part of the experimental work done in this thesis. The main idea of using these datasets is to apply some of the ML techniques to real ML datasets. Furthermore, these datasets are readily available online and are often used in research papers. The objective of this analysis is to compare the BR, CC, ECC and LCC classifiers in terms of how accurately they classify the ML datasets. It is also of interest to obtain variable importance values from the LCC classifier, and to see whether label dependencies are present in ML datasets.

5.2 DATASETS

The experiments were carried out using three ML benchmark datasets, obtained from the *mldr.datasets* R library (Charte, 2016). These specific datasets were selected because the number of predictors and labels of the datasets were not too large and the classification analysis could be performed efficiently. A brief description of the characteristics of these datasets appears in Table 5.1. In this table, each dataset is assigned a value of label cardinality (*LCard*), label density (*LDens*) and the number of distinct label combinations. *LCard* is defined in Section 3.3.2, and *LDens* is the proportion of labels present in a dataset. It can be computed by using the following definition:

$$LDens = \frac{1}{LN}\sum_{i=1}^{N}\sum_{j=1}^{L} Y_{ij}.$$

The emotions dataset contains 72 feature variables extracted from music tracks. The six labels that represent the emotions that the specific tracks evoke are: *amased-surprised*, *happy-pleased*, *relaxing-calm*, *quiet-still*, *sad-lonely* and *angry-aggressive*. There are on average 1.87 emotions recorded per music track. The dataset contains 27 unique combinations of emotion labels and the label density is 0.31.

Stellenbosch University https://scholar.sun.ac.za

The scene dataset is an example of an image annotation ML dataset that assigns keywords to images with different scenes based on 294 attributes. The six label categories are: *Beach*, *Sunset*, *Fall Foliage*, *Field*, *Mountain* and *Urban*. Each image is on average labelled by 1.07 keywords. The dataset has label density of 0.18, and there are altogether 15 distinct keyword combinations.

Finally, the yeast dataset is from the biological domain, with focus on protein profiles. These profiles are associated with 103 feature variables and 14 protein labels. There are 198 unique label combinations that are populated with a density of 0.3. Each protein profile is on average associated with 4.24 labels.

Table 5.1: Summary of benchmark datasets used in experimental study.

Dataset	***N***	***L***	***P***	***Domain***	***LCard***	***LDens***	***Distinct***
Emotions	593	6	72	Music	1.87	0.31	27
Scene	2407	6	294	Image	1.07	0.18	15
Yeast	2417	14	103	Biology	4.24	0.30	198

5.3 EXPERIMENTAL DESIGN

The benchmark datasets analysis was performed for the three datasets separately. Every dataset was split into training and test set using the ratio 0.75/0.25. For each split, the classifiers BR, CC, LCC and ECC were fitted to the training set, each of which resulted in a set of predicted labels for the test set. The classifiers thus used exactly the same datasets. The ECC method fitted $M = L$ classifiers, and each of the classifiers was fitted on a random set of 67% of the training data. The parameter $M = L$ was chosen so that the computations were comparable to the L classifiers fitted by LCC. All of the classifiers used RF of 200 trees as the base classifier, and the value of *Mtry* was set to the default $\sqrt{P}$. Note that in the CC classifiers the value of P includes all the labels in the feature space at any point of the chain sequence. It is therefore not constant for the CC-based methods. The Gini index was used as a criterion for growing trees and for obtaining the variable importance values throughout the analysis.

Stellenbosch University https://scholar.sun.ac.za

The analysis was repeated 30 times for each dataset. For each run, the six evaluation measures, defined in Section 2.5, for every classifier were recorded, and the corresponding mean and standard deviation of the measures were computed. For every LCC classifier, the $\boldsymbol{A}_{final}{}_{LCC_i}$ matrix of importance values was obtained as defined in Section 3.3.3, for $i = 1, \dots, 30$. Finally, an $\boldsymbol{A}_{global}$ matrix was computed as the average matrix based on the 30 runs, $\boldsymbol{A}_{global} = \frac{1}{30}\sum_{i=1}^{30} \boldsymbol{A}_{final}{}_{LCC_i}$. Lastly, the Gini importance values in the $\boldsymbol{A}_{global}$ matrix were standardised by dividing each row of the matrix by the largest Gini value corresponding to that row. Therefore the Gini values were standardised to an index ranging from 0 to 100, where 100 represented the most important variable for the given label. The $\boldsymbol{A}_{global\_std}$ matrix thus represents the local relevance values for each predictor, including all the other labels. This matrix is used in the following section to explore the variable importance values based on LCC in more detail. The R code used for classification of the benchmark datasets is provided in Appendix B.1.

5.4 EXPLORATORY ANALYSIS

In the exploratory analysis, the results obtained from the benchmark datasets classification are analysed. The analysis is split into three sub-sections, each corresponding to one of the datasets. Analysing only the three ML datasets allowed for a more detailed exploratory analysis of the results. Graphical summaries of the results are provided for each dataset, including boxplots of ML evaluation measures for each classifier, density plots of variable importance indices and heatmap plots of the $\boldsymbol{A}_{global\_std}$ importance matrices. The R code used for the exploratory analysis of the benchmark datasets is provided in Appendix B.2.

5.4.1 EMOTIONS DATASET

The 30 classification runs for the emotions dataset resulted in six ML evaluation measures for every run, as summarised in the boxplot graphs in Figure 5.1. Previously done ML research concluded that for correlated labels the CC could perform better than BR, and ECC could be even more effective than CC, as described

Stellenbosch University https://scholar.sun.ac.za

in Chapter 3. It is of interest to observe how LCC performed compared to the other classifiers.

In terms of the HL and CL measures, the differences were fairly small. For the inversely related PR and RE measures, two groups consisting of (BR, CC) and (ECC, LCC) formed. By observing the information from PR and RE combined in the F2 measure, it could be seen that there was an increasing trend in favour of LCC. A similar trend occurred for the AC measure. Overall, LCC seemed to be a competitive method for ML classification based on the emotions dataset, although the differences between the procedures were fairly small.

In Figure 5.2, the estimated Gaussian kernel density plots of the standardised variable importance values are displayed for every label. These values were obtained from $\boldsymbol{A}_{global\_std}$, an output from the LCC classifier. Note that out of the four classifiers, only LCC provided such output, and it was explored in more detail. The importance values in $\boldsymbol{A}_{global\_std}$ were averaged over all RF trees, L classifier chains and 30 runs.

An arrow in Figure 5.2 indicates the most important predictor for each label. The most important predictor of index value 100 corresponded to another label in four out of the six cases. This indicated that the label dependence was a determining factor for the LCC classifier. The density plots are all positively skewed, thus the majority of the predictors had low importance index values compared to the most important predictor. The densities for label 1 and 2 followed a slightly different pattern, where more of the predictors showed moderately large indices. Figure 5.2 is useful for understanding the complexity of the ML datasets, since it shows that the relationship between input predictors and labels can be different for every label and that the presence of other label predictors can have an impact on the response labels as well. For the emotions dataset, a case was made for the importance of VS, since the majority of predictors added little value to the fit.

Stellenbosch University https://scholar.sun.ac.za

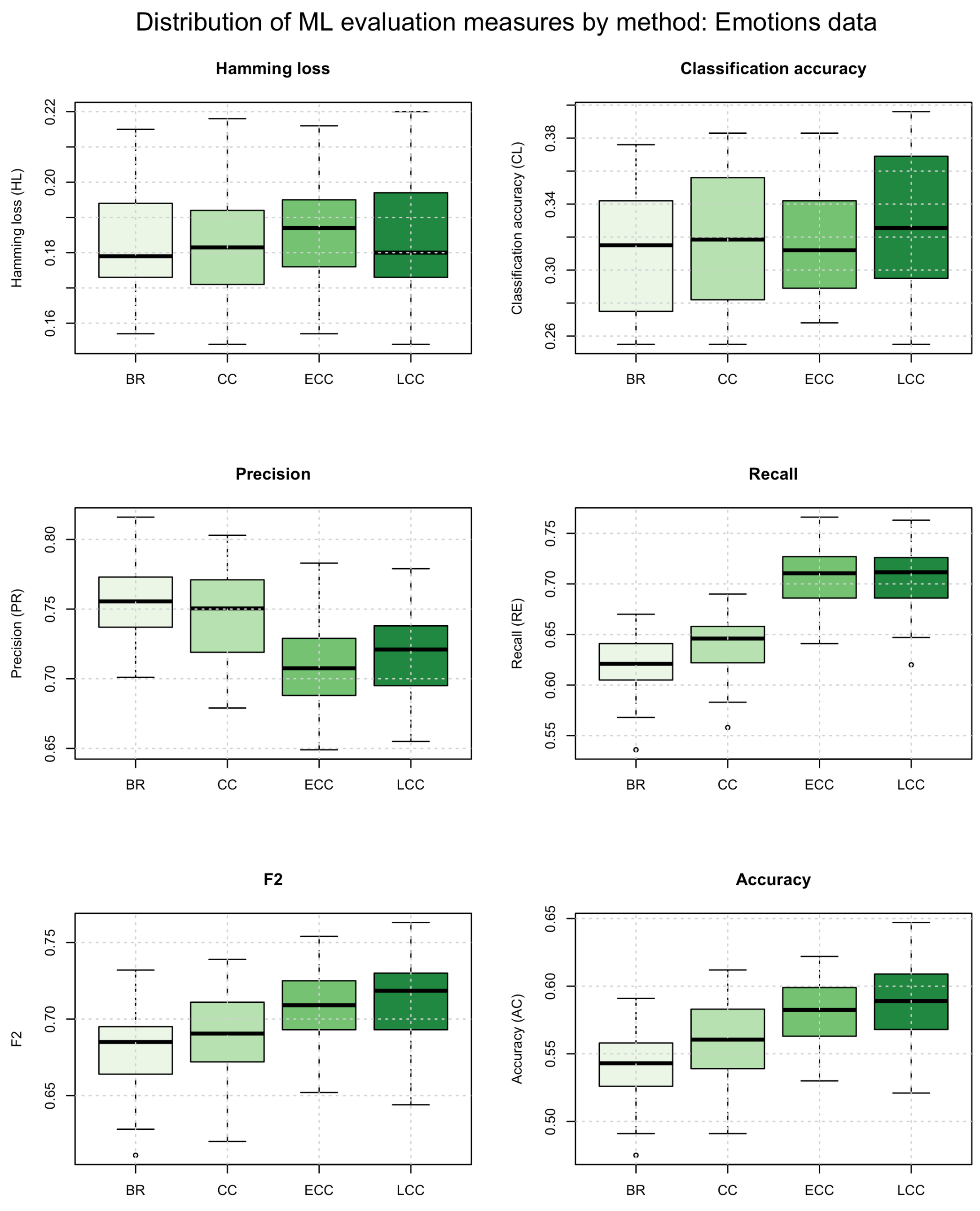


Figure 5.1: Emotions dataset: boxplots of ML evaluation measures for the four classifiers.

Stellenbosch University https://scholar.sun.ac.za

The last graphical representation of the LCC importance values is given in Figure 5.3. The heatmap graphed in the figure is a useful way of summarising the LCC variable importance indices. Compared to the density plots in Figure 5.2, it displays every relationship between labels and predictors, including all the other labels viewed as predictors. This figure could be constructed due to the symmetry of the LCC classifier.

By comparing the local relevance values across labels, it is clear that not all predictors were equally important for the labels, and that only a minority of predictors showed moderate or strong relationship with the labels. Figures 5.2 and 5.3 are directly related, since for example, both graphs indicate that there were a lot more important predictors for label 1 than for label 4. Furthermore, the heatmap reveals that the best predictors marked by the black bar were one of the other labels in nearly all cases. This information explains why the BR method in Figure 5.1 performed relatively badly - it did not take the label dependence information into account.

The dendrogram of the labels is displayed above the graph columns in Figure 5.3. It groups the labels based on a hierarchical cluster analysis of the similarities of the corresponding importance values. The six labels were initially split into two clusters: labels 1, 6 (*amased-surprised*, *angry-aggressive*), and the other four labels. These four labels were further split into labels 4, 5 (*quiet-still*, *sad-lonely*) and labels 2, 3 (*happy-pleased*, *relaxing-calm*). The clusters are intuitively sound, and provide an additional layer to the analysis of the results given by LCC.

Stellenbosch University https://scholar.sun.ac.za

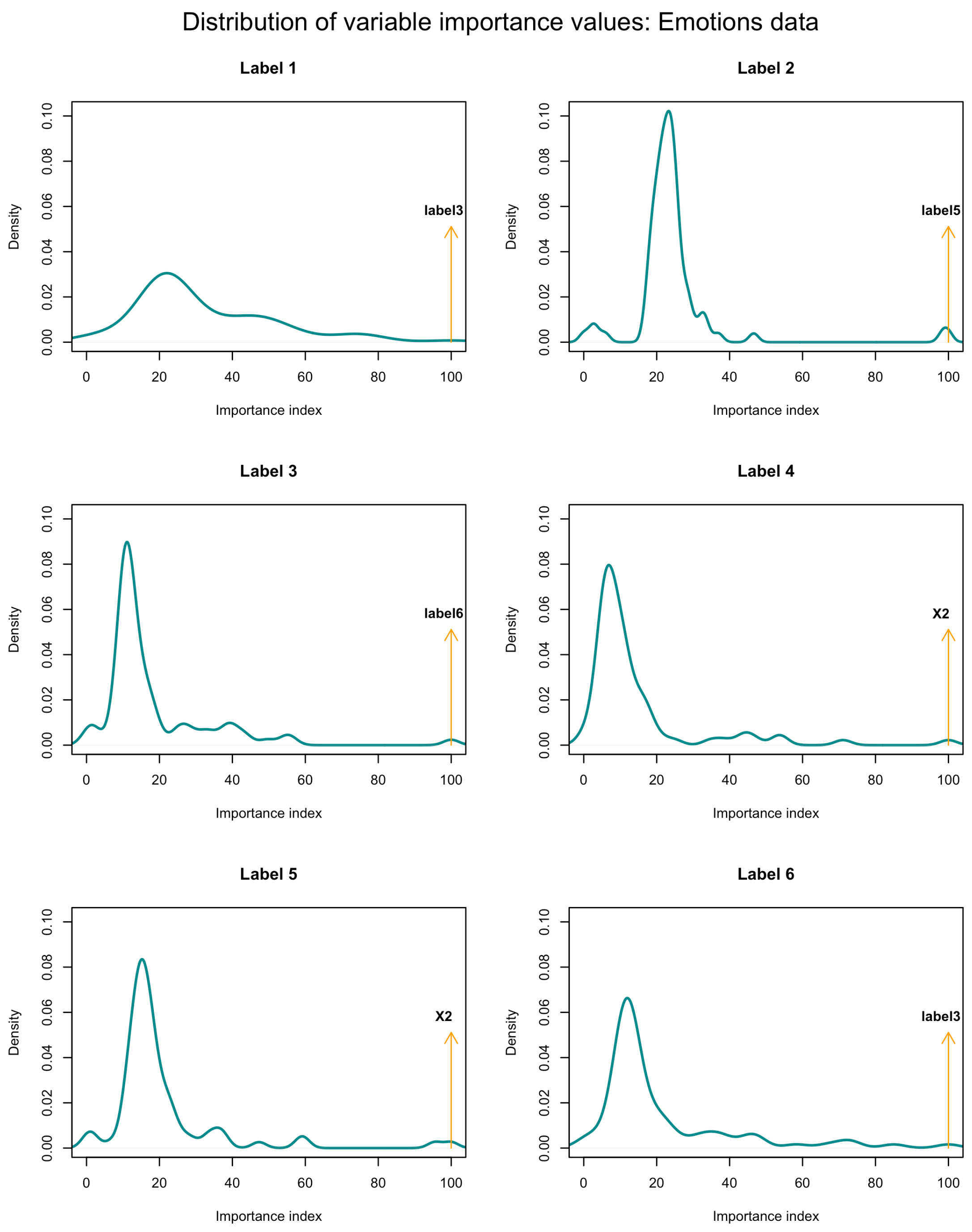


Figure 5.2: Emotions dataset: distribution of importance values for the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

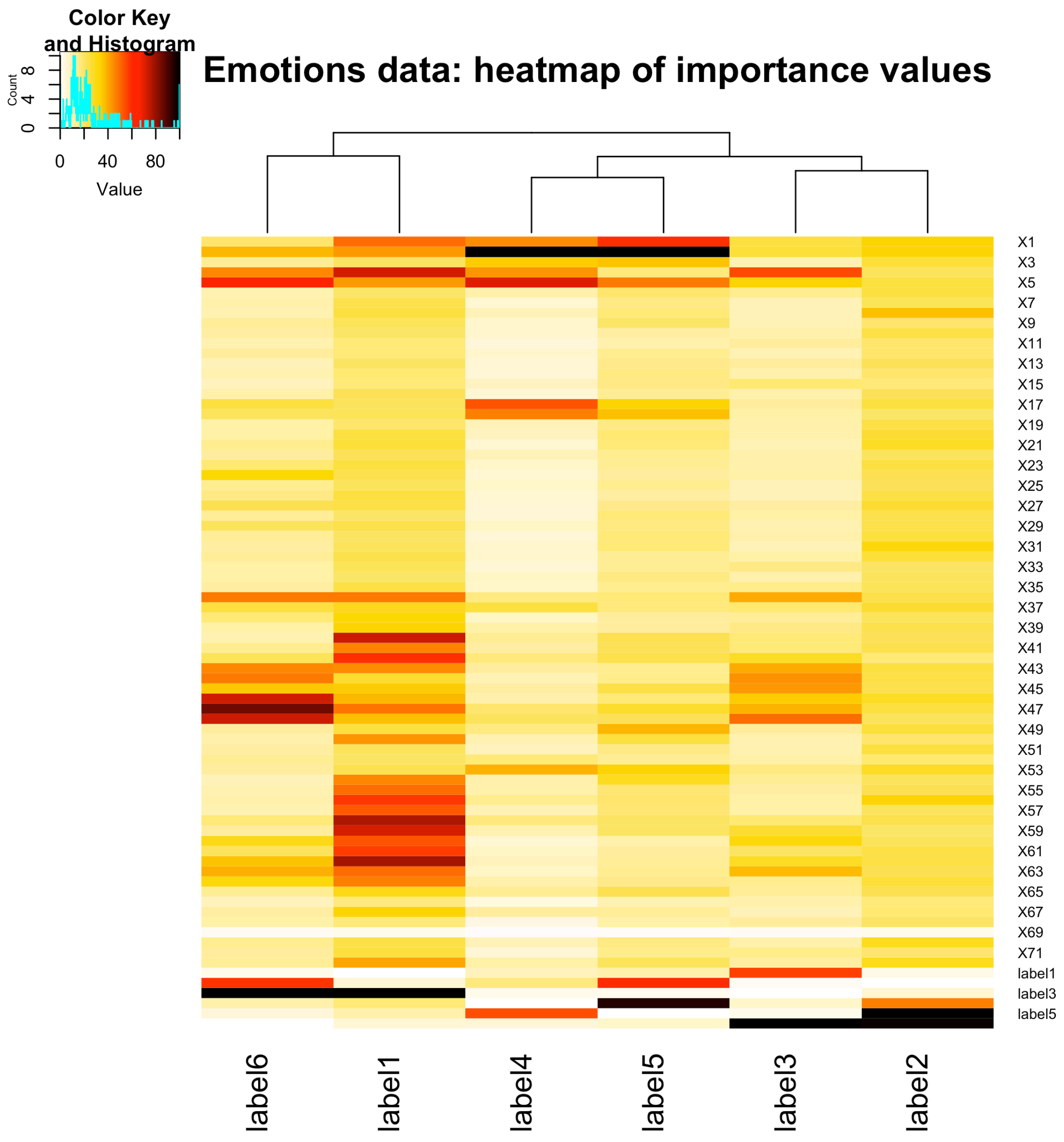


Figure 5.3: Emotions dataset: heatmap of importance values for LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

5.4.2 SCENE DATASET

Exploratory analysis of the results from the four classifiers was carried out for the scene data, as summarised in Figures 5.4, 5.5 and 5.6. The boxplots in Figure 5.4 show a clear distinction between the four classifiers. In terms of the HL, LCC performed the best, followed by ECC, CC and BR. There is also a clear upward trend for the CL measure, where LCC achieved the highest proportion of exact label matches. ECC and LCC on average predicted more labels per case than BR and CC, as reflected in lower values of PR. There was a grouping of methods in the RE, F2 and AC measures, formed by the (BR, CC) and (ECC, LCC) groups. In the latter group, the LCC method performed slightly better compared to the ECC. While CC performed slightly better than BR, ensemble methods ECC and LCC captured the ML relationships more accurately.

The density plots in Figure 5.5 indicate that the variable importance indices for labels 1, 2 and 3 were mostly low, represented by the positively skewed distributions. Indices for labels 4, 5 and 6 were even more strongly positively skewed, with a higher proportion of variables with very low importance indices. The graphs in Figure 5.5 are more skewed than the graphs for the emotions data, which could be explained by the fact that there were 294 input variables in the scene data versus 72 in the emotions data. Three of the labels used another label as the most important variable. The densities of the scene data also confirmed the importance of VS in ML classification.

Figure 5.6 depicts the heatmap plot of the scene variable importance indices. In this plot, the information from the density plots summarised in Figure 5.5 can be clearly seen. There were very few moderately and highly important variables, most of which were grouped in a block of variables. Labels 1 and 4 did not make extensive use of other labels, whereas labels 5 and 6 were almost fully explained by each other, as seen in Figure 5.5. Labels 2 and 3 were also highly important for each other’s fit. Therefore, label dependence was an important factor in this dataset. Ignoring this dependence resulted in less accurate predictions, as in the case of BR in Figure 5.4.

Stellenbosch University https://scholar.sun.ac.za

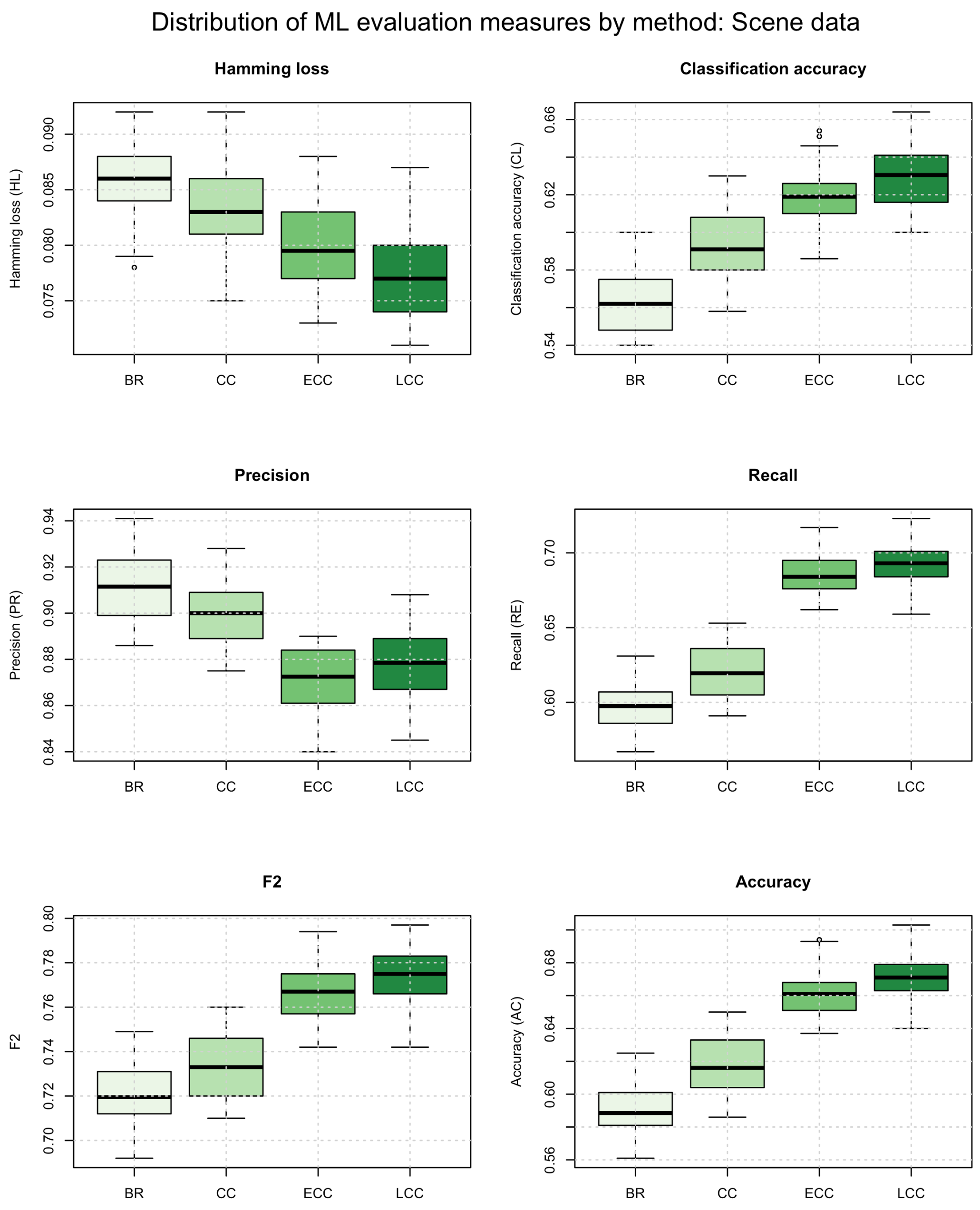


Figure 5.4: Scene dataset: boxplots of ML evaluation measures for the four classifiers.

Stellenbosch University https://scholar.sun.ac.za

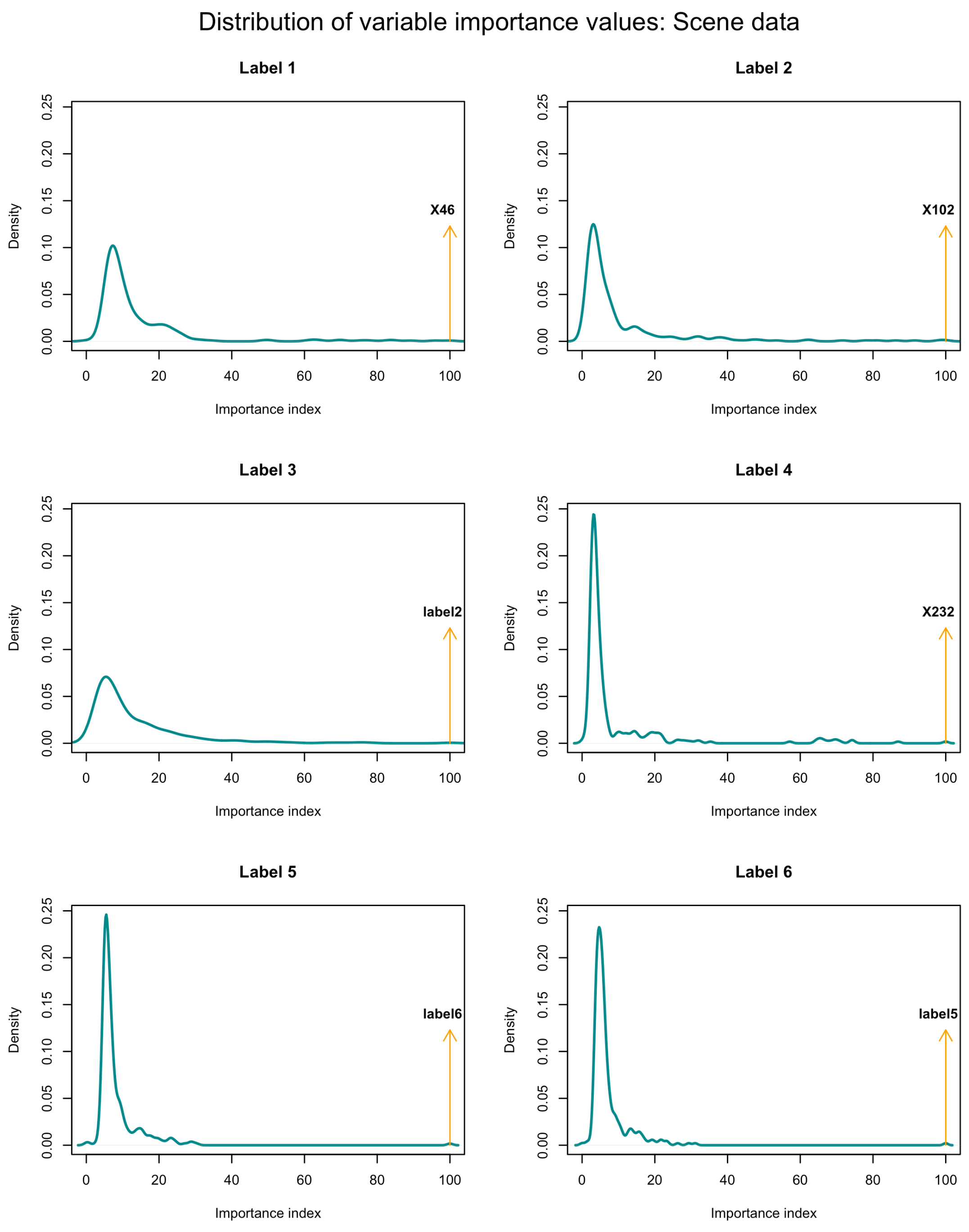


Figure 5.5: Scene dataset: distribution of importance values for the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

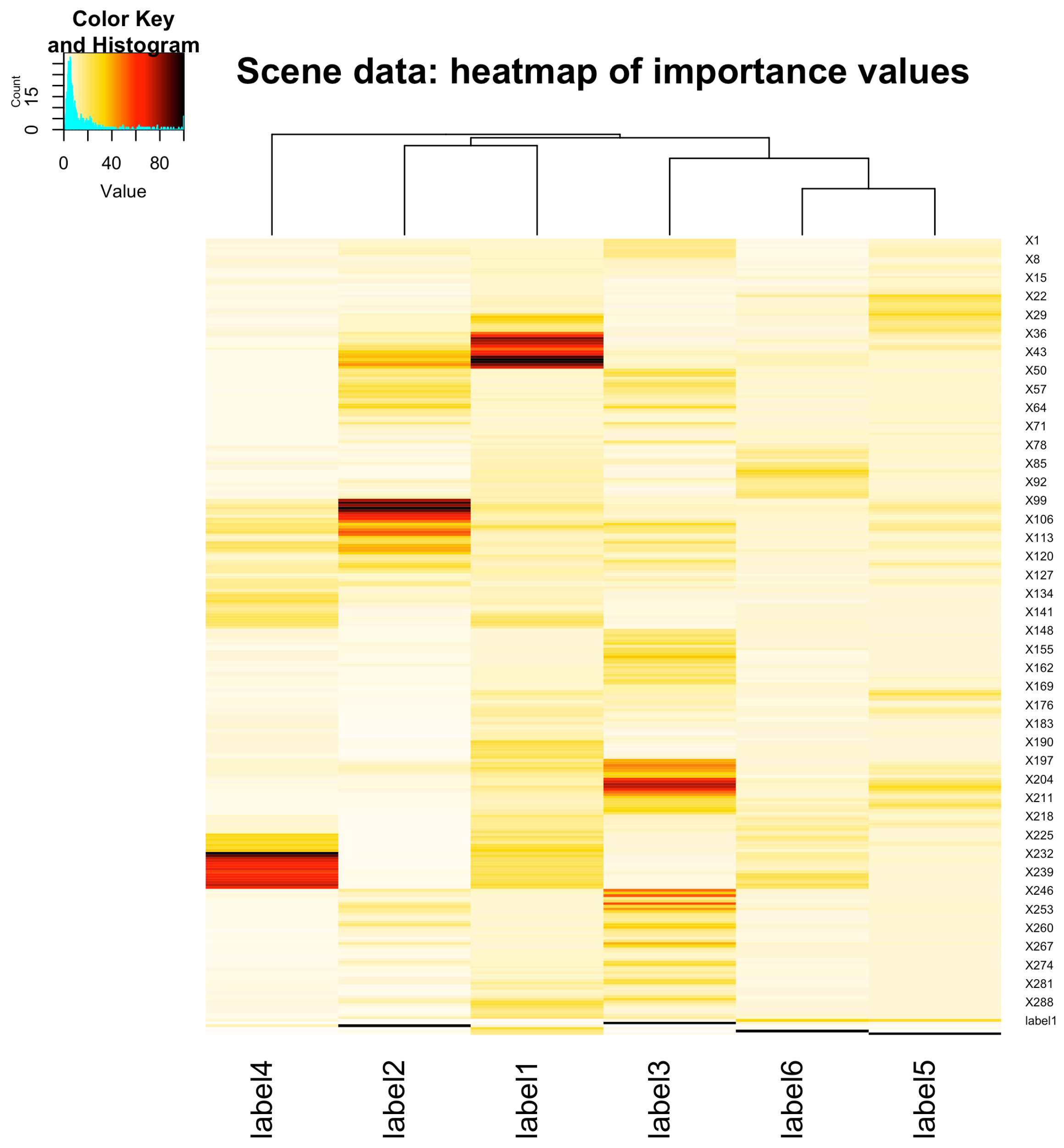


Figure 5.6: Scene dataset: heatmap of importance values for the LCC classifier.

The similarities of the labels in terms of their importance values are summarised in the dendrogram of Figure 5.6. Firstly, label 4 (*Field*) was dissimilar compared to all other labels. Secondly, labels 1 and 2 (*Beach*, *Sunset*) were clustered together, separate from the remaining labels. Lastly, the remaining labels were further split into label 3 (*Fall Foliage*) and labels 5 and 6 (*Mountain*, *Urban*). The heatmap showed the predictors used by the individual groups. For example, *Mountain* and *Urban* scenes were almost fully predicted from each other, likely because a *Mountain* scene image was almost never an *Urban* scene image simultaneously.

Stellenbosch University https://scholar.sun.ac.za

5.4.3 YEAST DATASET

The final benchmark analysis was performed on the yeast dataset, and the results are summarised in Figures 5.7, 5.8 and 5.9. Out of the three datasets analysed, the boxplots of the four classifiers fitted to the yeast data showed the largest differences in terms of the ML evaluation measures. The classifiers resulted in small differences for the HL values, and the CC-based classifiers outperformed BR in terms of the CL measure. The BR classifier obtained the best HL values, but did poorly in terms of CL accuracy. This is likely due to the presence of 14 labels. The HL measure is an average of separate losses for each label, and therefore the BR approach is suitable for minimising HL. On the other hand, in the CL measure all labels are considered simultaneously and the BR approach is not expected to do well.

The PR and RE displayed the inversely related relationship, where the two groups consisting of (BR, CC) and (ECC, LCC) differed by fairly large amounts. The harmonic mean of PR and RE, summarised in the F2 measure, favoured the ensemble methods. The AC measure looked very similar to F2, suggesting that the ensemble methods were better classifiers than BR and CC for this dataset. Each of the graphs in Figures 5.8a and 5.8b plots the variable importance values density functions for two labels at a time. The most important predictors for every label are indicated by arrows, and colour-coded for every pair of labels. It was interesting to note that the most important predictors were other labels for 13 out of the 14 labels. All of the densities are skewed to the right, with only a few significant predictors. Labels 12 and 13 were almost fully explained by only one predictor, which can also be seen in the heatmap of Figure 5.9.

The heatmap indicated that the label dependencies were indeed utilised by RF. Unlike in the previous datasets, the heatmap of the yeast data put the majority of emphasis on other labels acting as predictors (referred to in future as label predictors). Protein category 14 was unlike the rest of the labels. In this case, the RF model did not make an extensive use of the other labels, and used many of the protein features to predict label 14. It could be of interest to pay closer attention to this label in order to explore this deviation.

Stellenbosch University https://scholar.sun.ac.za

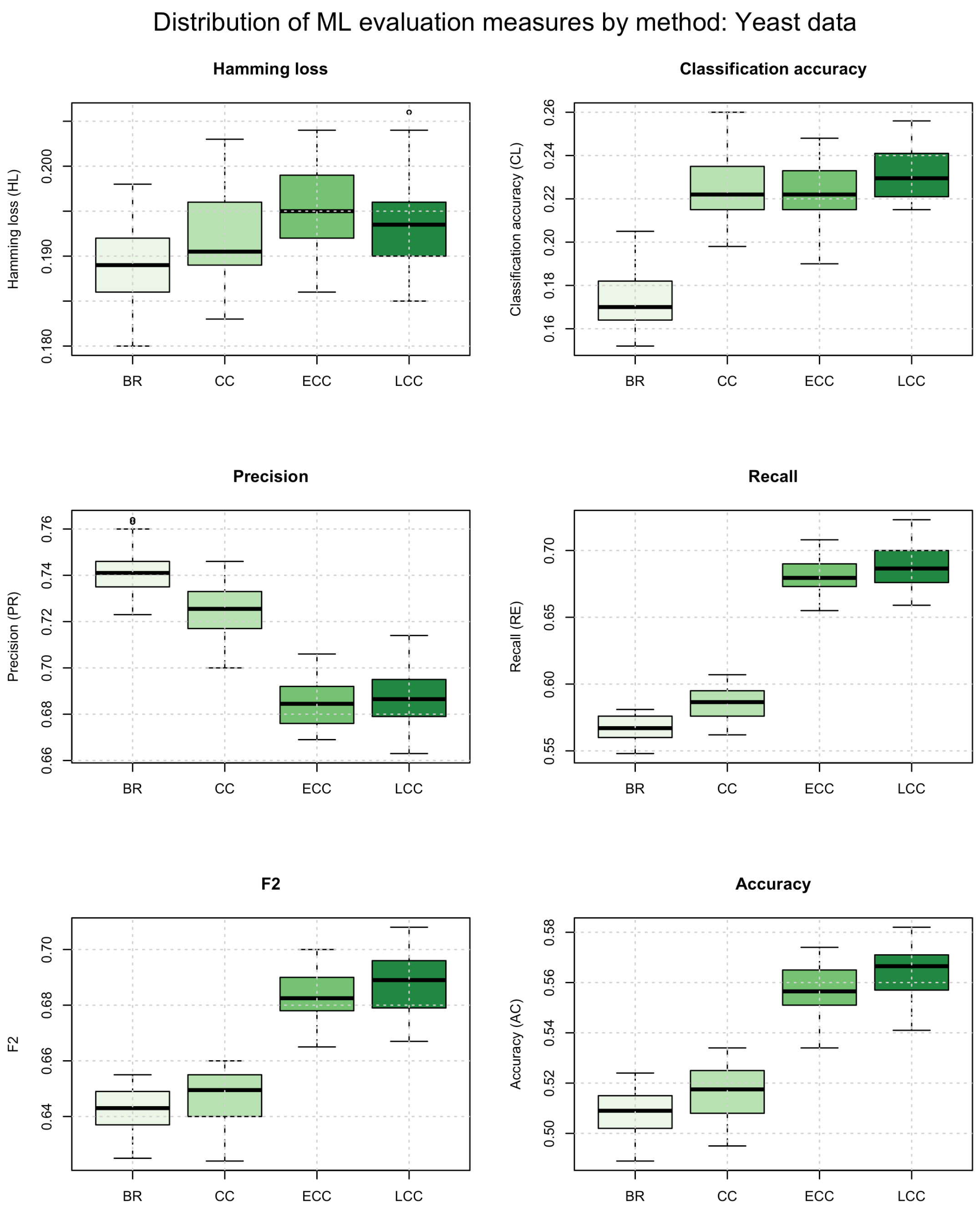


Figure 5.7: Yeast dataset: boxplots of ML evaluation measures for the four classifiers.

Stellenbosch University https://scholar.sun.ac.za

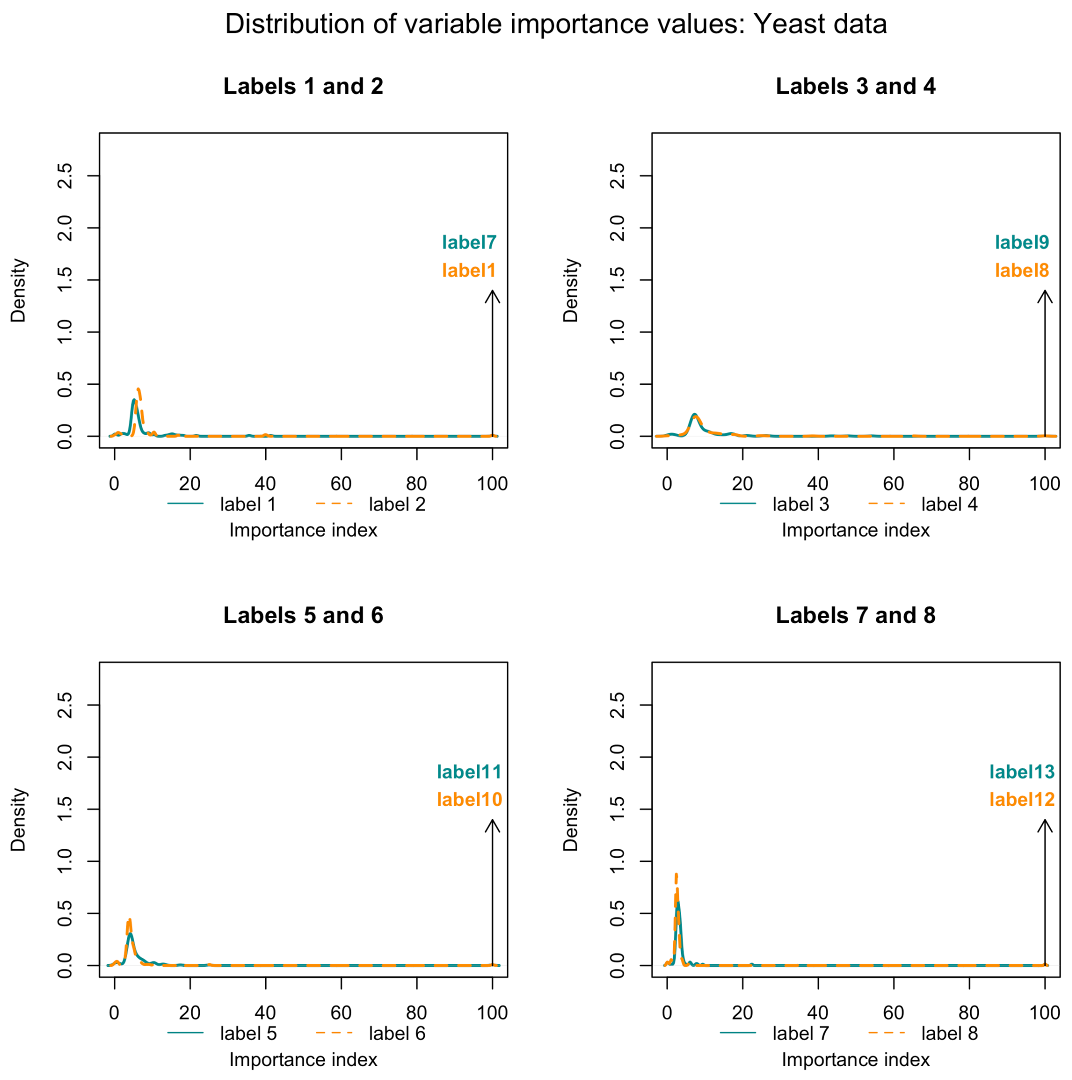


Figure 5.8a: Yeast dataset: distribution of importance values for the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

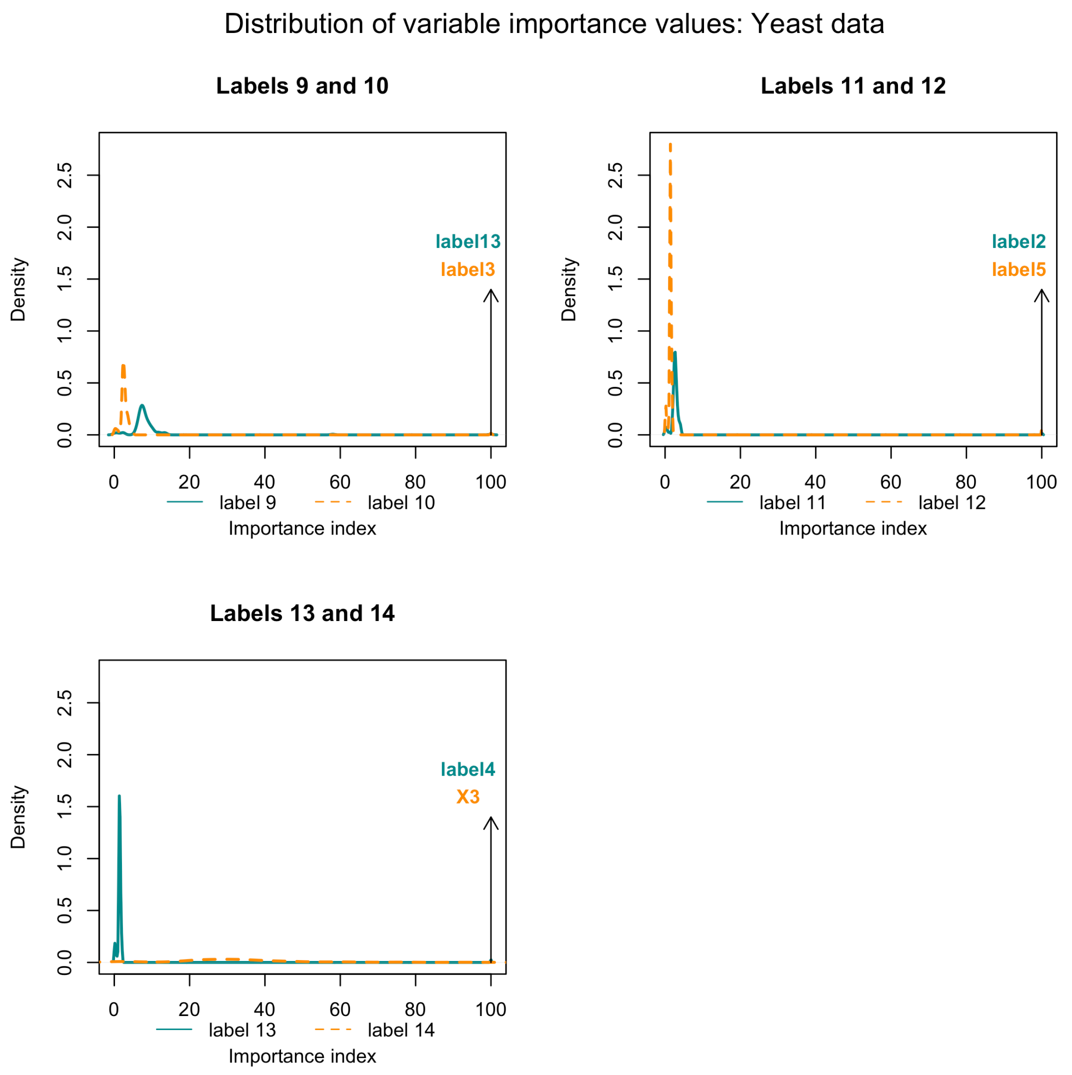


Figure 5.8b: Yeast dataset: distribution of importance values for the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

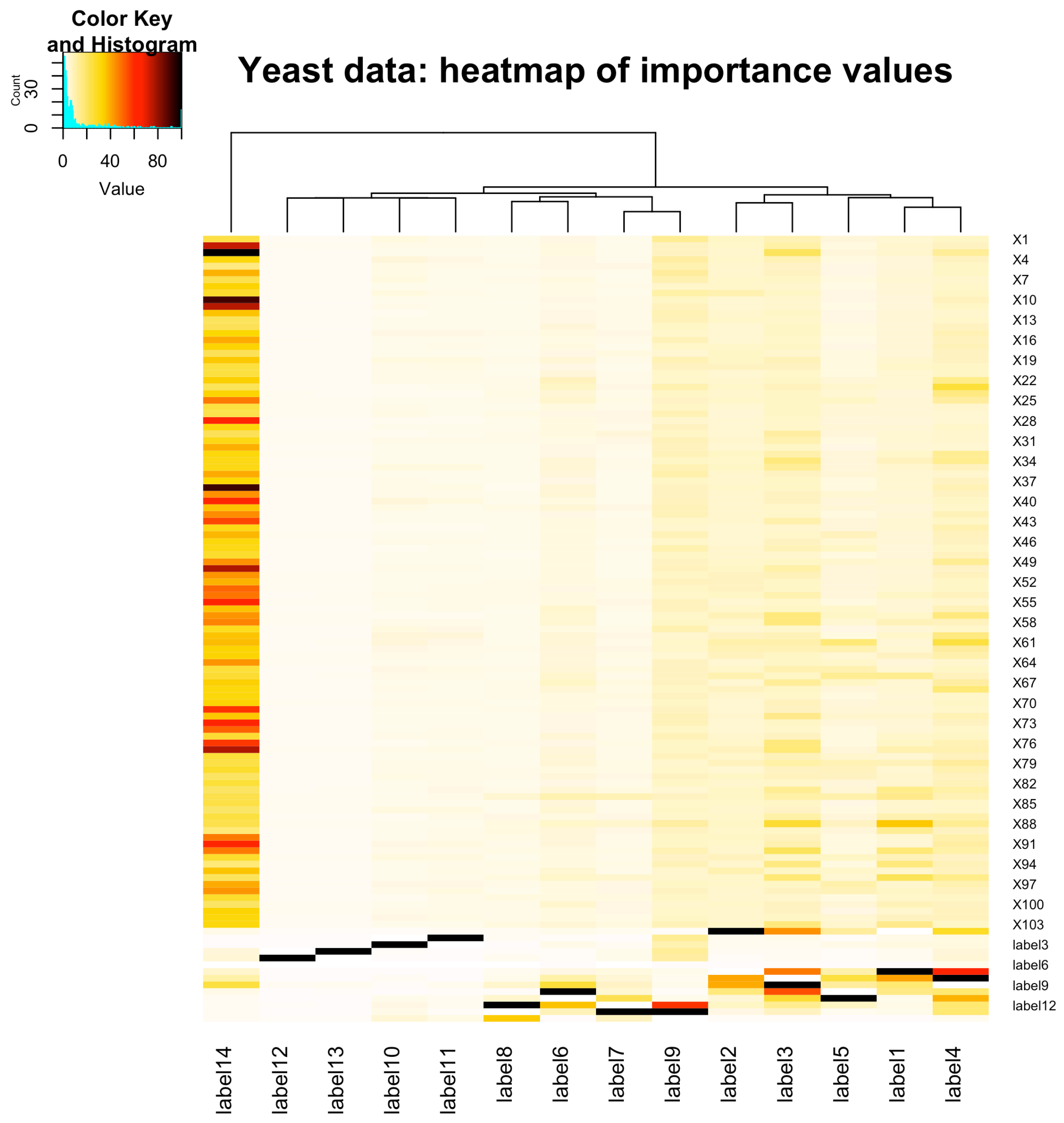


Figure 5.9: Yeast dataset: heatmap of importance values for the LCC classifier.

Figure 5.9 also displays the dendrogram of the labels in terms of their importance values. Protein category 14 differed from the rest, since it made use of most of the ***X*** predictors and almost none of the other labels. The remaining labels were further split into two big groups. The main difference between these two groups was in the use of the ***X*** predictors: the group on the left used predominantly other labels, whereas the group on the right was predicted by the ***X*** predictors as well, as shown in the darker yellow colour in the heatmap. There were other minor splits of the labels, resulting in a set of groups consisting of mostly two labels.

Stellenbosch University https://scholar.sun.ac.za

5.5 CONFIRMATORY ANALYSIS

In this section the four ML classifiers are compared based on the benchmark datasets, using Friedman's significance test as well as the Bonferroni-Dunn's and Holm's post-hoc comparison tests. These tests were done in order to evaluate the significance of the differences found in the exploratory analysis in Section 5.5.

5.5.1 FRIEDMAN'S TEST

Friedman's test (Friedman, 1937) provides a non-parametric alternative to the randomised complete block ANOVA test for dependent samples. It tests for equality of K treatments given N blocks. In the ML experiments, the treatments referred to the four classifiers, and the 18 blocks were the six evaluation measures for the three benchmark datasets analysed. The data values of interest were the mean evaluation measures based on the 30 runs. Note that the 18 blocks were not independent, as it is assumed in Friedman's test. Therefore, the results of the following tests were used only as an indication of the differences between the classifiers.

For Friedman's test, the hypothesis of interest is:

H_0: There are no significant differences amongst the four classifiers,

H_A: At least two of the four classifiers differ significantly.

For each of the evaluation measures, the four classifiers were ranked from 1 (best) to 4 (worst) and average ranks were assigned in case of ties. Let r_k^i represent the rank of the k^{th} classifier on the i^{th} measure, $k = 1, \dots, K = 4;\ i = 1, \dots, N = 18$. The mean ranks for every classifier were computed as $\bar{r}_k = \frac{1}{N}\sum_{i=1}^{N} r_k^i$. Friedman's test statistic and its distribution are given by:

$$\chi_F^2 = \frac{12N}{K(K+1)}\sum_{k=1}^{K}\left\{\bar{r}_k - \frac{1}{2}(K+1)\right\}^2, \qquad \chi_F^2 \sim \chi_{K-1}^2.$$

The results of Friedman's test for the benchmark datasets analysis are given in Table 5.2.

Stellenbosch University https://scholar.sun.ac.za

Table 5.2: Friedman's test for benchmark datasets analysis.

	BR	**CC**	**LCC**	**ECC**
$\bar{r}_k$	3.222	2.556	1.583	2.639
χ^2_F	14.950			
χ^2_{crit}	7.815			
p-value	0.00186			

In Table 5.2 the mean ranks of each classifier showed that on average, LCC obtained the best rank, followed by CC, ECC and BR. The Friedman's test p-value corresponding to the test statistic was 0.00186. Therefore, the null hypothesis was rejected at the 1% level of significance, and therefore there were significant differences between the classifiers for the benchmark datasets analysis. This result was based on a limited number of datasets and measures, but it indicated that the LCC classifier was definitely a competitive ML classifier, when compared to other established methods.

5.5.2 POST-HOC TESTS

Two post-hoc comparison tests were used following the rejection of the null hypothesis by Friedman's test. Of interest was to see whether the LCC classifier performed better than the other three classifiers. Therefore, LCC was set to be the control classifier, and three comparison tests were done for the hypotheses:

H_0: Rank of LCC = rank of classifier j,

H_A: Rank of LCC < rank of classifier j, for j = BR, CC, ECC.

The three hypothesis tests were denoted by H_1, H_2, H_3, and the corresponding p-values of the one-sided test, p_1, p_2, p_3, were computed from the test statistic:

Stellenbosch University https://scholar.sun.ac.za

$$z_j = \frac{(\bar{r}_{LCC} - \bar{r}_j)}{\sqrt{\frac{K(K+1)}{6N}}}, \qquad j = BR, CC, ECC, \qquad z_j \sim N(0,1).$$

Due to multiple testing, the family-wise error rate (the probability of falsely rejecting a true H_0 in at least one of the tests) was controlled by using the following two approaches. In the first approach, the Bonferroni-Dunn test adjusted the family-wise error rate to $\frac{\alpha}{(K-1)}$ (Dunn, 1961). The results of the test are given in Table 5.3.

Table 5.3: Bonferroni-Dunn's test for the benchmark datasets analysis.

$\alpha = 0.05$

i	***Hypothesis***	$z = \frac{\bar{r}_{LCC} - \bar{r}_j}{SE}$	***P-value***	$\frac{\alpha}{K-1}$	***Decision***
1	LCC vs. BR	-3.808	0.0001	0.017	Significant
2	LCC vs. ECC	-2.453	0.0071	0.017	Significant
3	LCC vs. CC	-2.259	0.0119	0.017	Significant

The Bonferroni-Dunn tests were conducted at 5% level of significance. The table includes the hypotheses tested, the corresponding test statistic, p-value, as well as the final decision about the hypotheses based on the adjusted value of α. At the 5% level of significance, the LCC classifier performed significantly better than BR, CC and ECC.

The second post-hoc test, known as Holm's test, used a step-up procedure that sequentially tested each of the hypotheses. After the p-values were computed and ordered so that $p_{(1)} \leq p_{(2)} \leq p_{(3)}$, the family-wise error rate was adjusted to be $\frac{\alpha}{K-i}$, i indicating the i^{th} hypothesis test considered. Holm's test then considered the smallest p-value. If it was smaller than the adjusted value of α, the null hypothesis was rejected, and the second smallest p-value was evaluated. This process continued until a hypothesis could not be rejected, after which there was not enough evidence to reject any of the following hypotheses (Holm, 1979:66-67). The results of Holm's test are given in Table 5.4.

Stellenbosch University https://scholar.sun.ac.za

Table 5.4: Holm's post-hoc test for benchmark datasets analysis.

$\boldsymbol{\alpha} = 0.05$

i	Hypothesis	$z = \frac{\bar{r}_{LCC} - \bar{r}_j}{SE}$	P-value	$\frac{\alpha}{(K-1)}$	Decision
1	LCC vs. BR	-3.808	0.0001	0.017	Significant
2	LCC vs. ECC	-2.453	0.0071	0.025	Significant
3	LCC vs. CC	-2.259	0.0119	0.050	Significant

According to Holm's test, LCC performed significantly better than all other classifiers at 5% level of significance. Holm's test has a higher power than the Bonferroni-Dunn test (Demšar, 2006:13), which could be seen in the results that corresponded to the two approaches. In this application, the power is the ability of a method to correctly identify significant improvements between LCC and other classifiers.

5.6 CONCLUSION

The exploratory as well as the confirmatory analysis in this chapter were carried out in order to explore the concepts of ML classification and VS applied to practical datasets. Using three benchmark datasets, the results of ML classification using the BR, CC, LCC and ECC classifiers were obtained.

In the exploratory analysis, the ML classification results were summarised using a boxplot representation of the six evaluation measures. Overall, the LCC classifier was comparable to, if not better than the others, especially in terms of the RE, F2 and AC measures. The variable importance values obtained from the LCC classifier were further analysed using the density and heatmap plots for every label. These graphs provided evidence for the importance of VS in high-dimensional ML classification, as implemented in RF models. Furthermore, it was observed that using other labels as predictors in ML classification was useful. Lastly, while both the LCC and ECC ensemble methods utilised label dependence and resulted in better performance than BR and CC, only the LCC classifier was constructed in such a way that the symmetry of the importance matrix could be used for interpretation.

Stellenbosch University https://scholar.sun.ac.za

In the confirmatory analysis, the six evaluation measures obtained from the classification of the three benchmark datasets were used as blocks in Friedman's test of treatment effects. The treatments corresponded to the four ML classifiers. Of interest was to determine whether there were any differences between the classifiers in terms of the measures. The results of the confirmatory tests were only indicative, since the evaluation measures were not independent. Using Friedman's test, the null hypothesis was rejected, therefore the classifiers were not equally accurate. The analysis proceeded with the Bonferroni-Dunn's and Holm's post-hoc tests to determine whether the LCC classifier performed better than the other three. At 5% level of significance, both of the tests found that LCC performed significantly better than the other classifiers.

Overall, the exploratory and confirmatory analyses of the four classifiers were useful in putting the proposed LCC classifier to a test, and in showing some of the useful outputs from this classifier. These results were however limited to using three datasets, and the assumption of independence in the significance tests. Therefore, it would not be appropriate to make any final claims about whether LCC is in general significantly better than the other classifiers. For example, the BR classifier is a very efficient classifier. It can provide good performance, especially in terms of the HL measure. Similarly, the CC classifier is equally important, since it makes use of label dependence while keeping the algorithm fast.

The LCC classifier offers competitive results to the other CC-based methods, and it allows for interpretation of the local importance values of the predictors. It is therefore not necessarily meant to replace all other classifiers, but rather to provide an alternative approach, which allows for understanding the ML datasets on a deeper level. While the benchmark datasets allowed for ML classification of real datasets, the true label dependence structure was unknown. In order to explore the ML datasets and the classification thereof in more detail, a simulation study was conducted, and it is summarised in the following chapter.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 6: Multi-label simulation analysis

6.1 INTRODUCTION

In the previous chapter, four classifiers were fitted to three benchmark datasets, and compared using six evaluation measures. While it was possible to determine which of the classifiers did overall better than others for a specific dataset, the results looked slightly different when another dataset was analysed. This finding is intuitively acceptable, since in general there is no single model that performs best for all datasets. There are many data-dependent factors that have an influence on how well a classifier predicts the outcome, including the number of predictors and the type of relationship between features and responses. For example, in ML classification the BR classifier is a relatively simple algorithm that has been outperformed by more complex models, such as the CC and ECC classifiers. However, given a ML dataset with a strong input signal and weak label dependence, the BR model may be the preferred choice for prediction, especially in high-dimensional problems.

In a simulation study the influence of the factors which impact on the performances of the different procedures can be investigated. This entails generating output data for all combinations of levels of the factors being studied. Analysing the generated data typically requires an investigation of main effects (where the focus is on a single factor) and possibly low-order interaction effects (involving two or maybe three factors simultaneously). Obviously these objectives cannot be achieved by using benchmark datasets.

6.2 SIMULATION ANALYSIS

Statistical simulation analysis involves using processes to generate datasets with a structure that resembles that of real-world datasets. Once the datasets have been simulated, a model can be fitted to the training set and evaluated on a test set. This process is repeated for numerous simulated datasets. Depending on the analysis, the factor of interest can for example be the number of predictors or the number of training data cases. Evaluating the performance of a specific classifier on simulated datasets provides an understanding of the main and low-order interaction effects.

Stellenbosch University https://scholar.sun.ac.za

Furthermore, should multiple models be fitted to the simulated datasets, it is possible to effectively compare the models in specific settings.

In the study described in this chapter, ML classification was performed using the BR, CC, ECC and LCC classifiers in 32 simulation settings, as described in the experimental design section. The results are summarised in exploratory and confirmatory analyses, where the outcomes of the classifiers are compared in different ML settings. In the exploratory part, the effect of five factors on the classifiers is analysed by using boxplots. This part also includes a heatmap analysis of the influence of the factors on the distribution of the variable importance values of LCC. Furthermore, the proportions of relevant variables that were correctly identified by the LCC are summarised. Lastly, in the confirmatory analysis Friedman's test is used to determine whether the performances of the four ML classifiers differ significantly. A description of the ML data simulation process, as well as the definition of the main factors are provided in the following section.

6.3 DATASETS

The simulation analysis involved the process of generating ML datasets according to the method proposed by Sandrock and Steel (2016). Specification of the following parameters was required in order to generate a ML dataset. Let the ML dataset consist of $\boldsymbol{X}_{mat}$, an $N \times P$ matrix of P input features, and $\boldsymbol{Y}_{mat}$, an $N \times L$ matrix consisting of L binary label responses. The values of the number of observations (N), number of variables (P) and the number of labels (L) need to be specified. The label densities can be specified by an L-component vector ***d***, where the density of label l is the proportion of data cases where label l is present, given by $LDens_l = \frac{1}{N}\sum_{i=1}^{N} Y_{il}, l = 1, \dots, L$. Vector ***d*** essentially determines how sparsely the labels are to be generated. Furthermore, parameter ρ is a correlation coefficient value that needs to be specified in order to control the underlying label dependence in the ML dataset.

The following parameters control the relationship between the predictors and labels. Let $\boldsymbol{A}_{mat}: P \times L$ be a matrix of true variable relevancies, where $A_{mat_{p,l}} = 1$ if variable p is relevant for label l, and $A_{mat_{p,l}} = 0$ if it is irrelevant, $p = 1, \dots, P;\ l = 1, \dots, L$. The matrix $\boldsymbol{A}_{mat}$ is computed randomly, based on specified values of $P(\boldsymbol{A}_{mat} = 1)$

Stellenbosch University https://scholar.sun.ac.za

and $P(\boldsymbol{A}_{mat} = 0)$. These probabilities must be specified beforehand. They are contained in the parameter $\boldsymbol{v}$. The final parameter, t, is used to distinguish between the locally relevant and irrelevant $\boldsymbol{X}$ predictors for the labels. The controlling parameters are summarised in Table 6.1, followed by a brief overview of the ML data generation process.

Table 6.1: ML dataset generation: controlling parameters.

N	Number of observations.
P	Number of input predictors.
L	Number of labels.
ρ	Label correlation coefficient.
t	Tuning parameter distinguishing the locally relevant and irrelevant predictors.
$\boldsymbol{d}$	L-component vector of label densities.
$\boldsymbol{v}$	Two-component vector consisting of $P\left(A_{mat_{p,l}} = 1\right)$ and $P\left(A_{mat_{p,l}} = 0\right)$.

The ML data generation process can be described in the following way. Let $[\boldsymbol{X}\ \boldsymbol{Y}]$, with $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices of dimensions $N\times P$ and $N\times L$ respectively, denote the dataset which has to be generated. Consider a given row, $[\boldsymbol{x}_i^T\ \boldsymbol{y}_i^T]$, of $[\boldsymbol{X}\ \boldsymbol{Y}]$, where $i \in \{1, 2, \dots, N\}$. In principle, two approaches can be followed to generate $[\boldsymbol{x}_i^T\ \boldsymbol{y}_i^T]$: generate $\boldsymbol{x}_i^T$ first, followed by $\boldsymbol{y}_i^T$ depending on $\boldsymbol{x}_i^T$, or generate $\boldsymbol{y}_i^T$ first, followed by $\boldsymbol{x}_i^T$ depending on $\boldsymbol{y}_i^T$. The latter of these two possibilities was used in this study.

Generating $\boldsymbol{y}_i^T$ requires generating values of the random vector $\boldsymbol{Y}: L\times 1 = [Y_1\ Y_2 \dots Y_L]^T$ having a multivariate Bernoulli distribution. Such a distribution can be specified in terms of the quantities $q_l = P(Y_l = 1)$, $l = 1, \dots, L$, and ρ, a parameter controlling the assumed common dependence amongst the components of $\boldsymbol{Y}$. Sandrock and Steel (2016) use an approach proposed by Oman (2009). This approach proceeds as follows.

Define $\theta_1, \dots, \theta_L$ as follows: $\theta_l = \phi^{-1}(q_l)$, $l = 1, \dots, L$, where $\phi(.)$ denotes the cumulative distribution function of the standard normal distribution. Then

Stellenbosch University https://scholar.sun.ac.za

$P(Z \leq \theta_l) = q_l$, $l = 1, \ldots, L$, if $Z \sim N(0;1)$. Also, let $u_1, \ldots, u_L$ be independent Bernoulli random variables with common success probability $\rho^{\frac{1}{2}}$. Put

$$w_l = u_l z_0 + (1 - u_l) z_l, \qquad l = 1, \ldots, L,$$

where z_0 and $z_1, \ldots, z_L$ are independent and identically distributed standard normal random variables. Finally, define $Y_l = Ind(w_l \leq \theta_l)$, so that Y_l is a binary random variable, $l = 1, \ldots, L$. It can then be shown that $P(Y_l = 1) = q_l$, as required by the definition of $q_l, l = 1, \ldots, L$. Also, for $l, k \in \{1, \ldots, L\}$, with $l \neq k$, it follows from conditioning on u_l and u_k that

$$P(Y_l = 1,\ Y_k = 1) = \rho P[z_0 \leq \min(\theta_l, \theta_k)] + (1 - \rho) q_l q_k,$$

and consequently $Covar(Y_l,\ Y_k) = \rho P[z_0 \leq \min(\theta_l, \theta_k)] - \rho q_l q_k$. The correlation between Y_l and Y_k is therefore given by

$$Corr(Y_l,\ Y_k) = \frac{\rho \min(q_l, q_k) - \rho q_l q_k}{\sqrt{q_l(1 - q_l) q_k (1 - q_k)}},$$

which equals ρ if $q_l = q_k$.

The approach described above was used in the simulation study to generate the required label vectors. Given a generated label vector $\boldsymbol{Y}$, the corresponding predictor vector $\boldsymbol{X}$ was generated as follows from a proposal by Sandrock and Steel (2016). It was assumed that $\boldsymbol{X}|\boldsymbol{Y}$ follows a P-variate normal distribution with mean vector $\boldsymbol{\mu}(\boldsymbol{Y})$ depending on $\boldsymbol{Y}$ and with covariance matrix Σ_x, independent of $\boldsymbol{Y}$. In the experimental work, the covariance matrix was taken to be $\boldsymbol{I}_P$. The j^{th} component of $\boldsymbol{\mu}(\boldsymbol{Y})$ was taken to be

$$\mu_j(\boldsymbol{Y}) = t \sum_{l=1}^{L} A_{mat_{jl}} Y_l, \qquad j = 1, \ldots, P,$$

with t a specified value. The rationale underlying this assumption is as follows. Recall that $A_{mat_{jl}} = 1$ if and only if variable X_j is (locally) relevant for label Y_l. Consequently, $\mu_j(\boldsymbol{Y})$ will increase by an amount t for every label Y_l which is present in $\boldsymbol{Y}$ (i.e. $Y_l = 1$), provided X_j is relevant for Y_l. The quantity t regulates the strength

Stellenbosch University https://scholar.sun.ac.za

of the 'signal': if t is large, the change in $\mu_j(\boldsymbol{Y})$ caused by changing Y_l from 0 to 1 will also be large, provided once again that X_j is relevant for Y_l. The simulated training dataset consisted of N_{train} observations $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{Ntrain}$ and the simulated test dataset was generated in the same way to give $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{Ntest}$. The parameters used, as well as the experimental settings in which the classifiers were fitted, are described in the following section.

6.4 EXPERIMENTAL DESIGN

The simulation analysis was performed in 32 scenarios, where for every design the classifiers were fitted to datasets generated using different combinations of parameters. The structure of the designs is summarised in Table 6.2. Of interest was to explore the effect of the following five parameters: the number of training observations (N_{train}), number of predictors (P), number of labels (L), label correlation (ρ), as well as the strength of the signal (t). The number of test observations (N_{test}) was set to 1000, the density vector ($\boldsymbol{d}$) was assigned a constant value of 0.3 for all labels, and the variable importance vector ($\boldsymbol{v}$) was set to give $P\left(A_{mat_{p,l}} = 1\right) = 0.2$ and $P\left(A_{mat_{p,l}} = 0\right) = 0.8,\ p = 1, \dots, P,\ l = 1, \dots, L$.

Table 6.2: Structure of the simulation design.

			$\rho = 0.1$		$\rho = 0.8$	
	P	***L***	$t = 0.5$	$t = 5$	$t = 0.5$	$t = 5$
$N_{train} = 200$	**50**	**5**	SIM_1	SIM_2	SIM_3	SIM_4
		20	SIM_5	SIM_6	SIM_7	SIM_8
	100	**5**	SIM_9	SIM_{10}	SIM_{11}	SIM_{12}
		20	SIM_{13}	SIM_{14}	SIM_{15}	SIM_{16}

			$\rho = 0.1$		$\rho = 0.8$	
	P	***L***	$t = 0.5$	$t = 5$	$t = 0.5$	$t = 5$
$N_{train} = 1000$	**50**	**5**	SIM_{17}	SIM_{18}	SIM_{19}	SIM_{20}
		20	SIM_{21}	SIM_{22}	SIM_{23}	SIM_{24}
	100	**5**	SIM_{25}	SIM_{26}	SIM_{27}	SIM_{28}
		20	SIM_{29}	SIM_{30}	SIM_{31}	SIM_{32}

Stellenbosch University https://scholar.sun.ac.za

The following steps were used for each of the 32 simulation settings. For every setting a matrix of underlying relevance values $\boldsymbol{A}_{mat}$ was randomly generated, according to the probabilities from $\boldsymbol{v}$. A set of training and test observations was then generated, according to the process described in Section 6.3, using the parameters for the simulation design of interest. Classifiers BR, CC, ECC ($M = L$) and LCC were fitted to the training data, using the RF base classifier of 50 trees. Gini index was used throughout the analysis. The test dataset was used for ML prediction, using the four classifiers. The six ML evaluation measures HL, CL, PR, RE, F2 and AC were computed from the predicted and true label sets. The $\boldsymbol{A}_{final}$ matrix of variable importance values from the LCC classifier was recorded.

The abovementioned steps were repeated 30 times for the specific simulation design, using the same $\boldsymbol{A}_{mat}$ for all runs. After completing all 30 repetitions, the mean and standard deviation of the evaluation measures were computed. Furthermore, the variable importance matrix $\boldsymbol{A}_{final}$ was averaged over the 30 simulation runs and stored in $\boldsymbol{A}_{global}$, in the same way as in the benchmark datasets analysis. The matrix $\boldsymbol{A}_{global}$ was standardised to represent index values in [0, 100], given in $\boldsymbol{A}_{global\_std}$. Lastly, for each of the labels the $\boldsymbol{X}$ predictors that corresponded to the 20% highest Gini index values were identified. The relevance information of the predictors was stored in matrix $\boldsymbol{A}_{LCC}: P \times L$, where $A_{LCC_{p,l}} = 1$ signified that variable p was relevant for label l, according to the LCC classifier, and $A_{LCC_{p,l}} = 0$ otherwise. For each of the simulation designs, the proportion of relevant variables correctly identified by LCC, was computed as:

$$VI_{prop} = \frac{1}{0.2PL} \sum_{l=1}^{L} \sum_{p=1}^{P} A_{mat_{p,l}} \times A_{LCC_{p,l}}.$$

Therefore, assuming that it was known that only 20% of all predictors were relevant, it was of interest to calculate the proportion of the most significant LCC predictors that were truly relevant. For example, a VI_{prop} of 0.8 meant that 80% of variables that were declared relevant by LCC were in fact relevant. These proportions were used in the exploratory analysis later in the chapter. Note that this analysis considered only the $\boldsymbol{X}$ predictors, since the relevance values for the label predictors were not

Stellenbosch University https://scholar.sun.ac.za

given in $\boldsymbol{A}_{mat}$. The R code corresponding to ML classification of the simulated datasets is given in Appendix C.1, and the following sections analyse the results from the simulation study.

6.5 EXPLORATORY ANALYSIS

The following exploratory tools were used to summarise the results from the simulation study. Firstly, boxplots of the distributions of ML evaluation measures were constructed for the four classifiers. Secondly, the variable importance indices from the LCC classifier were plotted using heatmaps. Furthermore, the proportions of relevant predictors correctly identified by LCC were summarised for all simulations. Lastly, a comparative analysis using Friedman's test was done to test for significant differences amongst the four classifiers in low and high label correlation settings. The R code used for the exploratory analysis is given in Appendix C.2.

6.5.1 BOXPLOTS

One of the advantages of the simulation study involved knowing the true parameters that generated the ML datasets. In the 32 simulation settings, the parameters of interest, as well as their corresponding values, were: N_{train} (200 vs. 1000), P (50 vs. 100), L (5 vs. 20), ρ (0.1 vs. 0.8) and t (0.5 vs. 5). Boxplots of the six evaluation measures described in Section 2.5 for all of the simulation settings are plotted for the LCC classifier in Figure 6.1. The columns represent different parameters for ρ and t, and the rows correspond to the parameters N_{train}, P and L, as summarised in Table 6.2.

One of the most prominent factors affecting the performance of LCC in Figure 6.1 was the t parameter. The evaluation measures in columns 2 and 4 represent the simulations that used the larger value of t. Comparing these results to the measures in columns 1 and 3, corresponding to the lower t value, it can be seen that the LCC classifier performed much better when t was higher, in terms of all the measures. A larger t value provided for an easier distinction between relevant and irrelevant predictors, so the classifier was expected to be better at utilising the relevant data and predicting labels accurately.

Stellenbosch University https://scholar.sun.ac.za

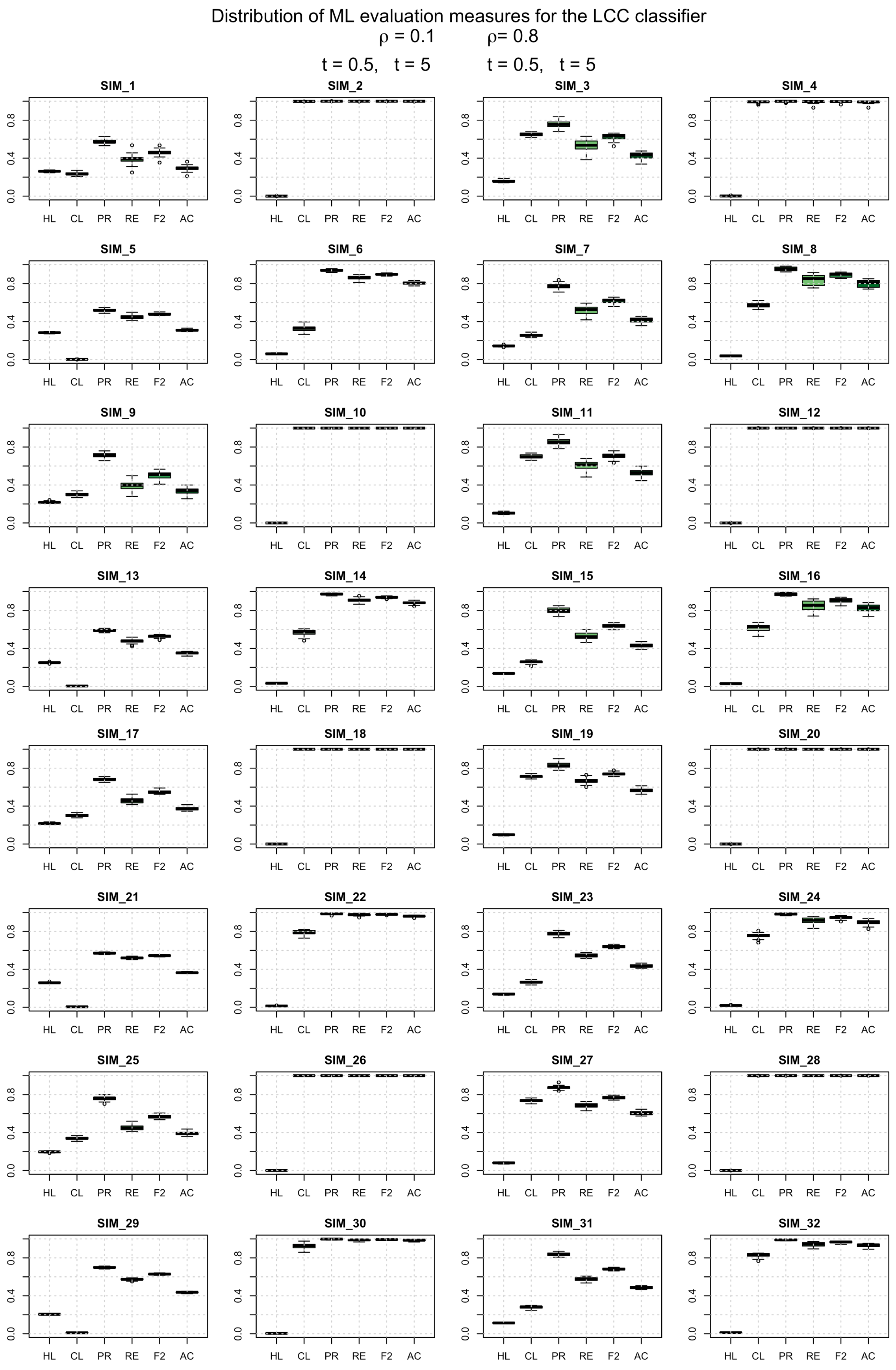


Figure 6.1: Distribution of ML evaluation measures for all simulation designs using the LCC classifier.

Stellenbosch University https://scholar.sun.ac.za

Furthermore, the first two columns were computed for ρ of 0.1, while the last two columns for ρ of 0.8. Generally, the LCC classifier did better for strongly correlated labels than when the label dependence was low. Figure 6.1 serves mainly as a reference graph for comparing all 32 simulations at the same time. A more detailed analysis of the influence of the various factors on the performance of the four classifiers follows below.

In each of the Figures 6.2, 6.3, 6.4, 6.5 and 6.6, the effect of one of the five parameters on the performances of the four classifiers is plotted using boxplots. In each of the figures, two simulation designs are compared, representing the two values of the parameter of interest. The effect of the parameter value was quantified using the six ML evaluation measures for every classifier. Of interest was to determine whether the different values of the simulation parameters had a positive, negative, or no effect on the classifiers. Moreover, the boxplots analysis also showed the relative performance of each of the classifiers within the specific parameter settings.

The boxplots compared two values of parameters N_{train} (200 vs. 1000), P (50 vs. 100), L (5 vs. 20), ρ (0.1 vs. 0.8) and t (0.5 vs. 5). A vertical line was placed between the two sets of boxplots in order to distinguish between the two parameter values. In the description of each of the figures, the simulation settings that were randomly selected and used for the comparison are given. These simulation numbers correspond to Table 6.2, which can be used for reference. All of the y-values of the boxplots were scaled to (0, 1) in order to compare the relative differences of the evaluation measures.

In Figure 6.2, the effect of low and high label correlation on the performance of the four classifiers is depicted. Overall, the ensemble methods did better than BR and CC for all but the PR measure. The differences between the classifiers were larger for ρ = 0.1 than for ρ = 0.8. Furthermore, all of the classifiers performed better in the high correlation scenario, including the BR classifier. The BR method should not have been affected by the higher label correlation, yet the results contradicted this claim.

Stellenbosch University https://scholar.sun.ac.za

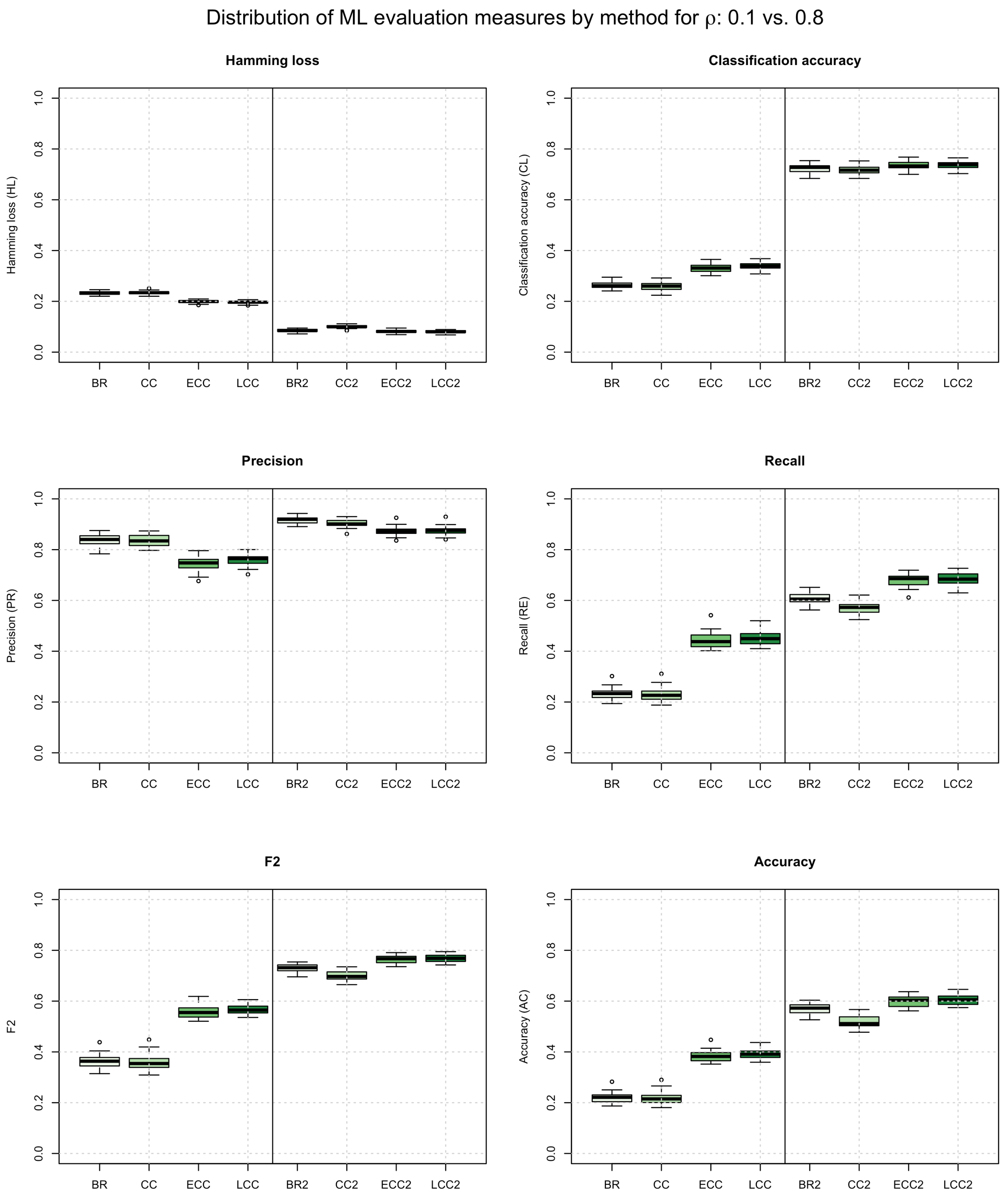


Figure 6.2: Distribution of ML evaluation measures for ρ: 0.1 vs. 0.8, (Simulation 25 vs, 27).

Stellenbosch University https://scholar.sun.ac.za

This thesis provides the first simulation work done using the ML data generation technique described above. The results obtained from this simulation work lead to a realisation that the correlation parameter has an effect on the strength of the input predictor signal. A closer look at this phenomenon revealed that the way in which the ML datasets were generated had an impact on the strength of the signal of predictors in high correlation designs. The signal of the inputs is not independent of the label correlation, but it strengthens as the label dependence increases. For more details on the theoretical explanation of this phenomenon, see Sandrock and Steel (2016).

Due to the interaction between the label dependence and signal strength, it was not possible to determine the direct impact of ρ on the classifiers. Ideally, given constant input signal, increasing the label dependence would lead to a larger utilisation of the label predictors by the CC-based methods, and potentially an improved performance. However, both an increase in ρ and a stronger input signal played a role in the classification process. Therefore, in order to explore the sole impact of the ρ parameter on the classifiers, the effect of label correlation on the signal strength would need to be adjusted for. This adjustment was not pursued further in this thesis.

In Figure 6.3, the effect of the training sample size on the four classifiers is observed. By comparing the differences between the classifiers for the two N_{train} values it can be seen that the ensemble methods performed better than BR and CC in terms of the RE, F2 and AC measures, but not PR. For HL and CL, the ensemble classifiers did slightly better than BR and CC, especially for the larger N_{train}.

Furthermore, the results showed an improved overall performance of the classifiers for 1000 training cases compared to the 200 cases, for all evaluation measures. This improvement was magnified for the ensemble methods, which on average improved by a larger margin than the BR and CC methods. The LCC classifier provided good results, and in some of the scenarios it performed the best out of all the classifiers.

The effect of the number of predictors on the accuracy of the classifiers is shown in Figure 6.4. Changing P from 50 to 100 did not seem to have a large impact on the classifiers. However, the classifiers did perform slightly better when the number of predictors was 100.

Stellenbosch University https://scholar.sun.ac.za

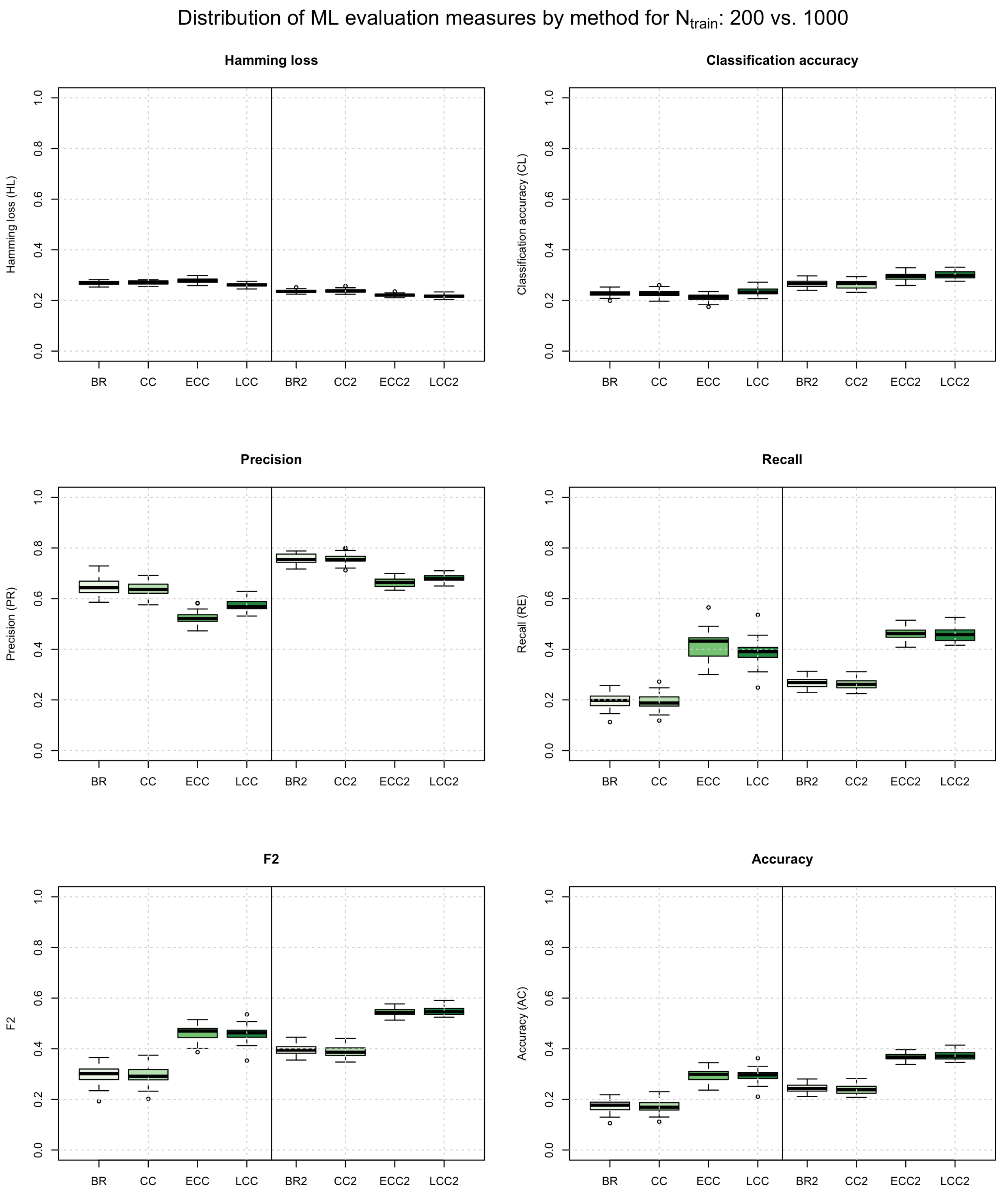


Figure 6.3: Distribution of ML evaluation measures for N_{train}: 200 vs. 1000, (Simulation 1 vs, 17).

Stellenbosch University https://scholar.sun.ac.za

In both scenarios, the proportion of relevant predictors was about 20%. Therefore, there were on average 10 relevant predictors when $P = 50$ and 20 relevant predictors for $P = 100$. Consider for example the BR classifier, where each of the RF models considered $\sqrt{P}$ predictors as candidates for tree splitting. In the case of 50 predictors, RF selected from $\sqrt{50} \cong 7$ variables, and when P was 100, the number of candidates was 10. On average, 20% of the number of candidates were relevant. Therefore, there were on average 1.4 and 2 relevant predictors out of the candidate variables, for P of 50 and 100 respectively. Seeing that this number was larger for P of 100, the BR classifier used more relevant predictors in this scenario and performed better.

In Figure 6.5, the effect of using 5 vs. 20 labels as responses is observed. Predicting a larger number of labels lead to worse performance in terms of all evaluation measures. The HL measure was affected the least by this change, and CL the most. This was to be expected, since HL treats the labels independently, whereas CL considers all labels simultaneously. Furthermore, the differences between the four classifiers were mostly larger when 20 labels were predicted compared to using 5 labels. The ensemble methods outperformed the BR and CC classifiers for all but the PR measure.

Figure 6.6 depicts the effect of the t parameter on the performance of the four classifiers. In this figure, simulations 13 and 14 are compared, which correspond to simulation designs with label correlation of 0.1. A larger value of t provided a stronger input signal to the classifiers. Therefore, all of the classifiers focused mostly on the input variables. Overall, the ensemble methods performed better than BR and CC in both designs. Increasing the t parameter had a large positive impact on the classifiers in terms of all evaluation measures. Furthermore, the differences between the classifiers were much smaller for a higher value of t. Therefore, while the performance of the ensemble classifiers was still better when t was higher, the single-chain models managed to utilise the input signal efficiently and be more competitive compared to LCC and ECC.

Stellenbosch University https://scholar.sun.ac.za

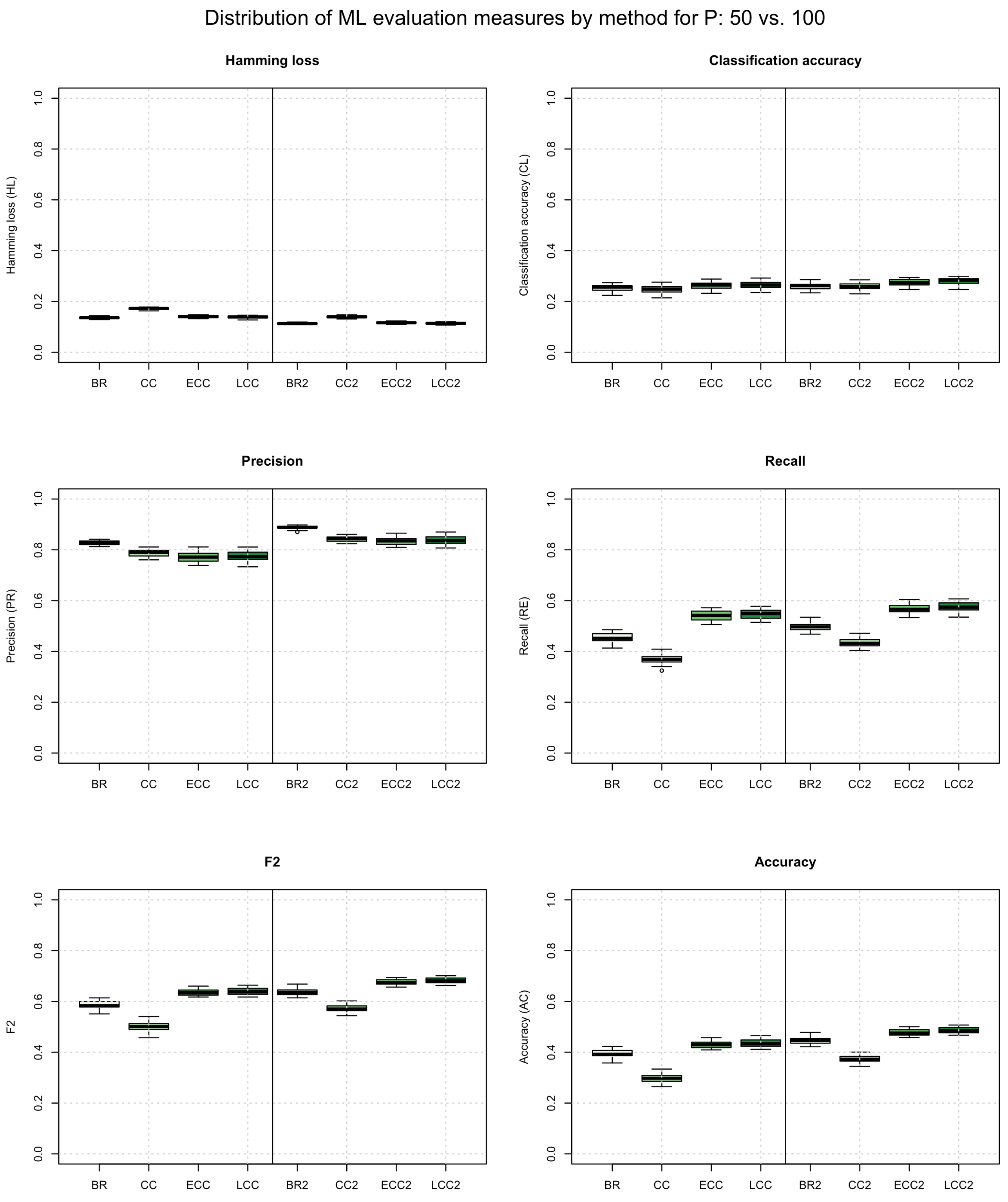


Figure 6.4: Distribution of ML evaluation measures for P: 50 vs. 100, (Simulation 23 vs, 31).

Stellenbosch University https://scholar.sun.ac.za

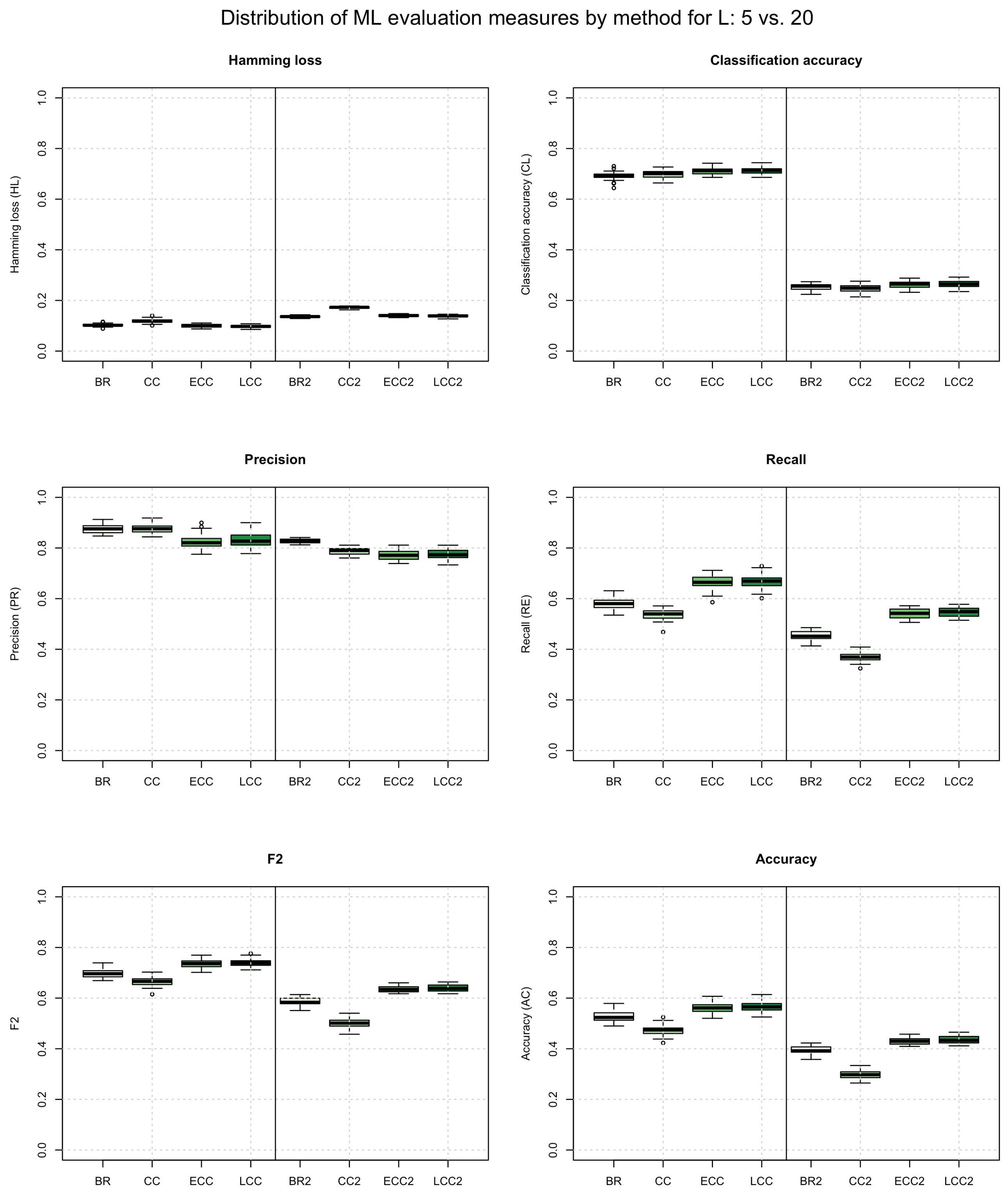


Figure 6.5: Distribution of ML evaluation measures for L: 5 vs. 20, (Simulation 19 vs, 23).

Stellenbosch University https://scholar.sun.ac.za

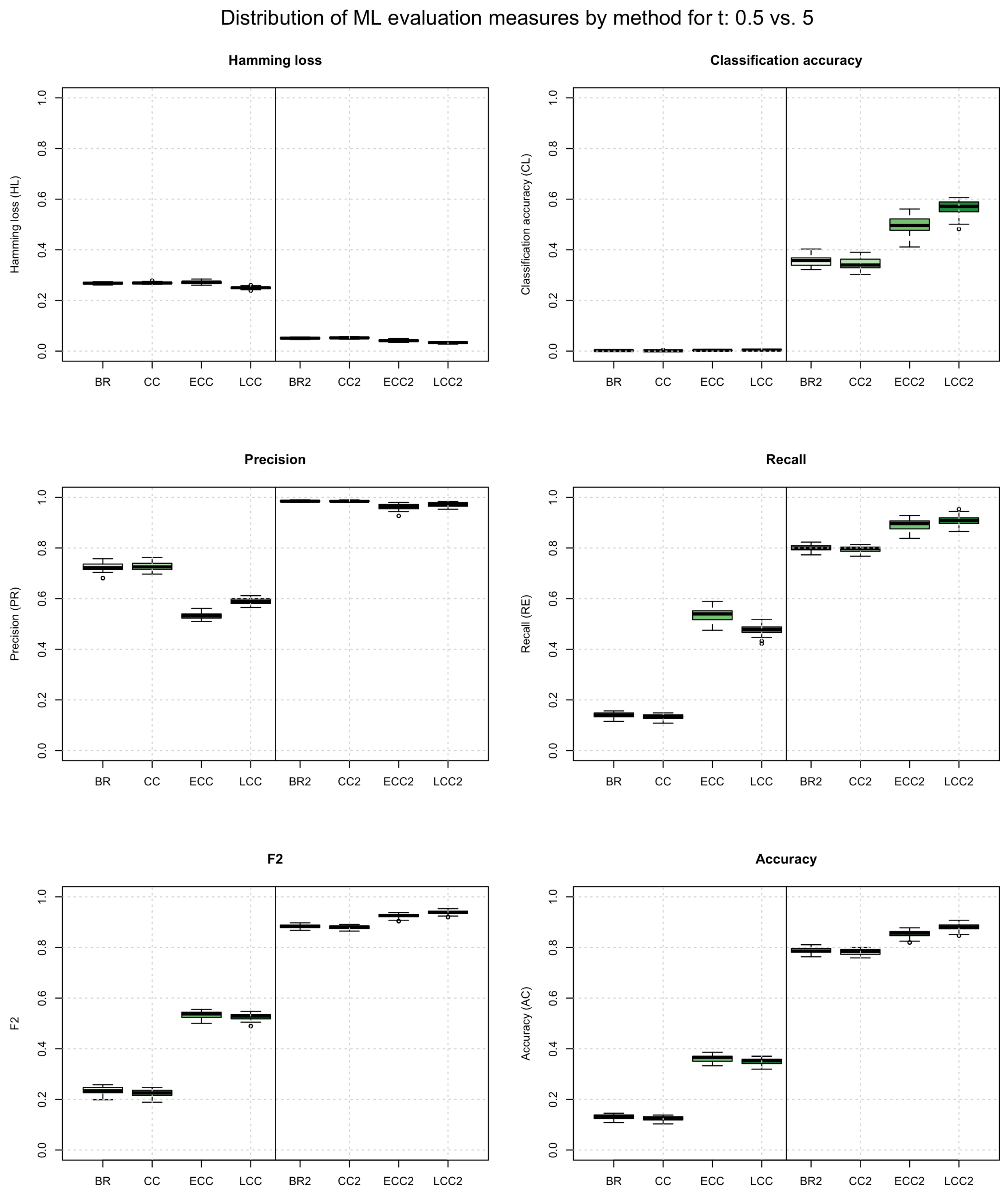


Figure 6.6: Distribution of ML evaluation measures for t: 0.5 vs. 5, (Simulation 13 vs, 14).

Stellenbosch University https://scholar.sun.ac.za

6.5.2 L-CLASSIFIER CHAINS: HEATMAPS

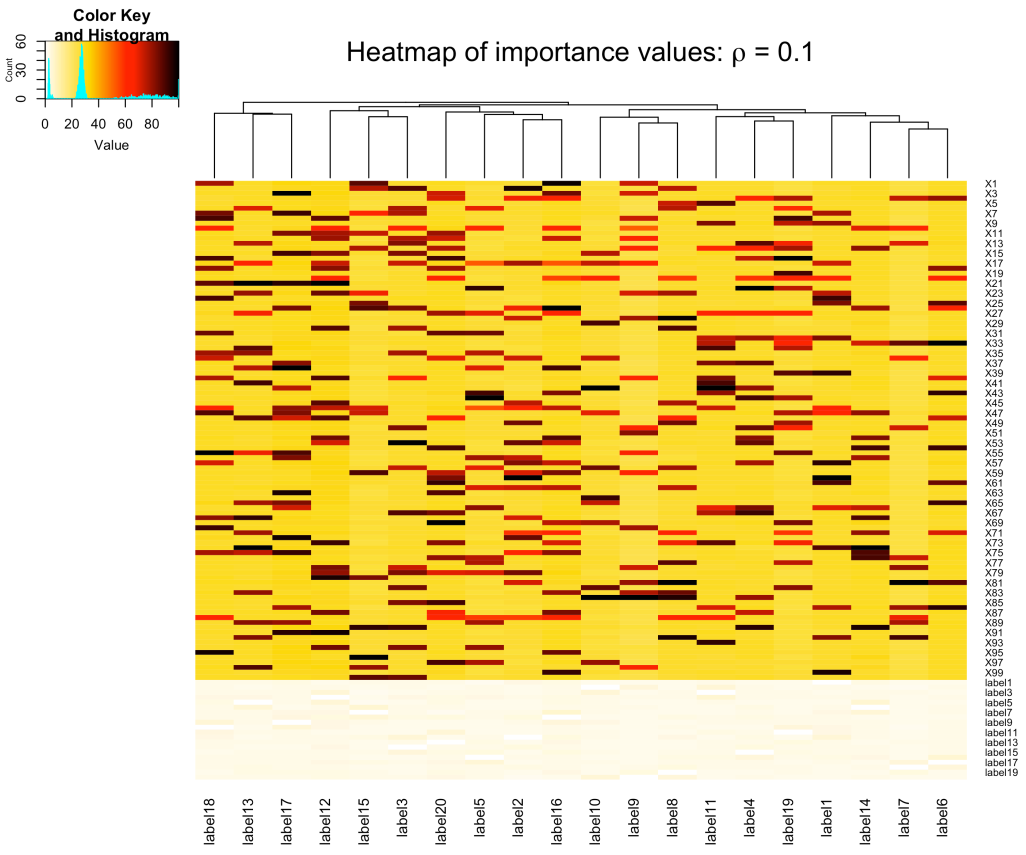
In this section of the exploratory analysis, the effect of the five simulation parameters on the variable importance values obtained from the LCC classifier is analysed using heatmap plots. For every parameter, two simulation settings were chosen randomly, each of which used a different value of the parameter of interest. The other four simulation parameters were kept constant for this analysis. The simulation numbers are shown under the description of each of the figures, and refer to the settings from Table 6.2. Each of the heatmaps includes a colour key and a histogram plot of the variable importance values. Predictors that were considered to be significant by the LCC were assigned a darker cell representation on the heatmap.

The heatmaps plotted in Figure 6.7 represent simulation designs of low and high label correlation, using Simulations 29 and 31. These simulations used 1000 training observations to train 20 labels and 100 predictors, with a t value of 0.5. In the low correlation heatmap, the LCC classifier almost completely disregarded other label information and focused on the input variables. The signal of the features was relatively low, so the LCC classifier was not expected to identify the relevant predictors with high confidence. Indeed, LCC assigned a medium to high importance to all of the inputs, and did not manage to identify the 80% irrelevant variables.

A very different picture is depicted in the heatmap of the high correlation scenario. The LCC classifier made extensive use of the label information, since many of the label predictors had a high importance value. There were on average about two highly relevant label predictors for each label. Moreover, the signal provided by the input variables was considered to a lesser degree, with only a few inputs of medium importance values.

Overall, the variable importance results for the different label correlation values are consistent with the theoretical expectations, and the LCC classifier used the label information effectively when ρ was 0.8. The extent to which the presence of a higher label correlation affected the performance of LCC can however not be fully known, due to the increased input signal strength in the highly correlated case, as explained previously.

Stellenbosch University https://scholar.sun.ac.za



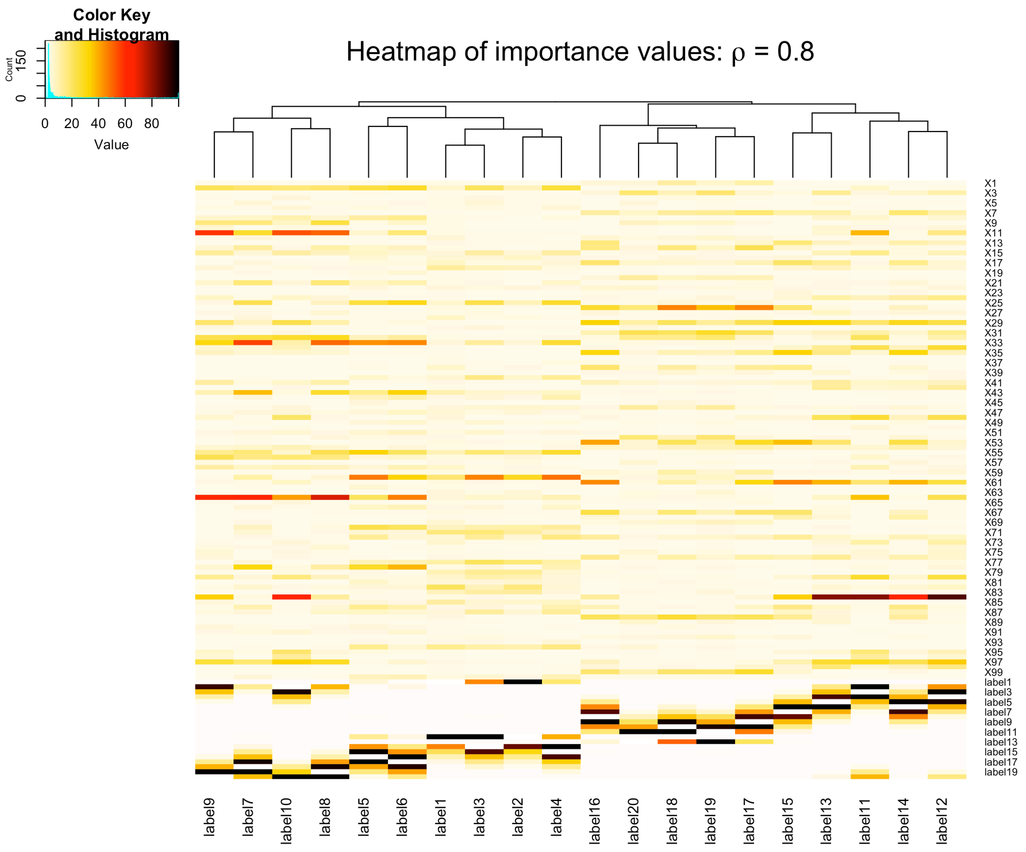


Figure 6.7: Heatmap plots of variable importance values for ρ: 0.1 vs. 0.8, (Simulation 29 vs, 31).

Stellenbosch University https://scholar.sun.ac.za

The effect of increased training sample size on the variable importance values is displayed in the heatmaps of Figure 6.8. These plots summarise Simulations 11 and 27, where the LCC classifier made use of 100 input predictors to predict five labels. The simulations corresponded to ρ of 0.8, and a t value of 0.5.

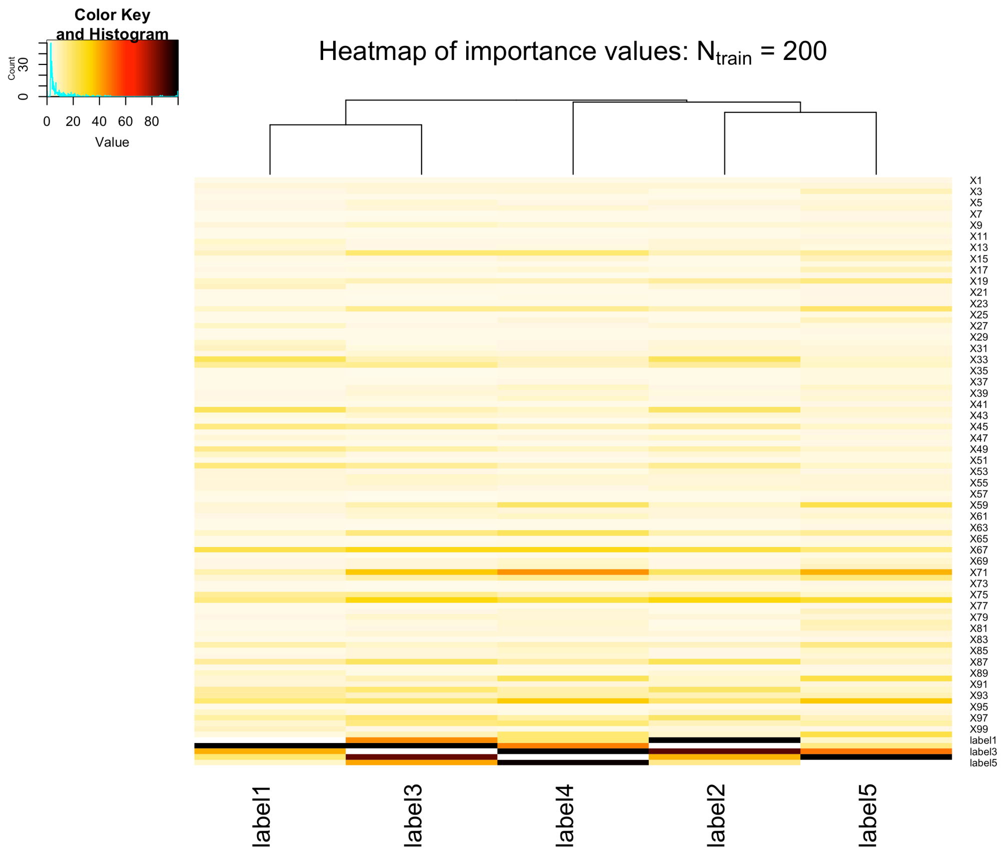
The first heatmap refers to the scenario of using 200 training observations to fit the model. The classifier considered the label predictors to be highly important, due to the high label correlation. Other input predictors received mostly low to medium importance values. In the second heatmap, 1000 training observations were used for model fitting. The label predictors remained very relevant with high importance values. In this case, the LCC classifier used fewer input features for prediction, since the histogram of the importance values was more positively skewed than for N_{train} of 200. Overall, the colour of the heatmap indicates a sparser distribution of the input importance values.

The larger number of training cases likely increased the ability of the classifier to differentiate between the relevant and irrelevant inputs more effectively. Overall, this finding is consistent with the general notion that when more training observations are available, the trends present in the data are easier to detect and predict, all other things kept constant.

Figure 6.9 illustrates the effect of the number of variables on the importance values of LCC. The simulation settings chosen, 8 and 16, refer to the high correlation scenario, where 200 training observations and a t value of 5 were used to predict 20 labels.

The distributions of both importance heatmaps are skewed to the right, with only a few of the predictors having large importance values. Interestingly, the label predictors did not have a large impact on the model fit, compared to the input predictors. This phenomenon could be attributed to the large value of t, which caused the strength of the signal to be high. Due to the large signal, the LCC classifier managed to successfully identify the few significant predictors, marked by dark colour in the heatmaps.

Stellenbosch University https://scholar.sun.ac.za



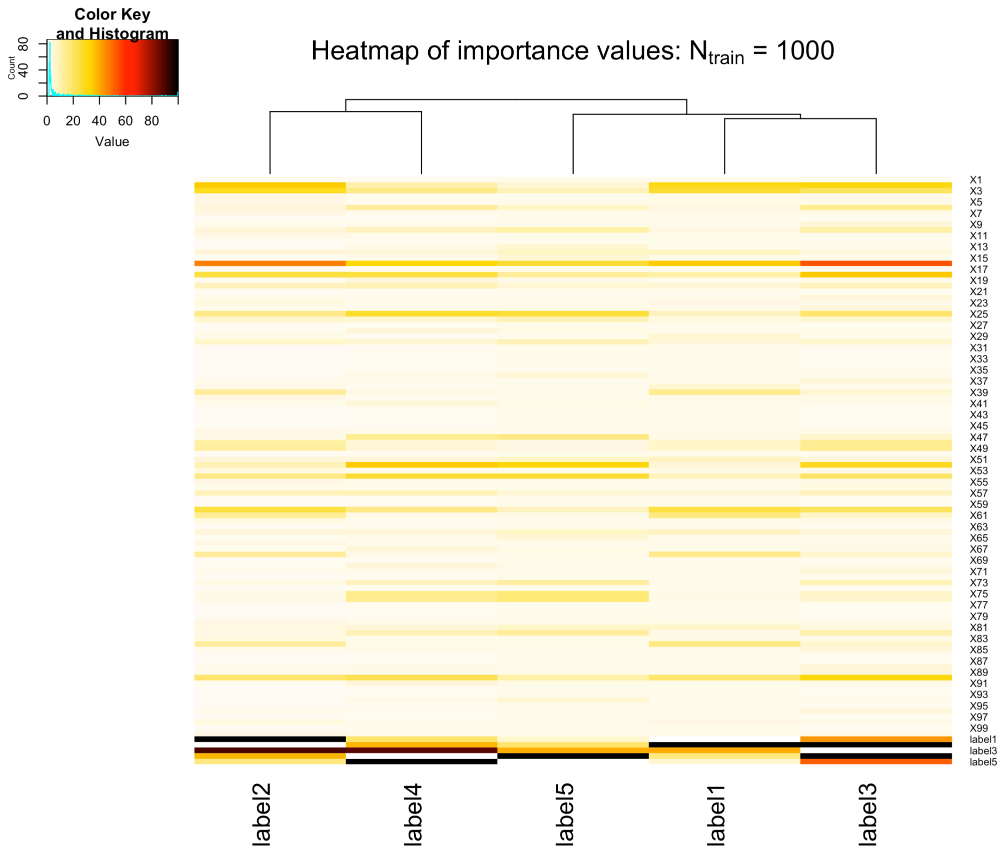
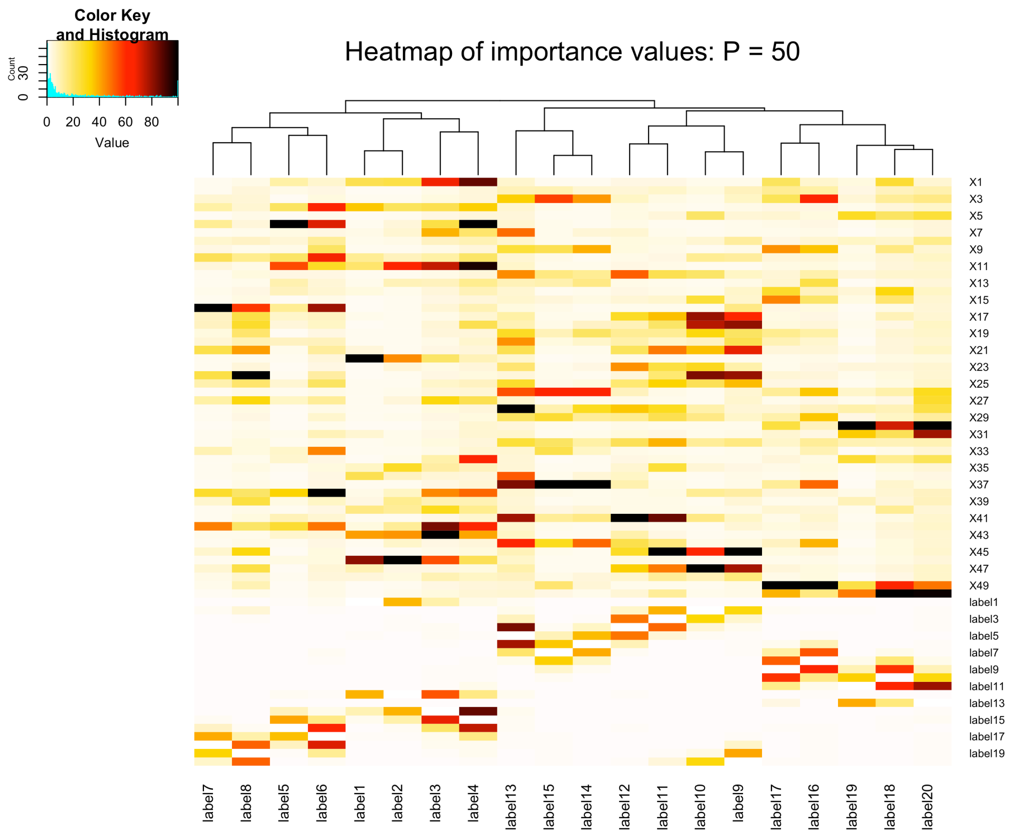


Figure 6.8: Heatmap plots of variable importance values for N_{train}: 200 vs. 1000, (Simulation 11 vs, 27).

Stellenbosch University https://scholar.sun.ac.za



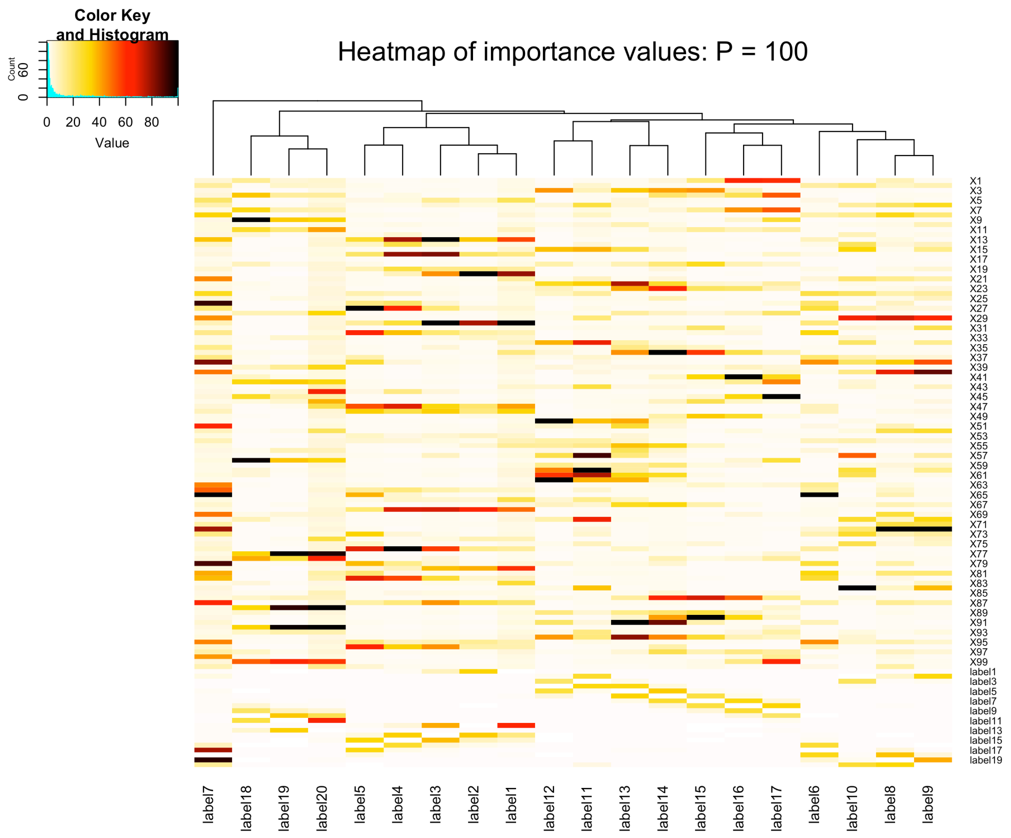


Figure 6.9: Heatmap plots of variable importance values for P: 50 vs. 100, (Simulation 8 vs, 16).

Stellenbosch University https://scholar.sun.ac.za

Comparing the heatmap plots, two interesting trends are observed. Firstly, the label predictors were used to a lesser extent when P was higher. This was likely as a result of having on average more relevant $\boldsymbol{X}$ predictors available for tree splitting when a larger P value was used (as explained in Section 6.5.1). Secondly, the second heatmap reflects that overall the LCC classifier made use of fewer predictors when P increased, as indicated by a larger proportion of insignificant predictors and a more positively skewed variable importance distribution. Therefore, as P increased, the LCC classifier was more successful at identifying the relevant predictors.

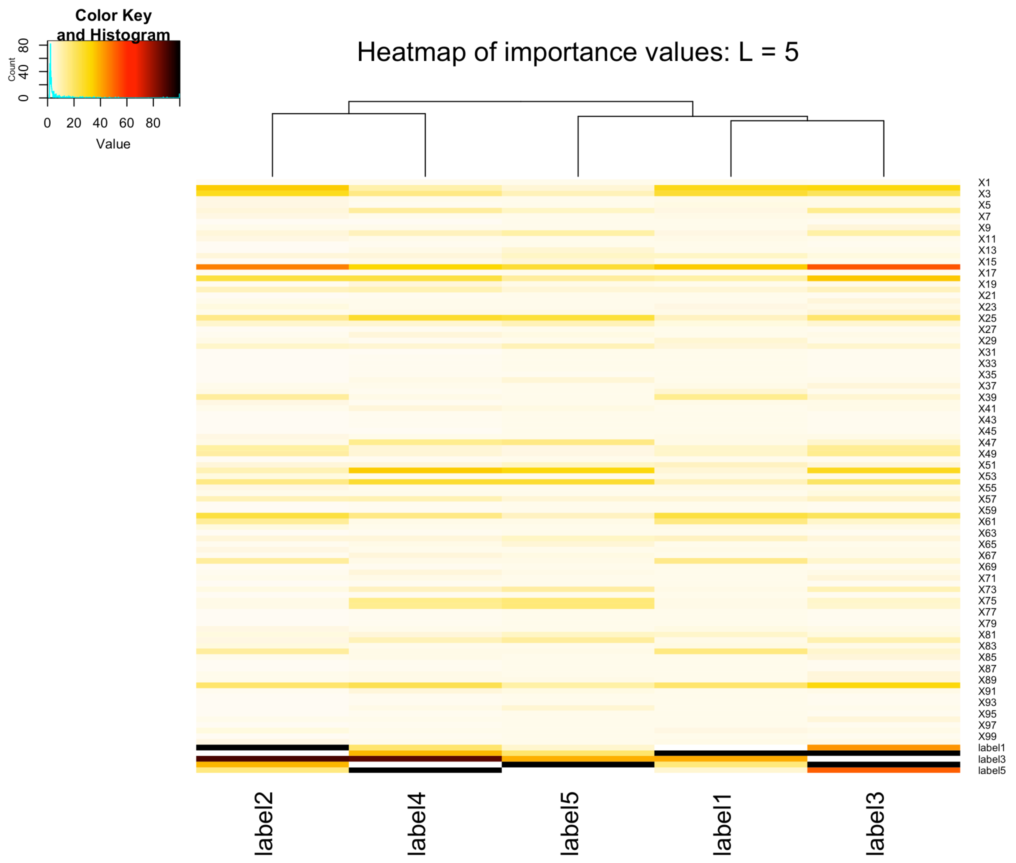
In Figure 6.10, the variable importance distributions are summarised for Simulations 27 and 31. The settings used 100 input predictors, t of 0.5 and 1000 training observations to predict labels with 0.8 label correlation. In the first heatmap, the distribution of the importance values when five labels were predicted is shown. Due to the relatively low input signal and high label dependence, the classifier used mostly the label predictors, and placed low importance on the input predictors.

In the second heatmap the distribution of importance values when 20 labels were predicted is illustrated. These values resembled the values in the first graph. One or two highly significant label predictors were present for each label, and the input variables had mostly low significance. However, there were a few significant input predictors for most of the labels as well, as indicated by the red coloured cells and the slightly less positively skewed distribution when $L = 20$.

The final set of heatmap plots is displayed in Figure 6.11, comparing Simulations 19 and 20. In this case, the effect of the parameter t on the importance value distribution was of interest. The simulation designs used 1000 observations of 50 input predictors to predict five labels with dependence of 0.8. These two graphs clearly depict the effect of using t values of 0.5 and 5.

With t of 0.5 a relatively lower input signal was present. The classifier made extensive use of the highly correlated label predictors, without much regard for the input predictors. On the other hand, the stronger input signal that corresponded to using a t value of five overshadowed the importance of the highly correlated label predictors. Consequently, the classifier could identify precisely which input features were relevant, and placed low importance on the label predictors.

Stellenbosch University https://scholar.sun.ac.za



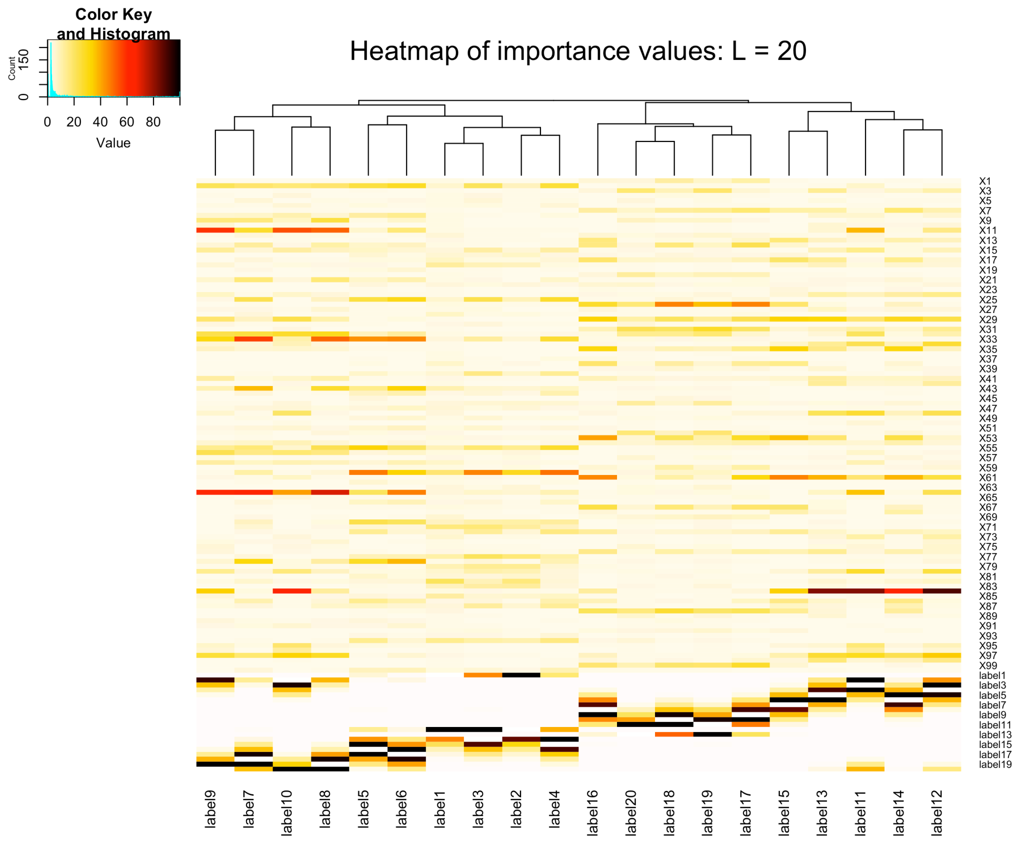
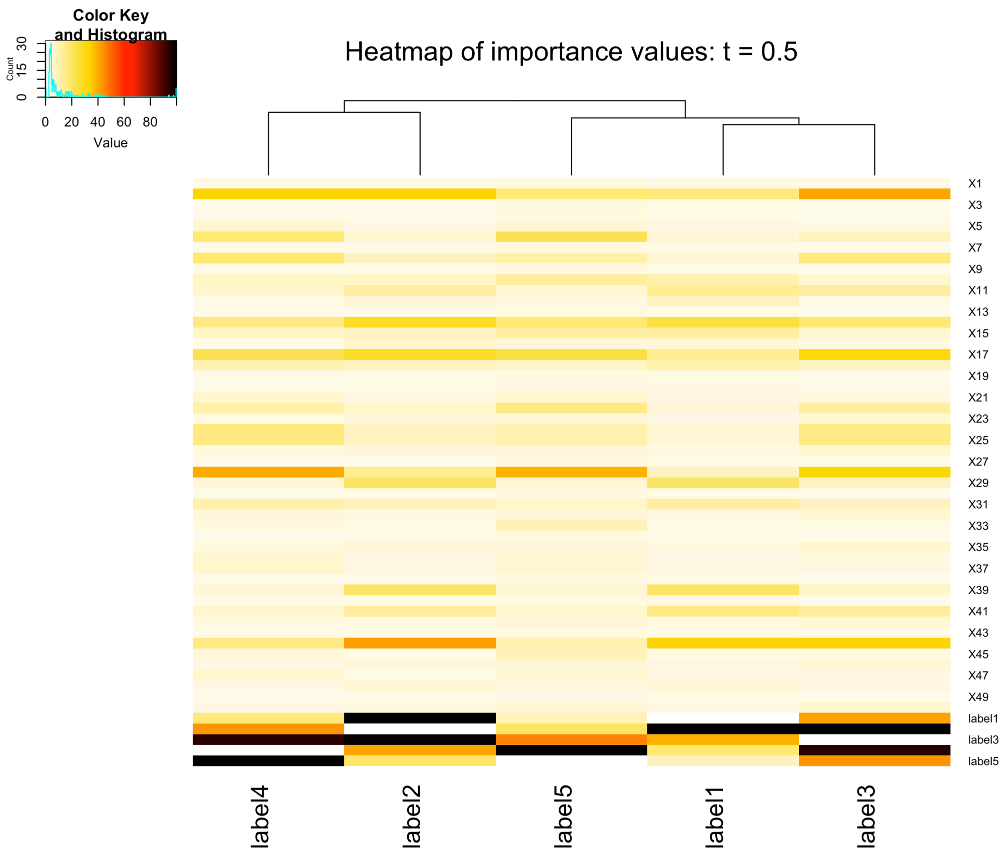


Figure 6.10: Heatmap plots of variable importance values for L: 5 vs. 20, (Simulation 27 vs, 31).

Stellenbosch University https://scholar.sun.ac.za



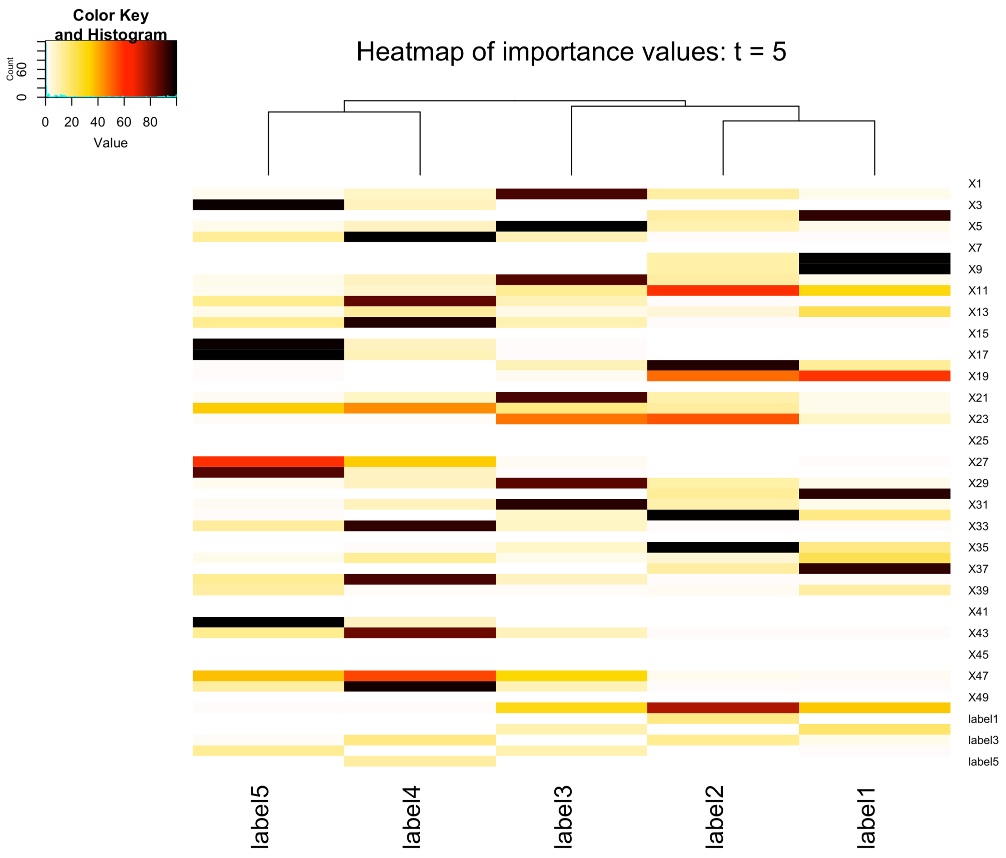


Figure 6.11: Heatmap plots of variable importance values for t: 0.5 vs. 5, (Simulation 19 vs, 20).

Stellenbosch University https://scholar.sun.ac.za

In all of the heatmap figures, the distributions of the variable importance values for the labels fitted by the LCC classifier are illustrated. It is fair to question whether the predictors that corresponded to the most significant importance values were truly relevant, according to the simulated dataset generated. In the following section, this concept is explored in more detail.

6.5.3 L-CLASSIFIER CHAINS: IDENTIFYING RELEVANT VARIABLES

In the previous analysis the distributions of the variable importance values in various scenarios were compared. These sets of importance indices described the relationships between every label and predictor combination. The heatmap plots indicated which of the predictors were considered relevant, according to the LCC model. The following analysis was done in order to quantify how accurately the LCC classifier identified the truly relevant predictors.

The proportion of the most significant LCC predictors that were truly relevant (VI_{prop}) was computed, as described in Section 6.3. For every label, the LCC classifier assigned a Gini importance value to all variables, including the other label predictors, and stored them in $\boldsymbol{A}_{final}$. Section 3.3.3 described this process in more detail. The Gini values reflected the extent to which the LCC classifier made use of the corresponding predictors. However, no threshold was set in order to determine which of the Gini values were high enough for the variables to be considered relevant. For the purpose of this analysis, it was assumed that the LCC classifier identified 20% of the $\boldsymbol{X}$ predictors with the highest Gini values as relevant. For every label, the relevant predictors were assigned a value 1 and irrelevant predictors a value 0 in $\boldsymbol{A}_{LCC}$, as described in Section 6.3.

The computation of VI_{prop} involved comparing information from $\boldsymbol{A}_{LCC}$ and $\boldsymbol{A}_{mat}$, and finding the proportion of predictors that were relevant, according to both of these matrices. Note that VI_{prop} considered only the $\boldsymbol{X}$ variables, disregarding the label predictors. This assumption was made in order to proceed with the analysis, since the true relevance information of other label predictors was not part of $\boldsymbol{A}_{mat}$.

Stellenbosch University https://scholar.sun.ac.za

The resulting values of VI_{prop} are summarised in Table 6.3 for all simulation designs.

Table 6.3: The proportion of correctly identified relevant variables by the LCC, VI_{prop}.

			ρ = 0.1		***ρ = 0.8***		
	P	***L***	***t = 0.5***	***t = 5***	***t = 0.5***	***t = 5***	***Mean***
N_{train} = *200*	**50**	**5**	0.800	0.880	0.640	0.760	0.770
		20	0.940	0.895	0.670	0.575	0.770
	100	**5**	0.980	1.000	0.620	0.860	0.865
		20	0.910	0.942	0.582	0.588	0.756
Mean			0.908	0.929	0.628	0.696	

			ρ = 0.1		***ρ = 0.8***		
	P	***L***	***t = 0.5***	***t = 5***	***t = 0.5***	***t = 5***	***Mean***
N_{train} = *1000*	**50**	**5**	0.920	0.940	0.720	0.900	0.870
		20	0.820	0.830	0.565	0.700	0.729
	100	**5**	0.790	0.950	0.720	0.920	0.845
		20	0.928	0.890	0.640	0.655	0.778
Mean			0.865	0.903	0.661	0.794	

Overall, the resulting proportions were fairly large, with the highest proportion of 100% correctly identified relevant predictors in Simulation 10. The parameter *t* controlled the amount by which $\mu_{X_j}, j = 1, \ldots, P$, differed for relevant vs. irrelevant predictors. A higher value of *t* created a larger difference in μ_{X_j}, and the distinction between the relevant and irrelevant predictors was more prominent, all other parameters kept constant. In Table 6.3, the average VI_{prop} is higher for larger values of *t*, for low and high label correlation scenarios. Therefore, the LCC classifier was more successful at identifying the relevant predictors when *t* was large.

Stellenbosch University https://scholar.sun.ac.za

In Section 6.4.1 it was stated that higher label dependence results in a stronger input signal in the data generation process. However, the LCC classifier was on average much more successful in identifying the relevant inputs when ρ was 0.1, compared to a larger ρ, as observed in Table 6.3. Therefore, the signal increase as a result of the higher label correlation was a smaller factor in VI_{prop}, than the presence of the highly dependent labels. While the stronger signal present in the highly correlated setting made the distinction between relevant and irrelevant predictors more clear, the overall result was influenced more strongly by the presence of highly significant label predictors, with a negative effect on VI_{prop}.

Considering the number of training observations, it would be expected that more data cases would allow for easier identification of relevant predictors by the classifier, and a higher value of VI_{prop}. However, the average results did not show a clear increase or decrease in the proportions.

Classifying 20 labels consistently resulted in the same or lower VI_{prop} values, compared to the case of classifying five labels. This result was expected, since in a more complex design the relevant predictors were likely to be more difficult to identify.

Finally, it was previously stated that increasing the value of P leads to a higher number of relevant predictors out of all $\sqrt{P}$ candidate predictors at the tree splits. However, based on the simulations, the number of predictors did not seem to have a clear positive or negative impact on VI_{prop}.

This analysis is meant to be a brief indication of the ability of LCC to identify relevant predictors correctly. However, a couple of things were assumed about the way in which the VI_{prop} was computed, so the results should be considered with care. Furthermore, the inconclusive results could also be attributed to the assumptions made for the purpose of this analysis. For example, note that the computation of VI_{prop} was based on the randomly generated matrix $\boldsymbol{A}_{mat}$, for which the proportion of relevant predictors fluctuated and was not a constant value of 20%.

Stellenbosch University https://scholar.sun.ac.za

In the next section, Friedman's test is used to determine whether there was a significant difference between the ML classifiers in low and high label dependence settings. This is the final section of the analysis based on the simulated datasets.

6.6 CONFIRMATORY ANALYSIS

In this section, Friedman's test is used to compare the four ML classifiers. The 32 simulation settings were split into 16 simulations with ρ = 0.1, and 16 simulations with ρ = 0.8. Friedman's test was then performed individually for each of the six ML evaluation measures, separately in low and high correlation settings. For every measure and label correlation setting, the 16 simulations represented 16 blocks in the Friedman's design, and the BR, CC, LCC and ECC classifiers were the treatments.

Due to the nature of the simulation process, the blocks were independent. Therefore, the statistical interpretation of Friedman's tests was acceptable. Of interest was to compare the four classifiers based on the six evaluation measures in low and high label correlation settings. The critical value for all 12 Friedman's tests was χ^2_{Crit} = 7.815, with 3 degrees of freedom. Each of the tests involved testing the following hypothesis at 5% level of significance:

H_0: There are no significant differences amongst the four classifiers,

H_A: At least two of the four classifiers differ significantly.

The results of the 12 Friedman's tests are summarised in Table 6.4. For every test, the mean rank $\bar{r}_k$ of every classifier, test statistic χ^2_F and the corresponding p-value and decision rule are provided.

Comparing the classifiers in terms of the HL and for ρ = 0.1, the LCC classifier performed the best, followed by BR, ECC and CC. For highly correlated simulation designs with ρ of 0.8, BR improved and scored almost as well as LCC, while CC and ECC performed worse than for ρ = 0.1. The differences between the classifiers were considered significant in both cases.

Stellenbosch University https://scholar.sun.ac.za

Table 6.4: Friedman's tests for the simulations analysis.

HL, ρ = 0.1	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.63	3.03	1.56	2.78
χ_F^2	12.06			
p-value	0.007	Significant difference		

HL, ρ = 0.8	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	1.94	3.22	1.91	2.94
χ_F^2	13.22			
p-value	0.004	Significant difference		

CL, ρ = 0.1	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.88	3.16	1.53	2.44
χ_F^2	14.53			
p-value	0.002	Significant difference		

CL, ρ = 0.8	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.66	2.88	1.75	2.72
χ_F^2	7.44			
p-value	0.059	Not significant difference		

PR, ρ = 0.1	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	1.94	1.63	2.81	3.63
χ_F^2	23.48			
p-value	3.215E-05	Significant difference		

PR, ρ = 0.8	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	1.47	1.94	2.88	3.72
χ_F^2	28.86			
p-value	2.401E-06	Significant difference		

RE, ρ = 0.1	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.91	3.34	1.84	1.91
χ_F^2	15.94			
p-value	0.001	Significant difference		

RE, ρ = 0.8	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.56	3.34	1.44	2.66
χ_F^2	17.94			
p-value	0.0005	Significant difference		

Stellenbosch University https://scholar.sun.ac.za

F2, ρ = 0.1	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.91	3.34	1.66	2.09
χ_F^2	16.84			
p-value	0.0008	Significant difference		

F2, ρ = 0.8	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.47	3.25	1.56	2.72
χ_F^2	14.31			
p-value	0.003	Significant difference		

AC, ρ = 0.1	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.94	3.31	1.72	2.03
χ_F^2	16.14			
p-value	0.001	Significant difference		

AC, ρ = 0.8	***BR***	***CC***	***LCC***	***ECC***
$\bar{r}_k$	2.44	3.25	1.59	2.72
χ_F^2	13.78			
p-value	0.003	Significant difference		

For the CL measure, LCC outperformed all other classifiers by a large margin, in both cases of label dependence. Therefore, the LCC classifier managed to correctly predict exact label sets that corresponded to the test data the best. The differences between the classifiers are significant only for the low correlation design, at 5% level of significance.

The results for the PR measure favoured the BR and CC classifiers, followed by LCC and ECC, in both correlation cases. In both the scenarios, the differences between the classifiers were significant. In terms of the RE measure, the differences between the classifiers were significant in both scenarios as well. The LCC classifier ranked better than all other classifiers when the RE measure was considered.

For both the F2 and AC measures, the differences between the classifiers were found to be significant. In all scenarios, LCC performed the best, followed by either BR or the ECC classifier. The CC classifier had consistently the worst rank for these measures.

Overall, the Friedman's tests provide evidence for significant differences between the BR, CC, LCC and ECC classifiers. The LCC classifier did very well in terms of the HL, CL, RE, F2 and AC measures, but not for the PR measure. These results signify

Stellenbosch University https://scholar.sun.ac.za

that it is essential to consider multiple evaluation measures simultaneously, as they may favour different classifiers.

A final way of comparing the four classifiers based on Friedman's tests is given in Table 6.5. In this table, the mean ranks of the classifiers were averaged over all evaluation measures, for the low and high label correlation scenarios. Of interest was to determine whether certain classifiers tend to perform better in either of the label dependence scenarios.

The BR classifier overall scored a better rank for $\rho = 0.8$ compared to $\rho = 0.1$, relative to the other classifiers. Considering that the BR classifier does not take the label predictor information into account, it was not expected to perform better. However, as was mentioned earlier, there is an interaction between label dependence and strength of the input predictor signal in the data generation process. As the label correlation and signal strength increased, the BR classifier managed to predict the labels more effectively in the highly correlated case, relative to the other classifiers. Interestingly, by not taking the label information into account, the focus of the BR classifier was only on the input features, which lead to a relatively better performance than the CC and ECC classifiers, in the highly correlated case.

Overall, the LCC classifier scored the best mean rank in both the low and high correlation cases, followed by BR, ECC and CC. It seems that by introducing some structure into the ensemble method of classifying ML data, the performance of the model could be significantly improved.

Table 6.5: Summary of the mean ranks of Friedman's tests based on all evaluation measures.

	BR	**CC**	**LCC**	**ECC**
$\rho = 0.1$	2.70	2.97	1.85	2.48
$\rho = 0.8$	2.26	2.98	1.85	2.91
Mean	2.48	2.97	1.85	2.70

Stellenbosch University https://scholar.sun.ac.za

The Friedman's analysis of the differences amongst the classifiers conclude the analysis of the simulated ML data. The exploratory, as well as the confirmatory results are summarised in the next section.

6.7 CONCLUSION

In this chapter a simulation study was conducted, in order to gain a deeper understanding of ML data and the classification thereof. Four ML classifiers were compared based on six evaluation measures and 32 simulation scenarios. The effect of modifying the controlling parameters N_{train} , P, L, ρ and t was analysed. Furthermore, the distribution of the variable importance values, as well as the ability of the LCC classifier to correctly identify relevant input features was explored.

The boxplots exploratory analysis revealed that increasing the values of ρ, N_{train}, P and t lead to an overall improvement of the ML classifiers. The improved performance as a result of increased label dependence was partly due to the nature of the simulation data generation, where the increased ρ lead to a stronger predictor signal. On the other hand, predicting a larger number of labels resulted in weaker performance of the classifiers.

The variable importance information obtained from the LCC classifier was summarised using heatmap graphs. Of interest was to observe how the distribution of these values varied for different values of the five controlling parameters. Furthermore, the effect of these parameters on the ability of LCC to correctly identify relevant input variables was observed for the 32 simulation settings. It turned out that increasing the value of t resulted in a larger proportion of correctly identified variables, while a larger value of the ρ and L parameters lead to a smaller proportion. Parameters P and N_{train} did not show either positive or negative effect on the proportion.

Finally, the relative performance of the LCC classifier in the low and high correlation designs was compared to that of the other classifiers using a Friedman's test, for the six evaluation measures. The LCC classifier outperformed the BR, CC and ECC classifiers for HL, CL, RE, F2 and AC, but not for the PR measure.

Stellenbosch University https://scholar.sun.ac.za

Overall, based on the simulation analysis, the LCC proved to be a competitive ML classification technique, which can perform better than the BR, CC and even ECC classifiers. Moreover, it provides a valuable output of variable importance values that can be used for inference or further analysis of the data.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 7: Credit bureau dataset analysis

7.1 INTRODUCTION

This chapter contains a discussion of the final experimental work, which was conducted using a practical ML dataset obtained from a South African credit bureau. Following the benchmark datasets and simulation analyses, the purpose of the work reported in this chapter is to compare the performance of the four ML classifiers in a real data application. The analysis involves a description of the dataset, screening of predictors, as well as a comparison of the performances of the four classifiers. Furthermore, heatmap analyses of the variable importance values obtained from the LCC classifier are provided.

7.2 DATASET

The ML credit bureau dataset contained information on 87143 clients who had opened one or more credit accounts. For each client, 1410 explanatory variables were recorded, of which 174 were qualitative. The variables described various characteristics of the clients, including payment and credit related information that were collected by the bureau. For every credit account that was opened by a client, the corresponding account label was assigned a value of 1, whereas a value of 0 indicated that the account had not been opened. There were in total 15 labels present in this dataset, which represented the accounts information for a specific client.

In order to explore the structure of the credit bureau dataset in more detail, the dataset was randomly split into 50% training set, 25% validation set and 25% test set. The following exploratory analysis of the dataset was performed on the training and validation sets. In Table 7.1 the general characteristics of this dataset are summarised. There were on average 1.58 opened accounts per client, and the average label density of the dataset was 0.11. Furthermore, the dataset was characterised by 759 distinct label combinations.

Stellenbosch University https://scholar.sun.ac.za

Table 7.1: Summary of the credit bureau dataset.

Dataset	N_{train}	N_{val}	N_{test}	L	P	$LCard$	$LDens$	***Distinct***
Credit bureau dataset	43571	21786	21786	15	1410	1.58	0.11	759

In Table 7.2 the distribution of the number of accounts per client is summarised. Out of the 15 accounts, all clients had at least one account opened and none of the clients opened more than eight accounts simultaneously. The mean number of accounts per client of 1.579 corresponds to the *LCard* value in Table 7.1.

Table 7.2: Distribution of number of accounts per client.

Min	Q_1	Q_2	**Mean**	Q_3	**Max**
1.000	1.000	1.000	1.579	2.000	8.000

In Figure 7.1 the relative frequencies of the number of accounts opened per client, $|\boldsymbol{y}_i|,\ i = 1, \ldots, N,$ are depicted (R code in Appendix D.1). Based on this figure, it is clear that a majority of the clients opened only one account at a time, and the proportions of clients with more than one account decreased as the number of accounts increased. This figure indicates that clients would typically have less than about half of the accounts opened at a time. Furthermore, only about 0.2% of the clients had 6 or more accounts opened simultaneously. Based on this figure it can be concluded that the credit bureau dataset is an ML dataset and can therefore be used for ML classification analysis.

The densities of the individual accounts, $\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_{ij},\ j = 1, \ldots, L,$ are depicted in Figure 7.2 (R code in Appendix D.1). The dataset contained a fairly unbalanced distribution of labels. Based on the label densities, the labels can be split into two groups. The first group contains eight labels that were opened by more than 5% of the clients. In this group, account 2 was the most frequently present label in the dataset. The second group contains seven accounts that were opened by less than 3% of the clients. Classifying such an unbalanced dataset can be challenging, because the limited number of observations that correspond to the infrequent accounts may be insufficient for model fitting.

Stellenbosch University https://scholar.sun.ac.za

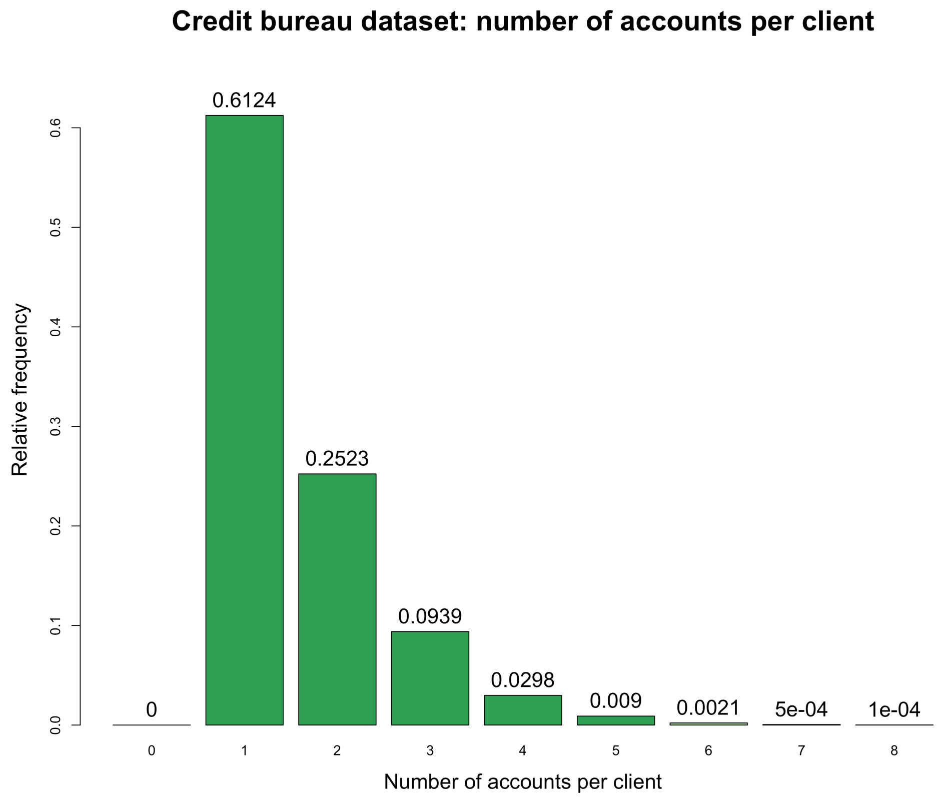


Figure 7.1: Distribution of the number of accounts per client.

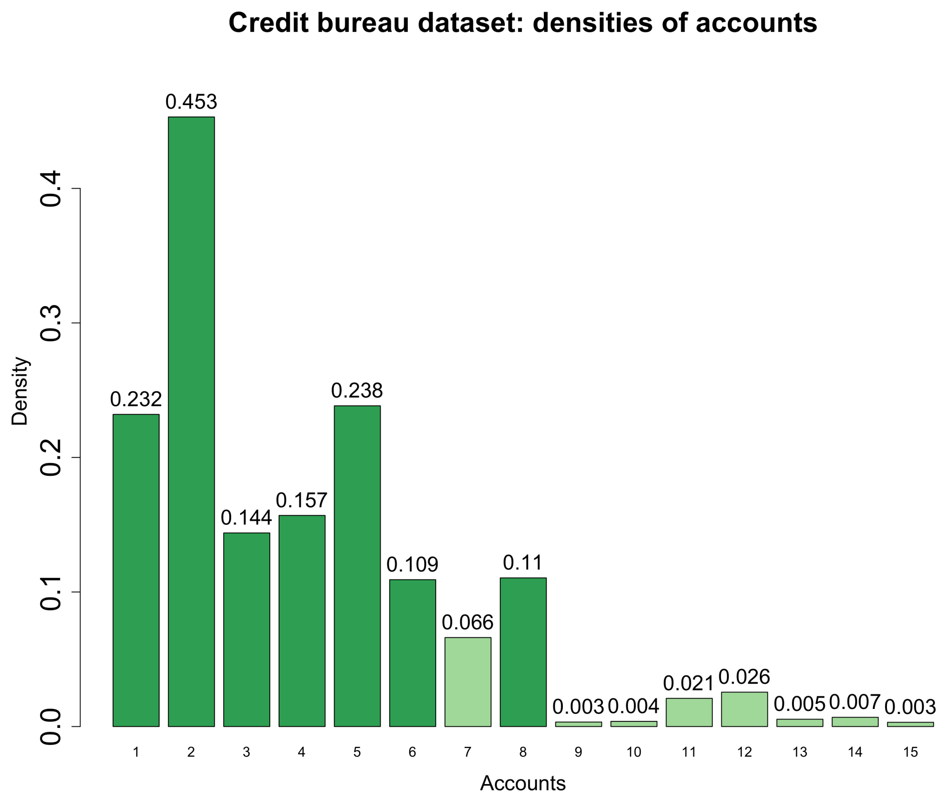


Figure 7.2: Densities of accounts in the credit bureau dataset.

Stellenbosch University https://scholar.sun.ac.za

7.3 EXPERIMENTAL DESIGN

The credit bureau dataset is a fairly large dataset consisting of 1410 predictors, 15 labels and 43571 training observations. In general, having a large number of predictors may be detrimental to the model, especially if a fairly large number of the predictors are irrelevant. This can be especially true for datasets from the credit bureau industry, where the predictors stored for every client usually reflect all credit related information. Therefore, a pre-processing step of removing some of the irrelevant predictors was conducted in order to achieve better performance.

While RF perform VS as part of the algorithm, if the ratio of irrelevant to relevant predictors is very high, the performance of the RF models may deteriorate. The latter is as a result of considering $\sqrt{P}$ predictors as candidates at each tree split, as explained by the following example.

Consider for example two datasets, one consisting of 1000 predictors and another one of 500 predictors. Each of the datasets has only 50 relevant predictors and the rest do not contribute to the fit. Therefore, 5% of the predictors are relevant in the first dataset and 10% are relevant in the second one. In BR, the RF model chooses from $\sqrt{1000} \cong 32$ and $\sqrt{500} \cong 22$ randomly selected candidate predictors at each split, for the two datasets respectively. This corresponds to on average 1.6 relevant predictors out of the 32 variables in the first dataset, and 2.2 relevant predictors out of the 22 variables in the second dataset. Therefore the RF model will on average choose more relevant predictors as candidates and perform better when the ratio of relevant to all predictors is larger.

Furthermore, the number of RF that need to be fitted by the ensemble classifiers is 225, since in LCC and in ECC with $M = L$, each of the 15 classifier chains uses 15 RF models to predict the corresponding labels. Consequently, reducing the number of predictors may not only improve the performance of a classifier but also make the modelling more efficient. The following analysis was performed in order to determine whether removing some of the predictors prior to the ML classification would be beneficial. A linear regression model was fitted to the training dataset using X_j as the response variable and the 15 labels as input variables, as summarised by:

Stellenbosch University https://scholar.sun.ac.za

$$X_{ij} = \beta_{0j} + \sum_{l=1}^{L=15} \beta_{lj} Y_{il} + \varepsilon_i\,, \qquad i = 1, \dots, N_{train}.$$

The linear regression was repeated P times, each time using a different X_j as the response variable, $j = 1, \dots, P$. For the purpose of this analysis, the qualitative predictors were converted into numeric predictors, according to the factor level. For every fitted model, the corresponding coefficient of determination was computed as:

$$R_j^2 = 1 - \frac{SS_{res_j}}{SS_{tot_j}},$$

$$SS_{res_j} = \sum_i \left(x_{ij} - \hat{x}_{ij}\right)^2, \qquad SS_{tot_j} = \sum_i \left(x_{ij} - \bar{x}_j\right)^2, \qquad j = 1, \dots, P.$$

The R^2 values were computed in order to quantify the strength of the relationship between all of the labels and the individual predictors. The values ranged from 0 to 1, with a high value of R^2 indicating a strong linear relationship between the independent and dependent variables. On the other hand, predictors with low values of R_j^2 showed little to no relationship with the labels and were likely not highly relevant in predicting the labels. Table 7.3 and Figure 7.3 provide a summary of the distribution of R^2 values (R code in Appendix D.2).

Table 7.3: Summary of the R^2 distribution.

Min	Q_1	Q_2	***Mean***	Q_3	***Max***
0.000	0.034	0.088	0.099	0.150	0.500

Based on the linear models fitted, the average R^2 value was 0.099, which indicated that on average the predictors were not very well explained by the labels. The largest value of R^2 was 0.5, but as can be seen in Figure 7.3 below, the frequency of such values was very low. The figure also shows that most of the $\boldsymbol{X}$ predictors had a weak linear relationship with the accounts.

Stellenbosch University https://scholar.sun.ac.za

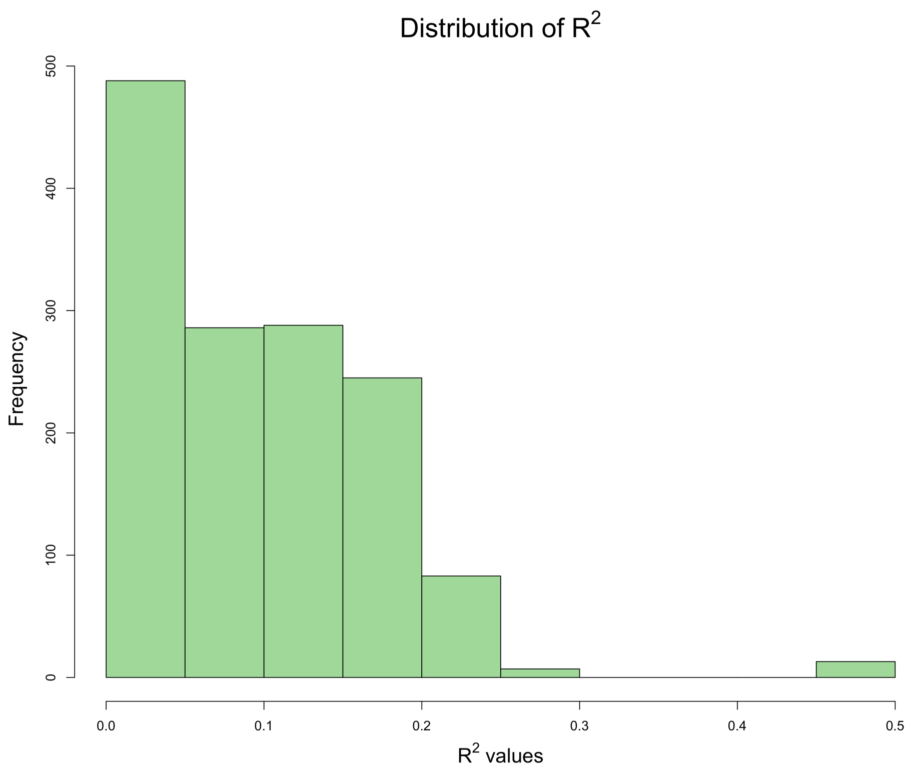


Figure 7.3: Distribution of the R^2 values.

In order to determine whether some of the 1410 predictors could be omitted, the following analysis was conducted based on the R^2 values. The BR classifier was fitted to the training data using 50 RF trees and the validation set was used to compute the six evaluation measures. This process was repeated eight times, each time using fewer numbers of predictors. The predictors for which the corresponding R^2 value was larger than a specified percentile of R^2 values were included in the BR classification. The percentile is referred to as threshold below and it was set to values 0%, 25%, 50%, 75%, 80%, 85%, 90% and 95%. The following table summarises the number of predictors included in the analysis for the eight scenarios.

Table 7.4: Threshold specification and corresponding number of predictors.

Threshold	**0%**	**25%**	**50%**	**75%**	**80%**	**85%**	**90%**	**95%**
P	1410	1066	706	354	282	214	142	71

In Figure 7.4, the ML evaluation measures are displayed for each threshold value (R code in Appendix D.2). For each of the plots, the range of the y-limits is 0.06, so that

Stellenbosch University https://scholar.sun.ac.za

the relative differences of the measures can be observed. From this figure it can be seen that removing up to 85% of the predictors did not seem to result in a large performance change of the BR classifier. However, disregarding 90% or 95% of the predictors lead to diminished performance in terms of most of the measures. Interestingly, precision started to improve at 90% and 95% threshold values.

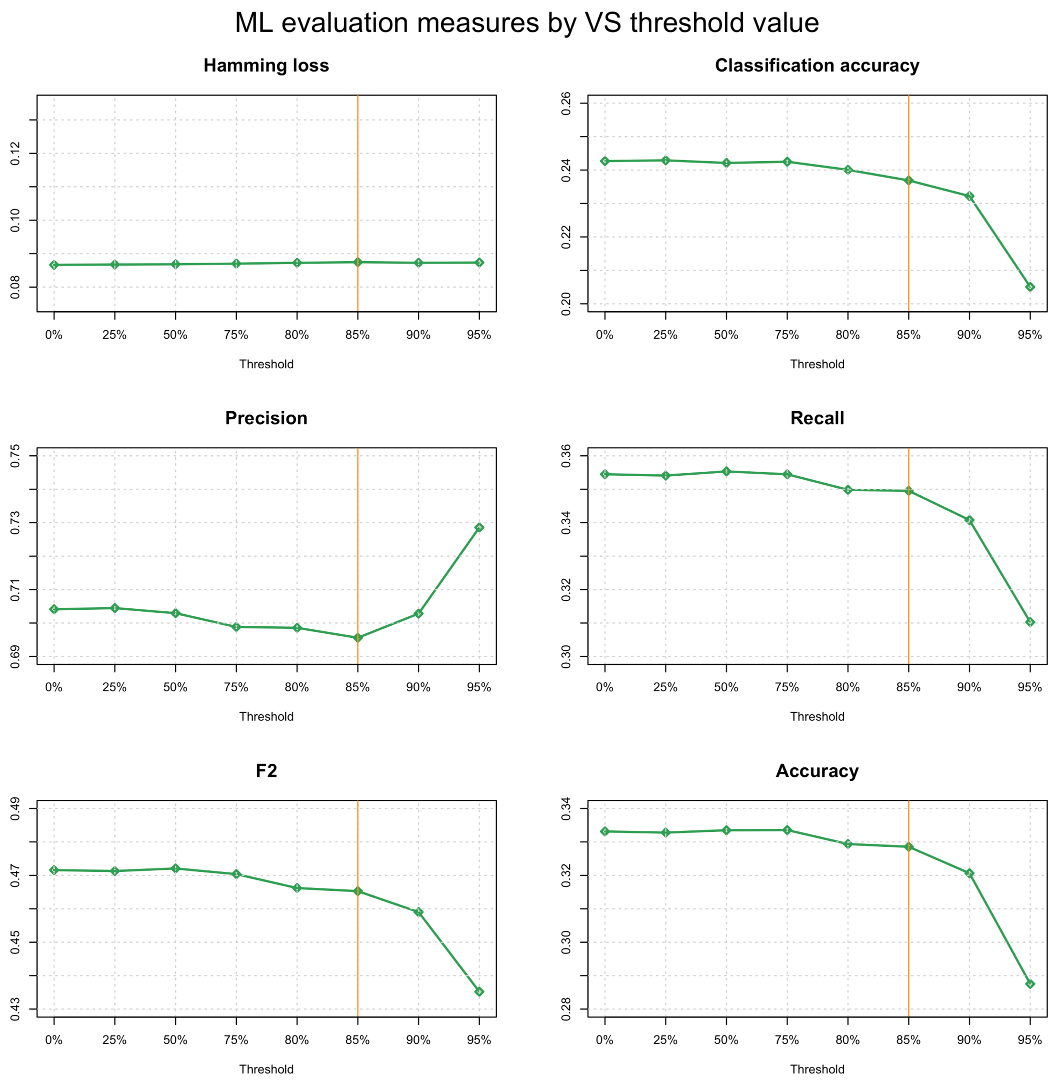


Figure 7.4: Evaluation measures for different threshold values.

For the purpose of this analysis, only 15% of the predictors were therefore included in the subsequent ML classification. Note that the screening step is not a part of the LCC classifier but it allowed for much faster ML classification. Furthermore, the screening was performed using only the BR classifier and it was assumed that the selected predictors would be relevant for the BR-based CC, ECC and LCC classifiers.

Stellenbosch University https://scholar.sun.ac.za

In order to assess the performance of the four ML classifiers for the credit bureau data, ML classification was done in the following way. Random forests of 200 trees were fitted to both training and validation sets, in order to train the model on all available data, and evaluated on the test set. The Gini index was used throughout the analysis. The six evaluation measures as well as the computational time and the variable importance values from LCC were recorded. The results are provided in the next section and they serve as an indication of the accuracies that can be obtained for the credit bureau data using the four classifiers.

7.4 RESULTS

The results on the test data of the credit bureau analysis are summarised in Table 7.5 (R code in Appendix D.3). The BR classifier performed the best in terms of the HL and PR measures, but ranked overall the worst when compared to the other classifiers. The CC method resulted in better rank than BR and an improved performance in terms of all measures except for HL and PR. These two classifiers also took about the same time to implement. The time of the analysis refers to the fitting of the models as well as making predictions for the test data.

The ensemble methods managed to outperform BR and CC in terms of CL by about 1%, RE by 10%, F2 by 5% and AC by roughly 7.5%. The ECC and LCC classifiers performed worse than BR and CC for the HL measure by about 1% and the PR measure by 10%. There were minor differences between the results of ECC and LCC. The LCC classifier however overall scored a better rank than ECC. The computational time of the ensemble methods was much longer than that of BR and CC and the LCC classifier took the longest time to compute.

Stellenbosch University https://scholar.sun.ac.za

Table 7.5: Credit bureau classification results for 15 labels.

	BR	**CC**	**ECC**	**LCC**
HL	0.0861	0.0867	0.0962	0.0944
CL	0.2441	0.2575	0.2627	0.2737
PR	0.7156	0.6893	0.5840	0.5925
RE	0.3489	0.3651	0.4944	0.4809
F2	0.4691	0.4774	0.5355	0.5309
AC	0.3291	0.3460	0.4164	0.4175
Time	20.88 min	20.44 min	3.18 h	5.23 h
Rank	3.00	2.67	2.33	2.00

The ranks in Table 7.5 were computed by putting an equal weight on all evaluation measures. However, in credit bureau applications, not all evaluation measures may be equally important. Note that the credit bureau would like to use this dataset for direct marketing. They would like to identify the accounts that clients are likely to open and advertise such accounts to them via SMSs or emails. In binary classification, there are two possible errors that can be made when classifying a label, as summarised in Table 7.6. The first error rate is called the false positive rate, FPR = $\frac{FP}{FP+TN}$, which is the proportion of accounts predicted as opened, when in fact the accounts were unopened. The second error rate is the false negative rate, FNR = $\frac{FN}{FN+TP}$, which is the proportion of opened accounts predicted as unopened. The FPR is also known as the Type I error rate (α) and FNR is the Type II error rate (β). Furthermore, the proportion of correctly classified opened accounts is known as the true positive rate, TPR = $\frac{TP}{TP+FN}$, also called sensitivity or recall. Lastly, correctly classifying unopened accounts results in the true negative rate or so-called specificity, TNR = $\frac{TN}{TN+FP}$.

Stellenbosch University https://scholar.sun.ac.za

Table 7.6: Confusion matrix.

		Prediction	
		0	1
Actual	0	True Negative (TN)	False Positive (FP)
	1	False Negative (FN)	True Positive (TP)

Minimising both FPR and FNR is usually of interest. However, these error rates are inversely related, so by adjusting a procedure to lower the one, the other consequently increases. In the credit bureau application, falsely predicting opened accounts as unopened (FNR) is not as serious as falsely predicting unopened accounts as opened (FPR). Keeping the cost of the direct marketing strategy low and not spamming clients is important, even at the cost of not marketing to potential customers. Note that FNR = 1 – TPR(Recall). Therefore, if a low value of FPR is desired, a higher value of FNR and a lower value of Recall need to be accepted. From Table 4.5 it can be observed that BR and CC resulted in lower values of RE than ECC and LCC. It is therefore suggested that the credit bureau considers this aspect of the analysis and decides whether the reduction in FPR is worth the loss of accuracy in some of the ML evaluation measures. Furthermore, future research of LCC can involve incorporating the cost of FPR when determining the value of the threshold t.

Returning to Table 7.5, it is seen that overall, the HL measure was fairly low, indicating that only about 9% of all individual accounts were predicted incorrectly. This was likely due to the low overall label density of 0.11. Indeed, an additional analysis that considered only the labels with densities exceeding 1%, resulted in an increase of the overall label density to 0.16 and a higher value of HL of about 13%. Moreover, there were only minor differences among the other evaluation measures.

In order to gain a better understanding of the classification performance in this unequally distributed dataset, a number of measures were recorded for every label individually, summarised in Table 7.7 (R code in Appendix D.4). Consider first HL. The HL measure of a label is the proportion of misclassified observations for that label, given by

Stellenbosch University https://scholar.sun.ac.za

$$\mathrm{HL} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I\big(Y_i \neq \hat{Y}_i\big).$$

The densities of the individual labels were defined in Section 7.2 and graphically illustrated in Figure 7.2. The area under the curve (AUC) is a well-known binary classification performance measure, not affected by the distribution of the response. In order to illustrate this concept, consider label l in the credit bureau dataset. The LCC classifier was used to make predictions for label l for all test cases. Remember that in LCC each of the L CC votes for the labels it predicts and a threshold t is then used to determine the final set of predicted labels (see Section 3.3.3). For the purpose of computing probabilities of the predictions, the votes in the $\boldsymbol{W}$ matrix are not thresholded, but divided by L in order to compute the probability of predicting label l:

$$\hat{P}(Y_{il} = 1) = \frac{1}{L} W_{il}, \qquad i = 1, \dots, N_{test}; \;\; l = 1, \dots, L.$$

Note that an alternative approach for estimating the probability for label l would be to average the probabilities obtained directly from the RF models fitted for each of the L CC. For a given label, the probabilities as well as the true label values were used to compute TPR and FPR for all values of a decision threshold. Whereas the RF models classify an observation to class 1 if $P(Y = 1) > 0.5$ and to class 0 otherwise, by considering all thresholds it is possible to quantify the performance based on the probabilities. For example, a probability of 0.9 indicates a higher confidence in the prediction than 0.51. The receiver operating characteristic (ROC) curve graphically represents the relationship between TPR and FPR simultaneously as the probability threshold is varied.

Examples of the ROC curves are depicted in Figure 7.5 (R code in Appendix D.4). The LCC classifier was used to predict the labels and the ROC threshold values ranged between 0 and 1. By moving diagonally from the bottom left corner of the graphs, where the threshold was the highest, it can be seen that lowering the value of the threshold results in more positive predicted labels and an increase in both TPR and FPR. For a good classifier, TPR increases at a faster rate than FPR. The performance of a classifier can therefore be quantified in terms of the area under the ROC curve (AUC). An AUC of 0.5 implies that the classifier is no better at

Stellenbosch University https://scholar.sun.ac.za

differentiating between the label classes than random guessing (James *et al.*, 2013:147). In Figure 7.5, the ROC curves for labels 2 and 3 are plotted, where the performance of the classifier for label 2 is superior to that for label 3.

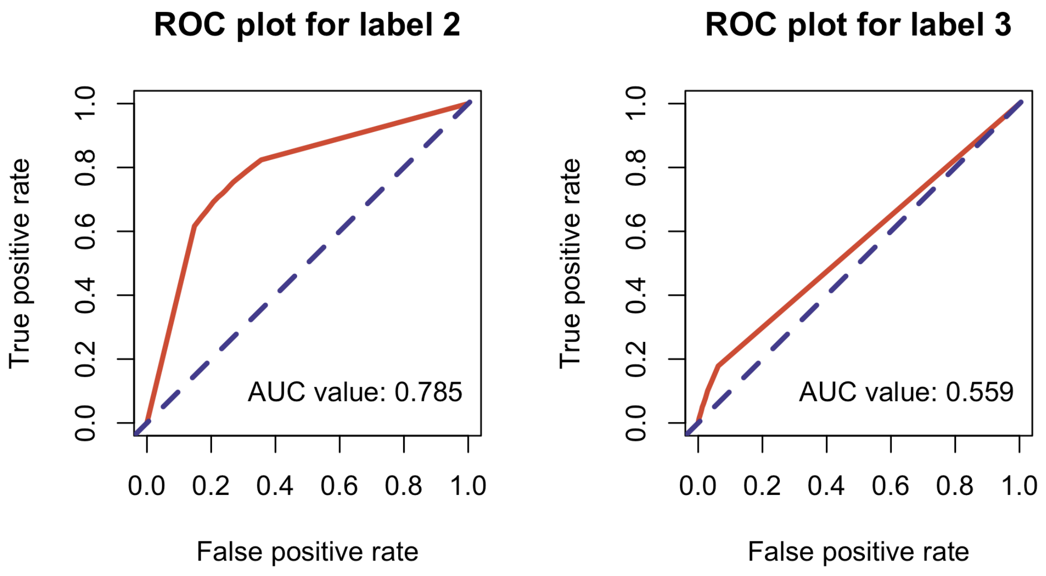


Figure 7.5: ROC curves for labels 2 and 3 for the credit bureau data.

The last measure summarised in Table 7.7 is the Gini coefficient. According to Japkowicz and Shah (2014:129), the Gini coefficient is a measure of model performance and it is directly computed from the AUC in the following way:

$$Gini = |1 - 2 * AUC|.$$

The Gini coefficient is commonly used in the credit risk industry. It ranges between 0 and 1 and it was computed in order for the credit bureau to be able to compare our model to other models. A Gini coefficient of 0 (AUC = 0.5) indicates that the classifier is no better than a random classifier. On the opposite scale, a Gini coefficient of 1 (AUC = 1) corresponds to a classifier that can differentiate between label classes perfectly. In the credit industry, according to Siddiqi (2006:124), an AUC of at least 0.7 (Gini of 0.4) is considered adequate.

The evaluation measures in Table 7.7 were computed in the following way. The LCC classifier was used to predict the 15 labels of the test data and for each label the individual HL value was determined, as defined above. Furthermore, for each of the

Stellenbosch University https://scholar.sun.ac.za

labels, the TPR and FPR values were computed for a set of probability thresholds ranging between 0 and 1. The AUC of the corresponding label was then quantified as the area under the ROC curve, given the values of TPR and FPR, and the Gini value were calculated from AUC.

Table 7.7: Evaluation measures per label for the LCC classifier.

	Label 1	Label 2	Label 3	Label 4	Label 5	Label 6	Label 7	Label 8
Density	0.232	0.453	0.144	0.157	0.238	0.109	0.066	0.110
HL	0.256	0.274	0.17	0.18	0.158	0.132	0.07	0.109
AUC	0.608	0.785	0.559	0.547	0.740	0.590	0.502	0.579
Gini	0.216	0.570	0.118	0.094	0.480	0.180	0.004	0.158

	Label 9	Label 10	Label 11	Label 12	Label 13	Label 14	Label 15
Density	0.003	0.004	0.021	0.026	0.005	0.007	0.003
HL	0.004	0.003	0.021	0.024	0.005	0.006	0.003
AUC	0.605	0.507	0.500	0.502	0.517	0.500	0.500
Gini	0.210	0.014	0.000	0.004	0.034	0.000	0.000

Of the 15 labels, labels 9 to 15 had very low densities of less than 3%. Table 7.7 shows that HL corresponding to these seven labels is considerably lower than for the rest. Recall that a lower value of HL indicates fewer misclassified cases. However, it might be incorrect to conclude that the classifier performed better for label 15 than it did for label 1 based purely on HL. Due to the low density of label 15, a vast majority of both the true labels and the predicted labels had a value 0, which consequently resulted in very few misclassified cases.

The AUC and Gini coefficient values in Table 7.7 show a different trend than HL. The AUC value of label 15 is 0.5 and the Gini coefficient is 0, indicating that the classifier performed no better than chance. On the other hand, the AUC for label 1 is 0.608 with a Gini coefficient of 0.216. The LCC classifier was therefore more successful at differentiating between label classes for label 1 than for label 15.

Stellenbosch University https://scholar.sun.ac.za

Furthermore, only the labels of the highest density, labels 2 and 5, had an AUC coefficient larger than 0.7.

Overall, as label density decreased, the HL measure improved but AUC and the Gini coefficient deteriorated. These findings further point to the challenges associated with analysing ML datasets with an unbalanced distribution of labels. It is suggested that the HL values from Table 7.5 be interpreted with caution and that the individual label densities are considered before analysing an ML dataset. If the dataset consists of labels with very low densities, it might be useful to collect a larger dataset and possibly increase this density. If it is not possible to collect more data, it might be essential to use classification methods that aim at overcoming the bias of unequally distributed labels. Since ML datasets will typically suffer from this problem, there is definitely scope for further research.

7.5 VARIABLE IMPORTANCE RESULTS

The heatmaps of variable importance values resulting from the LCC classifier are given in Figure 7.6. In the top heatmap, the importance values obtained from the $\boldsymbol{A}_{final}$ matrix were standardised by dividing each value by the largest Gini value of the entire matrix. The bottom heatmap shows the $\boldsymbol{A}_{final}$ values that have been standardised by dividing each row of the matrix, which corresponds to a specific label, by the largest Gini value for that label. Therefore, in both heatmaps the importance values were standardised to an index ranging from 0 to 100, but the first one represents values relative to importance values of all other predictors, and the second one depicts values relative to the most important predictor for a given label.

The dendrogram in the top heatmap indicates that the labels can be split into three groups, based on the importance values: very low, medium and medium-high values. The first group consists of the seven labels that had the lowest variable importance values: labels 9, 10, 11, 12, 13, 14 and 15. These labels were identified to have the lowest density values out of all labels in Figure 7.2. For the five labels with densities below 1%, the importance values were too small to differentiate in the heatmap. Furthermore, accounts 11 and 12, with densities just above 2%, showed slightly larger importance values.

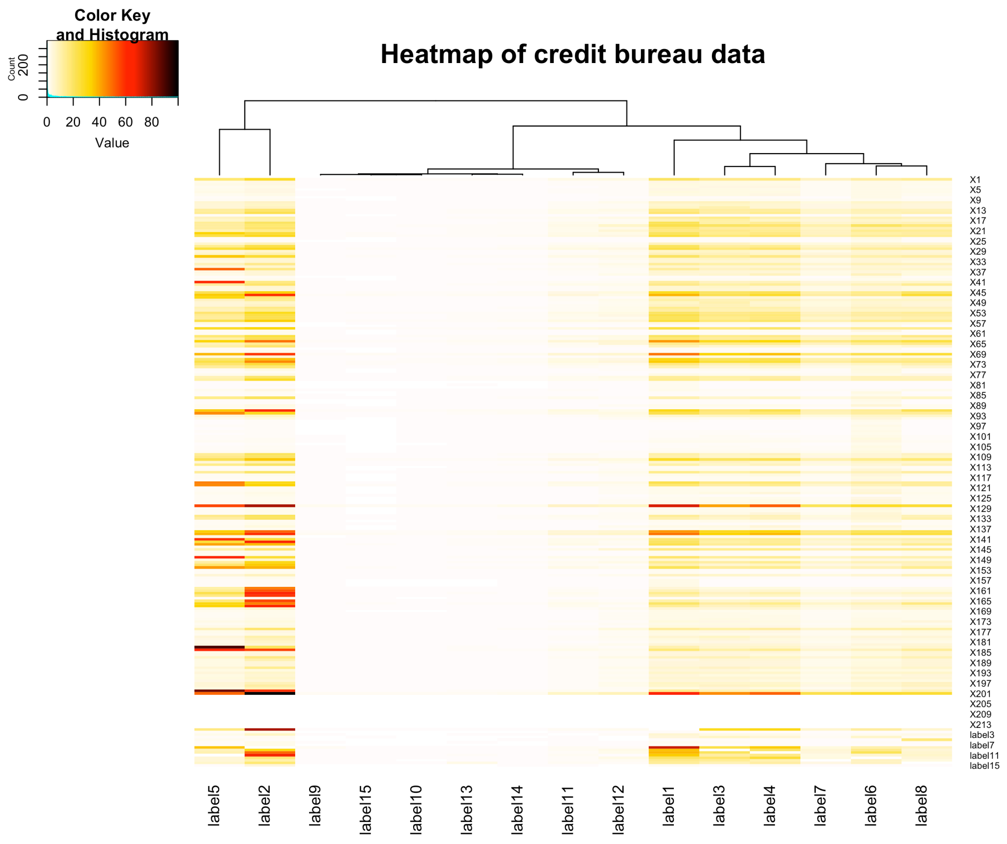
Stellenbosch University https://scholar.sun.ac.za

Labels 1, 3, 4, 6, 7 and 8 form the second group of labels with medium importance values, and the third group consists of labels 2 and 5 with medium-high values. Out of these two groups, labels 1, 2 and 5 had the largest densities and benefited the most from using the other predictors, as seen in the heatmap. These findings are intuitive, since the importance values reflect the average increase in label purity as a result of using a specific predictor in a RF model. A label with a higher label density is more likely to result in a larger increase in label purity than a label with lower density. Therefore, there is essentially no improvement for the five labels with densities below 1%. These labels could be ignored in the analysis, as mentioned in the previous section.

In the second heatmap, the local variable importance values were constructed in order to see the importance values clearly, irrespective of the label density. The horizontal lines in the heatmap indicate that some of the predictors were more relevant in an overall sense for improving the fit than others. It is clear that the importance values were fairly constant across the responses – some of the predictors were highly relevant for the majority of the labels, while others did not have an effect on the fit of most of the labels.

Based on this analysis, the most significant predictors could be identified either in a local or global fashion. Overall, the LCC classifier made use of other label predictors, as indicated by the medium-high importance values thereof.

Stellenbosch University https://scholar.sun.ac.za



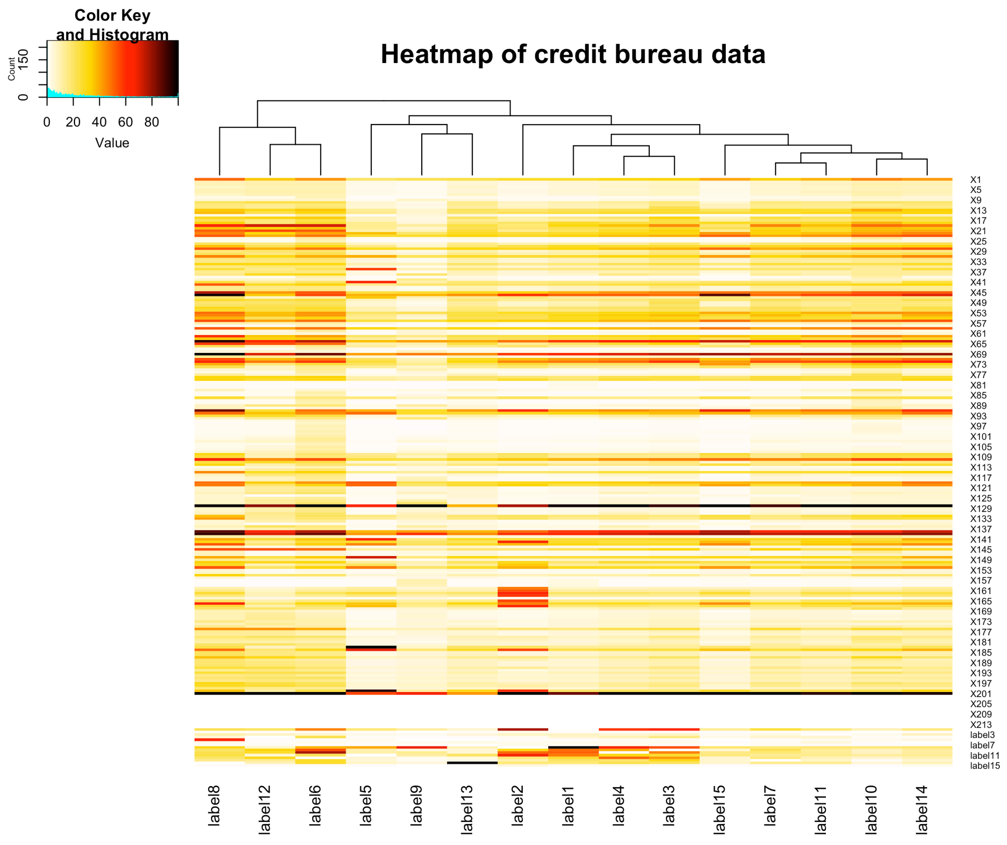


Figure 7.6: Heatmaps for the credit bureau data.

Stellenbosch University https://scholar.sun.ac.za

Figure 7.7 depicts the global importance values of all predictors in a variable importance plot. For every predictor, the values in this figure were computed as the average VI values for that predictor across all labels, based on the importance values from the top heatmap in Figure 7.6, and standardised to an index of 100:

$$VI_j = \frac{1}{L}\sum_{l=1}^{L} VI_{jl}, \qquad j = 1, \dots, L + P.$$

Recall that the 214 $\boldsymbol{X}$ predictors in Figure 7.7 are the most significant predictors out of the original 1410 inputs. It can be seen that the importance of the input and label predictors is distributed exponentially, with only a few highly significant predictors. Table 7.8 depicts the 10% most globally significant predictors. The top three inputs were X_{201}, X_{128} and X_{139}, and label predictors 8 and 1 were amongst the top 10% as well. Unfortunately, the meaning of these inputs cannot be disclosed in this thesis, but the credit bureau can use the information from this table to get a better understanding of their clients.

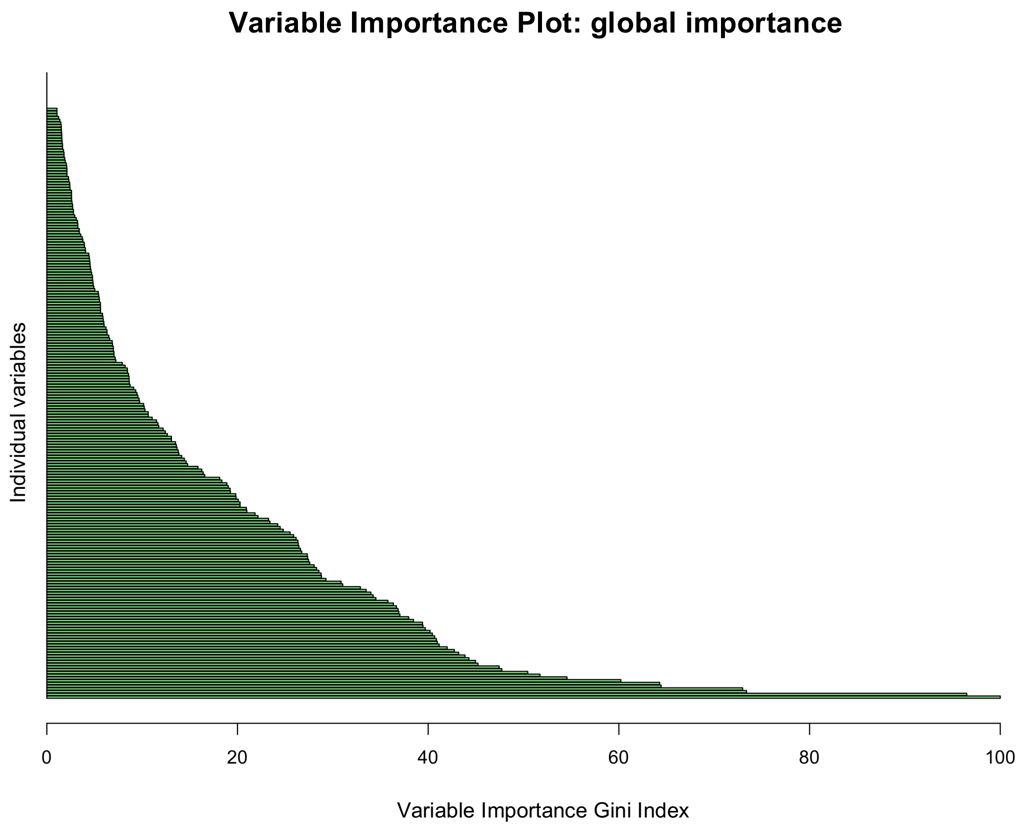


Figure 7.7: Variable importance plot for the credit bureau data.

Stellenbosch University https://scholar.sun.ac.za

Table 7.8: The 10% globally most important predictors and their Gini index values.

Globally important predictors					
No	*Predictor*	*Index*	*No*	*Predictor*	*Index*
1	X201	100	13	X45	45.22
2	X128	96.49	14	X110	44.96
3	X139	73.39	15	label1	44.27
4	X69	72.98	16	X71	43.86
5	X138	64.43	17	X119	43.2
6	X64	64.27	18	X92	42.74
7	X46	60.21	19	X148	41.99
8	X91	54.56	20	X59	41.16
9	X72	51.72	21	X152	40.96
10	X200	50.44	22	X56	40.87
11	label8	47.72	23	X143	40.68
12	X184	47.44			

Table 7.9 was constructed in order to summarise the five most locally relevant predictors per label. The most important predictors for some of the labels corresponded to those in Table 7.8, while others followed a different trend. For example, label 8 was the most important predictor for label 1 and label 14 was locally most significant for label 13. Again, the interpretation of these results will likely make more sense upon decoding the variable names.

The final graphical representation of the credit bureau data analysis is depicted in Figure 7.8. In this figure, the prediction probabilities (defined in Section 4.2) as well as the actual labels present are displayed for four randomly selected clients. In the top left graph, all of the 15 CC predicted account 2 as opened. The threshold t^* in this analysis was 1, therefore if at least one of the CC predicted an account to be opened, the final prediction for that label was 1 as well. This prediction was correct, since account 2 was truly opened for that client, as shown by the orange line. Therefore, the LCC classifier made correct predictions for this client.

The prediction for the second client shown in the top right graph was not as accurate. Label 2 was truly opened and 14 out of 15 CC predicted the label as opened. Furthermore, one of the CC voted for label 4, which was not truly present. Label 5 was completely missed by all of the CC. In the bottom left graph, label 2 was correctly predicted with high confidence and label 4 was falsely classified as not

Stellenbosch University https://scholar.sun.ac.za

present. Lastly, the graph in the bottom right corner indicates that label 2 was falsely predicted as present while label 4 was correctly identified as present, even though the corresponding probability was not very high.

Table 7.9: Summary of the five most important predictors and the corresponding index values for each label.

	Locally important predictors				
Response	1	2	3	4	5
Label 1	label8	X128	X201	X139	X69
	100	97.3	85.9	65.9	65.8
Label 2	X201	X128	label1	label11	X142
	100	77.7	77.5	67.5	67
Label 3	X201	X128	X139	X69	X64
	100	92.2	75.1	74.8	65.7
Label 4	X128	X201	X139	X69	label8
	100	98.6	71.6	71	67.3
Label 5	X183	X200	X148	X184	X41
	100	94.5	72.7	69	66.8
Label 6	X128	X201	X139	X69	label10
	100	99.8	85.8	85	77.4
Label 7	X201	X128	X139	X69	X64
	100	90.8	74.6	74	65.6
Label 8	X69	X201	X128	X46	X64
	100	99	98.2	98.1	94.6
Label 9	X128	label8	X201	X139	X69
	100	68.4	67.4	59.4	47.2
Label 10	X128	X201	X69	X139	X138
	100	98.9	80.3	79.5	66.5
Label 11	X128	X201	X69	X139	X138
	100	94.6	76.2	76.2	65
Label 12	X201	X128	X19	X139	X69
	100	81.5	72.4	71.4	70.9
Label 13	label14	X69	X139	X46	X201
	100	44.9	44.4	44.1	42.8
Label 14	X128	X201	X139	X69	X138
	100	95.7	81.2	79.8	72.9
Label 15	X128	X201	X46	X139	X69
	100	99.1	88.4	79.6	79.3

Stellenbosch University https://scholar.sun.ac.za

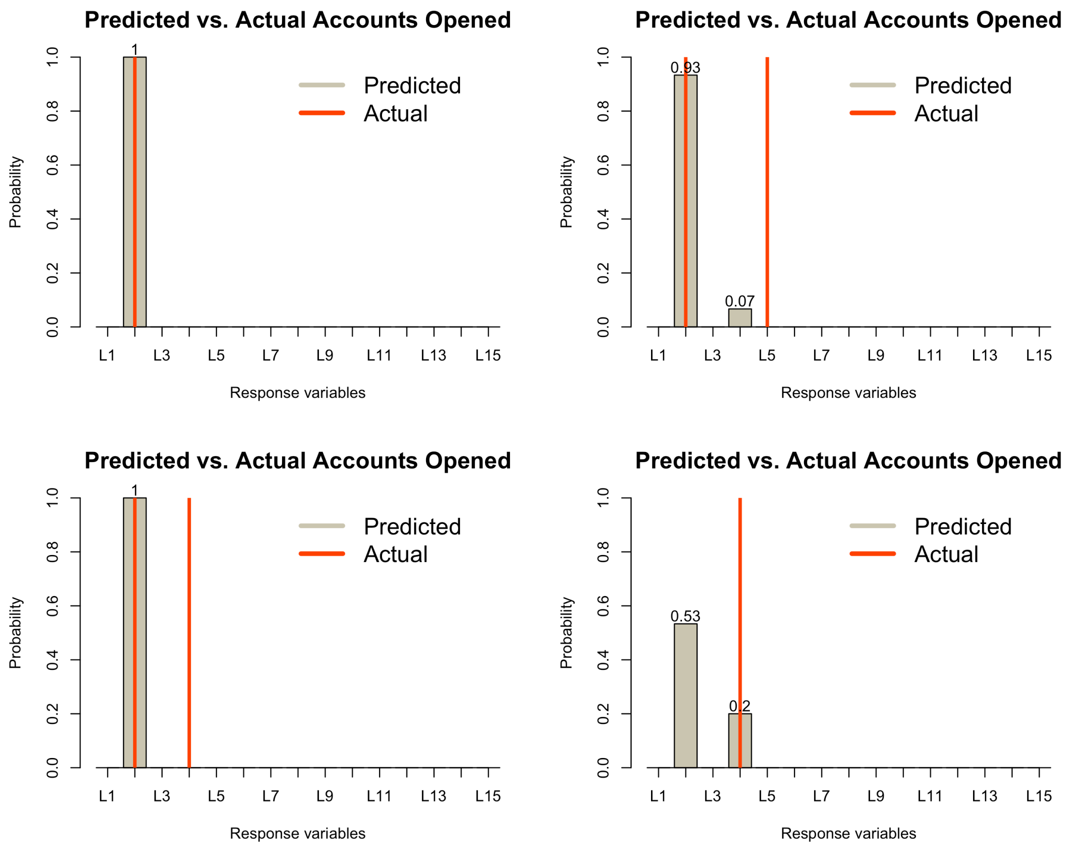


Figure 7.8: Graph of the predicted and actual accounts opened by four clients.

In future application of the model, the truly present labels will not be known for test cases, but the graph can still be used to plot the probabilities of opened accounts. The credit bureau can then determine whether the probability associated with a specific label is high enough to justify the cost of sending the client an SMS for the corresponding account. Overall, the graphs in Figure 7.8 are useful for understanding the ML analysis better and to use the LCC model in a way that would be suited for the application of interest. The R code that was used to construct all of the tables and figures in this section can be found in Appendix D.4.

Stellenbosch University https://scholar.sun.ac.za

7.6 CONCLUSION

In this chapter, analysis of the practical credit bureau dataset highlighted some of the important aspects of ML classification, such as the label density, label cardinality and dimensionality of the data. Of interest was to compare the BR, CC, ECC and LCC classifiers in a practical ML setting and to explore some interesting aspects of ML classification as well, including quantifying the importance of predictors.

The credit bureau dataset consisted of 15 labels and 1410 predictors. The label cardinality of 1.58 and label density of 0.11 indicated that the dataset was not very multi-labelled and that the labels were sparsely distributed. Five of the labels had density below 3%. The validation step showed that it was sufficient to include only 15% of the predictors and reduce the number of variables to 214. Furthermore, the four classifiers were used to predict previously unseen test data.

Overall, the ensemble classifiers outperformed BR and CC in terms of all measures except HL and PR. The computational time of the ensemble methods was longer than that of BR and CC. The LCC classifier ranked the best out of all classifiers. Its computational time was the longest, but it provided an output of variable importance values, from which the heatmaps of importance values could be constructed. Furthermore, HL, AUC and the Gini coefficient were computed for each of the labels separately. From this analysis it could be seen that as the label density decreased, HL improved but AUC and the Gini coefficient deteriorated. Therefore, the sparsely distributed labels were more difficult to predict accurately. However, depending on the nature of the ML problem, it may be less important to classify the sparsely distributed labels accurately.

The heatmap analysis showed that including the five labels with density below 1% lead to a very small increase in label purity, relative to all other labels. On the other hand, the more densely distributed labels had larger importance values. Some of the predictors were highly relevant for the majority of the labels, while the presence of other predictors did not affect the fit. Furthermore, the heatmaps showed that label dependence was present in the credit bureau dataset, as some of the VI values of other label predictors were moderately high.

Stellenbosch University https://scholar.sun.ac.za

The VI plot indicated that the importance of predictors was exponentially distributed. The 10% globally most significant predictors were identified as well as the five locally most relevant predictors for each label. Lastly, the prediction probabilities as well as the actual labels present in four different test cases were plotted and used as a tool to understand the output from the LCC classifier better.

While the LCC classifier did not lead to large classification improvements when compared to ECC, the variable importance outputs obtained by LCC were a very useful tool for ML inference, as shown in this chapter.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 8: Conclusion and future research ideas

In this chapter, a brief overview of the work presented in this thesis, as well as future research proposals are provided. The objective of the thesis was to propose a new ML classification method, which could incorporate VS and variable importance within the classifier. The first four chapters presented the theoretical background corresponding to ML classification, variable selection and variable importance values. The next three chapters presented experimental work, where the benchmark data analysis, simulation analysis and a practical credit bureau analysis were conducted in order to compare the proposed LCC classifier to other existing approaches.

Multi-label classification was presented as one of four classification methods in Chapter 2. The research for this type of classification is much less extensive than that for binary classification. This is partly as a result of the more complex structure of ML datasets, which was described in detail in this chapter. Conditional label dependence was identified as one of the complexity factors, mainly because it is difficult to estimate. As an idea for a future research initiative, it would be interesting to explore different ways of directly estimating the conditional label dependence. Chapter 2 also provided some of the most well known approaches to ML classification, with focus on the PT methods.

Chapter 3 presented some of the CC-based algorithms, including the original CC, 1-CC and ECC classifiers. The idea of having a single or multiple CC classifiers and random or non-random label sequences of labels in these classifiers was explored. The LCC classifier was introduced as an ensemble of L CC with semi-random sequence structure, which allows for an extraction of variable importance values. The contribution of LCC thus lies in its dual application of both ML classification and inference.

Some of the future research proposals regarding the LCC classifier include the following suggestions. Firstly, the semi-random structure of the label sequence could be adjusted to involve placing two labels (irrespective of order) in the last two positions of the sequence instead of one in the last position, other labels being

Stellenbosch University https://scholar.sun.ac.za

selected at random. This would allow for the two labels to benefit from using all other labels for classification, at a cost of having $\frac{L(L-1)}{2}$ classifiers. Furthermore, LCC is designed to output importance values for each predictor, using the Gini index values. In future research, these values could be transformed to binary values to reflect variable selection based on the fitted RF model. The transformation could work for either local or global variable selection, with respect to the labels. This step would of course no longer influence the ML predictions, but it could be used as an inference tool.

Variable selection and variable importance were the main topics of Chapter 4. In this chapter, a theoretical justification for VS was provided, and an overview of VS approaches was given. The chapter further focused on using RF as a way of classifying ML data and simultaneously performing VS and variable importance inference. The RF classifier was used throughout the experimental work in this thesis and the Gini index values from the models were used as the variable importance values of LCC, according to the method described in Chapter 3.

An idea for future research involves an alternative RF selection of candidate predictors at tree splitting. In LCC, the trees usually use $\sqrt{P}$ of all predictors as candidates for tree splitting. Instead of choosing from all input and label predictors, the label predictors could act as fixed candidates. Therefore, the random selection of $\sqrt{P} - L$ inputs would involve only the $\boldsymbol{X}$ predictors. This way the classifier is likely to consider the label dependence to a larger extent, since it is forced to consider all the previous label predictors at each tree split.

In the benchmark data analysis of Chapter 5, the BR, CC, ECC and LCC classifiers were compared using three ML datasets. For each of the datasets, exploratory and confirmatory analyses of the results were performed. Overall, Friedman's test showed that there were significant differences between the classifiers, and the post-hoc tests confirmed that LCC was significantly better than the other classifiers.

In Chapter 6, the effect of five factors on the performance of the four classifiers was studied in a simulated setting. It was found that the data generation method used lead to an increased input signal as a result of increased label correlation. Therefore, the

Stellenbosch University https://scholar.sun.ac.za

effect of the ρ parameter could not be determined. Increasing N_{train}, P and the tuning parameter t, that distinguished the locally relevant and irrelevant predictors, lead to a better performance of the classifiers, while increasing L lead to deterioration. Friedman's test was applied in low and high correlation scenarios, for each evaluation measure separately. The LCC classifier outperformed the others in terms of five out of six measures, and did overall the best in both scenarios. The effect of the factors on the variable importance results was studied using heatmap analysis, and the proportion of correctly identified relevant predictors was summarised. The latter analysis could in future be computed for both input and label predictors.

In the last experimental chapter, the ML credit bureau dataset was analysed. The analysis was performed using only 15% of the predictors and the four classifiers were used to predict the labels. Overall, the ensemble methods outperformed BR and CC and LCC ranked the best of all the classifiers. The heatmap analysis provided a useful way of determining the important predictors for the labels.

The credit bureau dataset consisted of unevenly distributed labels, where five labels had label density below 1%. The splitting of such data can therefore be challenging and classifying such labels can be ineffective. Methods such as stratified sampling in an ML setting could be explored in order to keep the label densities approximately equal in both training and test sets. This idea is potentially complex, especially for a large number of labels.

Since the main goal of the experimental work was to compare the four classifiers, the tuning of the RF parameters was not essential. However, practical applications of LCC could involve tuning RF parameters, such as the number of candidate predictors, number of trees and the size of the trees. It could be interesting to explore how sensitive the ML results are to changes in these parameters.

Overall, ML classification was shown to be a complex and important task in statistical learning. The proposed LCC classifier was shown to be a useful and very competitive method for performing ML classification. The empirical evidence supported this claim. Besides having better performance than BR and CC and comparable performance to ECC, the LCC classifier allowed for ML inference regarding the relevance of predictors.

Stellenbosch University https://scholar.sun.ac.za

REFERENCES

Breiman, L. 2001. Random Forests. *Machine Learning*, 45(1): 5-32.

Charte, F. 2016. mldr.datasets: R Ultimate Multilabel Dataset Repository. R package version 0.3.15. https://CRAN.R-project.org/package=mldr.datasets.

Cheng, W. & Hüllermeier, E. 2009. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning,* 76(2-3): 211–225.

Cheng, W., Hüllermeier, E. & Dembczynski, K. 2010. Bayes optimal multilabel classification via probabilistic classifier chains, in *Proceedings of the 27th International Conference on Machine Learning*. Israel: 279–286.

Clare, A. & King, R.D. 2001. Knowledge discovery in multi-label phenotype data, in *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. Freiburg: 42–53.

Dekel, O. 2010. *Multiclass-multilabel classification with more labels than examples* [Online]. Available: http://videolectures.net/aistats2010_dekel_mmcw/ [2016, June 29].

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research,* 7:1-30.

Dunn, O.J. 1961. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293): 52–64.

Esuli, A., Fagni, T. & Sebastiani, F. 2008. Boosting multi-label hierarchical text categorization. *Information Retrieval*, 11 (4): 287–313.

Fan, J. & Lv, J. 2010, A selective overview of variable selection in high dimensional feature space. *Statistica Sinica,* 20(1): 101–148.

Friedman, M. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200): 675–701.

Stellenbosch University https://scholar.sun.ac.za

Fürnkranz, J., Hüllermeier, E., Mencía, E.L. & Brinker, K. 2008. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2): 133–153.

Godbole, S. & Sarawagi, S. 2004. Discriminative methods for multi-labeled classification, in *Proceedings of the 8th Pacific-Asia Conference, PAKDD*. Sydney: 22-30.

Hastie, T., Tibshirani, R. & Friedman, J. 2009. *The elements of statistical learning*. New York: Springer.

Holm, S. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2): 65–70.

Hong, C. 2014. *Multi-label classification* [Online]. Available: https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class22.pdf [2016, June 29].

Hüllermeier, E., Fürnkranz, J., Cheng, W. & Brinker, K. 2008. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16–17): 1897–1916.

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2013. *An introduction to statistical learning with applications in R*. New York: Springer.

Japkowicz, N. & Shah, M. 2014. *Evaluating Learning Algorithms: A Classification Perspective*. New York: Cambridge University Press.

Keikha, M. & Hashemi, S. 2016. Ordered classifier chains for multi-label classification. *Journal of Machine Intelligence,* 1(1): 7–12.

Kocev, D., Vens, C., Struyf, J. & Džeroski, S. 2007. Ensembles of Multi-Objective Decision Trees, in *Proceedings of the 18th European Conference on Machine Learning*, Warsaw: 624-631.

Kumar, A., Vembu, S., Menon, A.K. & Elkan, C. 2012. Learning and inference in probabilistic classifier chains with beam search, in *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases*. Bristol: 24–28.

Li, L., Liu, H., Ma, Z., Mo, Y., Duan, Z., Zhou, J. & Zhao, J. 2014. Multi-label

Stellenbosch University https://scholar.sun.ac.za

feature selection via information gain, in *Proceedings of Advanced Data Mining and Applications*. Guilin: 345–355.

Madjarov, G., Kocev, D., Gjorgjevikj, D. & Džeroski, S. 2012. An extensive experimental comparison of methods for multi-label learning, *Pattern Recognition* 45(9): 3084–3104.

Mencía, E.L., Park, S. & Fürnkranz, J. 2010. Efficient voting prediction for pairwise multilabel classification. *Neurocomputing* 73(7-9):1164–1176.

Oman, S.D. 2009. Easily simulated multivariate binary distributions with given positive and negative correlations. *Computational Statistics & Data Analysis*, 53(4): 999–1005.

Read, J. 2008. A pruned problem transformation method for multi-label classification, in *Proceedings of the New Zealand Computer Science Research Student Conference*. Christchurch.

Read, J., Pfahringer, B., Holmes, G. & Frank, E. 2011. Classifier chains for multi-label classification. *Machine Learning*, 85(3): 333–359.

Read, J. 2013. Multi-label classification [Online]. Available: https://users.ics.aalto.fi/jesse/talks/Multilabel-Part01.pdf [2016, June 29].

Read, J., Martino, L. & Luengo, D. 2013. Efficient monte carlo optimization for multi-label classifier chains, in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing*. Vancouver: 3457–3461.

Sandrock, T. & Steel, S.J. 2016. Probe variables for multi-label variable selection. Technical report, University of Stellenbosch.

Schapire, R.E. & Singer, Y. 2000. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3): 135–68.

Shao, H., Li, G., Liu, G. & Wang, Y. 2013. Symptom selection for multi-label data of inquiry diagnosis in traditional Chinese medicine, *Science China Information Sciences*, 56(5): 1–13.

Stellenbosch University https://scholar.sun.ac.za

Siddiqi, N. 2006. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. New Jersey: John Wiley & Sons, Inc.

Spolaôr, N., Cherman, E.A., Monard, M.C. & 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.

Tai, F. & Lin, H. 2012. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9): 2508–2542.

Tsoumakas, G. & Vlahavas, I. 2007. Random k-Labelsets: An Ensemble Method for Multilabel Classification, in *Proceedings of the 18th European Conference on Machine Learning*. Warsaw: 406–417.

Tsoumakas, G., Katakis, I., & Vlahavas, I. 2008. Effective and efficient multilabel classification in domains with large number of labels, in *Proceedings of the ECML/PKDD Workshop on Mining Multidimensional Data*. Antwerp: 30–44.

Tsoumakas, G., Katakis, I. & Vlahavas, I. 2010. Mining multi-label data, in Mailom, O. & Rokach, L. (eds.). *Data Mining and Knowledge Discovery Handbook*, Springer. 667–685.

Zhang, M. & Zhou, Z. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7): 2038-2048.

Zhang, M., Peña, J.M. & Robles, V. 2009. Feature selection for multi-label naive Bayes classification, *Information Sciences Journal*, 179(19): 3218–3229.

Zhang, M. & Zhang, K. 2010. Multi-label learning by exploiting label dependency, in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington: 999-1008.

Zhang, Y. & Schneider, J. 2011. Multi-Label Output Codes using Canonical Correlation Analysis. *Journal of Machine Learning Research*, 15: 873–882.

Stellenbosch University https://scholar.sun.ac.za

APPENDIX A: Chapter 3

A.1 IMPORTANCE OF THE LABEL SEQUENCE ORDER

```
#######################################################
# CLASSIFIER CHAINS classification using L! sequences #
#######################################################

# The following code performs ML classification using all
L!
# label sequence permutations applied to the CC
algorithm.
CC_all_permut_function <- function(dataset,P,L,ss)
{
  library(randomForest)
  N <- nrow(dataset)
  set.seed(ss)
  train_index <- sample(N,floor(N*0.75),replace=FALSE)
  data_train <- dataset[train_index,]
  data_test <- dataset[-train_index,]
  xtrain <- data_train[,1:P]; ytrain <-
   data_train[,(P+1):(P+L)]
  xtest <- data_test[,1:P]; ytest <-
   data_test[,(P+1):(P+L)]
  ntrain <- nrow(xtrain)
  ntest <- nrow(xtest)

  # All of the L! label sequences are stored in PERM.
  library(gtools)
  PERM <- gtools::permutations(L,L,1:L)
  HL <- matrix(0,nrow=nrow(PERM),ncol=2)
  for (s in 1:nrow(PERM))
  {
    sequ <- PERM[s,]
    ypred <- matrix(1,nrow=ntest,ncol=L)
    xtrain_new <- xtrain
    xtest_new <- xtest
    # Classifier chain algorithm.
    for (i in 1:L)
    {
      xytrain <-
      data.frame(xtrain_new,y=as.factor(ytrain[,sequ[i]])
      )
      rffit <- randomForest(y~., data=xytrain, ntree=50)
      ypred[,sequ[i]] <-
      as.numeric(predict(rffit,xtest_new))-1
      xtest_new[,paste0("label_",sequ[i])] <-
      ypred[,sequ[i]]
```

Stellenbosch University https://scholar.sun.ac.za

```
      xtrain_new[,paste0("label_",sequ[i])] <-
      ytrain[,sequ[i]]
    }
    HL[s,1] <- round(as.numeric(paste(sequ, collapse =
    '')),0)
    HL[s,2] <-
    round(measures.norank(ylabels=ytest,zlabels=ypred)[[1
    ]],3)
  }
  HL_mat <- HL[order(HL[,2]),]
  #list(HL_mat)}
  hist(HL_mat[,2],main=expression(paste('Distribution of
HL')),
       xlab=expression(paste('HL
values')),cex.main=2,cex.lab=1.5,freq =
TRUE,col=brewer.pal(3,"Greens")[2])}
```

A.2 COMPARISON OF CLASSIFICATION METHODS

```
####################
# BINARY RELEVANCE #
####################

# The following function performs BR of ML dataset
BR_fun_parall <-
function(P,L,train_data,test_data,notree)
{
  library(ranger)
  library(randomForest)
  xtrain <- train_data[,1:P]; ytrain <-
train_data[,(P+1):(P+L)]
  xtest <- test_data[,1:P]; ytest <-
test_data[,(P+1):(P+L)]
  ntrain <- nrow(xtrain)
  ntest <- nrow(xtest)
  ypred <- matrix(1,nrow=ntest,ncol=L)

  # The following code implements BR algorithm using RF
  for (i in 1:L)
  {
    xytrain <- data.frame(xtrain,y=as.factor(ytrain[,i]))
    rffit <- ranger(y~., data=xytrain, write.forest=T,
num.trees=notree)
    ypred[,i] <-
as.numeric(predictions(predict(rffit,data=xtest)))-1
  }
  r <- measures.norank(ylabels=ytest,zlabels=ypred)
  list(r=r,ylabels=ytest,zlabels=ypred)
}
```

Stellenbosch University https://scholar.sun.ac.za

```
#####################
# CLASSIFIER CHAINS #
#####################

# The following function performs CC of ML dataset
CC_fun_parall <-
function(P,L,train_data,test_data,notree)
{
  library(ranger)
  library(randomForest)
  xtrain <- train_data[,1:P]; ytrain <-
  train_data[,(P+1):(P+L)]
  xtest <- test_data[,1:P]; ytest <-
  test_data[,(P+1):(P+L)]
  ntrain <- nrow(xtrain)
  ntest <- nrow(xtest)
  ypred <- matrix(1,nrow=ntest,ncol=L)

  # The following code implements CC algorithm using RF
  for (i in 1:L)
  {
    xytrain <- data.frame(xtrain,y=as.factor(ytrain[,i]))
    rffit <- ranger(y~., data=xytrain, write.forest=T,
    num.trees=notree)
    ypred[,i] <-
    as.numeric(predictions(predict(rffit,data=xtest)))-1
    xtest[,paste0("label_",i)] <- ypred[,i]
    xtrain[,paste0("label_",i)] <- ytrain[,i]
  }
  r <- measures.norank(ylabels=ytest,zlabels=ypred)
  r
}
```

The 1CC methods use the following R Codes. Every 1CC method uses different values of '*sequ*', as described below:

```
#################
# 1CC FUNCTIONS #
#################

CC_funXYZ <- function(P,L,train_data,test_data)
{
  library(randomForest)
  xtrain <- train_data[,1:P]; ytrain <-
  train_data[,(P+1):(P+L)]
  xtest <- test_data[,1:P]; ytest <-
  test_data[,(P+1):(P+L)]
  ntrain <- nrow(xtrain)
  ntest <- nrow(xtest)
  ypred <- matrix(1,nrow=ntest,ncol=L)
```

Stellenbosch University https://scholar.sun.ac.za

```
  ############################################
  # METHOD FOR COMPUTING THE LABEL SEQUENCE #
  ############################################
  sequ <- *

  # CC algorithm.
  for (i in 1:L)
  {
      xytrain <-
      data.frame(xtrain,y=as.factor(ytrain[,sequ[i]]))
      rffit <- randomForest(y~., data=xytrain, ntree=200)
      ypred[,sequ[i]] <-
      as.numeric(predict(rffit,xtest))-1
      xtest[,paste0("label_",sequ[i])] <- ypred[,sequ[i]]
      xtrain[,paste0("label_",sequ[i])] <-
      ytrain[,sequ[i]]
  }
  r <- measures.norank(ylabels=ytest,zlabels=ypred);r
}
```

* Methods for computing the '*sequ*' are given below:

```
    ######################
    # METHOD 1 using COR #
    ######################
    R <- abs(cor(ytrain))
    S <- matrix(0,ncol=L,nrow=1)
    for (i in 1:L)
      S[,i] <- mean(R[i,-i])
    colnames(S) <- paste0(1:L)
    sequ <- as.numeric(rownames(as.matrix(S[,order(S)])))
```

```
    #####################
    # METHOD 2 using RF #
    #####################
    I <- matrix(0,ncol=L,nrow=1)
    colnames(I) <- paste0(1:L)
    for (i in 1:L)
    {
     r <-
      randomForest(y=as.factor(ytrain[,i]),x=ytrain[,i],i
      mportance=TRUE)
     I[i] <- mean(importance(r)[,4])
    }
    sequ <- as.numeric(rownames(as.matrix(I[,order(I)])))
```

Stellenbosch University https://scholar.sun.ac.za

```
################
# METHOD 3 OCC #
################
Accuracy <- matrix(0,ncol=L,nrow=1)
colnames(Accuracy) <- paste0(1:L)
for (l in 1:L)
{
 rfmod <-
 randomForest(y=as.factor(ytrain[,l]),x=xtrain,mtry=f
 loor(sqrt(P)))
 Accuracy[,l] <- sum(ytrain[,l] ==
 rfmod$predicted)/ntrain
}
sequ <-
as.numeric(rownames(as.matrix(Accuracy[,order(Accurac
y,decreasing=T)])))
```

```
####################
# METHOD 4 ReliefF #
####################
rFvalue <- matrix(0,ncol=L,nrow=1)
colnames(rFvalue) <- paste0(1:L)
for (l in 1:L)
{
  rF <- attrEval(ytrain[,l]~.,
  data=as.data.frame(ytrain),
  estimator="ReliefFexpRank")
  rFvalue[,l] <- sum(abs(rF[-l]))
}
sequ <-
as.numeric(rownames(as.matrix(rFvalue[,order(rFvalue,d
ecreasing=F)])))
```

```
################
# METHOD 5 CCA #
################
cca <-
matrix(abs(cancor(x=xtrain,y=ytrain)$ycoef[,1]),ncol=
L,nrow=1,byrow=T)
colnames(cca) <- paste0(1:L)
sequ <-
as.numeric(rownames(as.matrix(cca[,order(cca,decreasi
ng=T)])))
```

Stellenbosch University https://scholar.sun.ac.za

```
#####################################################
# 1CC METHODS COMPARISON ANALYSIS USING BR, CC, 1CC #
#####################################################

# The following function compares the BR, CC and five 1CC
algorithms based on 30 random splits of the emotions
dataset.
EMP_1cc <-
function(Nsim=30,P=72,L=6,dataset=emotions_data,notree=20
0)
{
start.time <- Sys.time()
# Storage matrices
N <- nrow(dataset)
BR_R          <- matrix(0,ncol=6,nrow=Nsim+2)
CC_R          <- matrix(0,ncol=6,nrow=Nsim+2)
CC_cor_R      <- matrix(0,ncol=6,nrow=Nsim+2)
CC_rfimp_R    <- matrix(0,ncol=6,nrow=Nsim+2)
CC_occ_R      <- matrix(0,ncol=6,nrow=Nsim+2)
CC_reliefF_R <- matrix(0,ncol=6,nrow=Nsim+2)
CC_cancor_R   <- matrix(0,ncol=6,nrow=Nsim+2)

# Analysis performed over 30 random splits of the
emotions dataset.
for (m in 1:Nsim)
{
  set.seed(m)
  train_index <- sample(N,floor(N*0.75),replace=FALSE)
  train_data <- dataset[train_index,]
  test_data <- dataset[-train_index,]

  BR_R[m,] <-
  unlist(BR_fun(P,L,train_data,test_data,notree)[c(1,2,3,
  4,6,7)])
  CC_R[m,] <-
  unlist(CC_fun(P,L,train_data,test_data,notree)[c(1,2,3,
  4,6,7)])
  CC_cor_R[m,] <-
  unlist(CC_cor_fun(P,L,train_data,test_data)[c(1,2,3,4,6
  ,7)])
  CC_rfimp_R[m,] <-
  unlist(CC_rfimp_fun(P,L,train_data,test_data)[c(1,2,3,4
  ,6,7)])
  CC_occ_R[m,] <-
  unlist(CC_occ_fun(P,L,train_data,test_data)[c(1,2,3,4,6
  ,7)])
  CC_reliefF_R[m,] <-
  unlist(CC_relieff_fun(P,L,train_data,test_data)[c(1,2,3
  ,4,6,7)])
```

Stellenbosch University https://scholar.sun.ac.za

```
  CC_cancor_R[m,] <-
  unlist(CC_cancor_fun(P,L,train_data,test_data)[c(1,2,3,
  4,6,7)])
}

# Mean and std.dev. computations.
BR_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(BR_R[1:Nsim,],2,mean),apply(BR_R[1:Nsim,],
2,sd)),ncol=6,byrow=T)
CC_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_R[1:Nsim,],2,mean),apply(CC_R[1:Nsim,],
2,sd)),ncol=6,byrow=T)
CC_cor_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_cor_R[1:Nsim,],2,mean),apply(CC_cor_R[1
:Nsim,],2,sd)),ncol=6,byrow=T)
CC_rfimp_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_rfimp_R[1:Nsim,],2,mean),apply(CC_rfimp
_R[1:Nsim,],2,sd)),ncol=6,byrow=T)
CC_occ_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_occ_R[1:Nsim,],2,mean),apply(CC_occ_R[1
:Nsim,],2,sd)),ncol=6,byrow=T)
CC_reliefF_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_reliefF_R[1:Nsim,],2,mean),apply(CC_rel
iefF_R[1:Nsim,],2,sd)),ncol=6,byrow=T)
CC_cancor_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_cancor_R[1:Nsim,],2,mean),apply(CC_canc
or_R[1:Nsim,],2,sd)),ncol=6,byrow=T)

# Final presentation of results.
r_mean <-
round(rbind(BR_R[(Nsim+1),],CC_R[(Nsim+1),],CC_cor_R[(Nsi
m+1),],CC_rfimp_R[(Nsim+1),],

CC_reliefF_R[(Nsim+1),],CC_occ_R[(Nsim+1),],CC_cancor_R[(
Nsim+1),]),3)
r_sd <-
round(rbind(BR_R[(Nsim+2),],CC_R[(Nsim+2),],CC_cor_R[(Nsi
m+2),],CC_rfimp_R[(Nsim+2),],
CC_reliefF_R[(Nsim+2),],CC_occ_R[(Nsim+2),],CC_cancor_R[(
Nsim+2),]),3)
rownames(r_mean) <-
c("BR","CC","CC_cor","CC_rfimp","CC_relieff","CC_occ","CC
_cancor")
colnames(r_mean) <- c("HL","CL","PR","RE","F2","AC")
rownames(r_sd) <-
c("BR","CC","CC_cor","CC_rfimp","CC_relieff","CC_occ","CC
_cancor")
colnames(r_sd) <- c("HL","CL","PR","RE","F2","AC")
end.time <- Sys.time()
t.time <- end.time-start.time
list(t.time=t.time,result_mean=r_mean,result_sd=r_sd)}
```

Stellenbosch University https://scholar.sun.ac.za

APPENDIX B: Chapter 5

B.1 BENCHMARK ANALYSIS: CLASSIFICATION

```
#######################
# L-CLASSIFIER CHAINS #
#######################

# The following function performs LCC of ML dataset
LCC_fun <- function(P,L,train_data,test_data,notree)
{
  library(ranger)
  library(randomForest)
  xtrain <- train_data[,1:P]; ytrain <-
  train_data[,(P+1):(P+L)]
  xtest <- test_data[,1:P]; ytest <-
  test_data[,(P+1):(P+L)]
  ntrain <- nrow(xtrain)
  ntest <- nrow(xtest)

  # Label cardinality of training data
  LC_orig <- mean(apply(ytrain,1,sum))

  # Matrix W for storing votes for each label prediction
  W <- matrix(0,ncol=L,nrow=ntest)
  A_final <- matrix(0,nrow=L,ncol=L+P)
  colnames(A_final) <-
  c(paste0("X",1:P),paste0("label",1:L));
  rownames(A_final) <- paste0("label",1:L)

  # The following code implements LCC algorithm
  for (cc in 1:L)
  {
    ypred <- matrix(1,nrow=ntest,ncol=L)
    sequ <- sample(seq(1:L)[-cc])
    sequ[L] <- cc
    xtrain_new <- xtrain
    xtest_new <- xtest
    A <- matrix(0,ncol=P+L,nrow=L)
    colnames(A) <- c(paste0("X",1:P),sequ); rownames(A)
    <- sequ

    # The following code implements CC algorithm using RF
    for (i in 1:L)
    {
      xytrain <-
      data.frame(xtrain_new,y=as.factor(ytrain[,sequ[i]])
      )
```

Stellenbosch University https://scholar.sun.ac.za

```
      rffit <- ranger(y ~ ., data = xytrain, importance =
      "impurity", write.forest=T, num.trees=notree)
      A[i,1:(P+i-1)] <- rffit$variable.importance[1:(P+i-
      1)]
      ypred[,sequ[i]] <-
      as.numeric(predictions(predict(rffit, data =
      xtest_new)))-1
      xtest_new[,paste0("label_",sequ[i])] <-
      ypred[,sequ[i]]
      xtrain_new[,paste0("label_",sequ[i])] <-
      ytrain[,sequ[i]]
    }

    W <- W + ypred

    # Storing of importance values
    A[,(P+1):(P+L)] <-
    A[,P+(order(colnames(A)[(P+1):(P+L)]))]
    colnames(A)[(P+1):(P+L)] <- 1:L

    for (j in 1:L)
    {A_final[j,1:P] <- A_final[j,1:P] +
    A[which(sequ==j),1:P]}
    A_final[cc,(P+1):(P+L)] <- A[L,(P+1):(P+L)]
    }

  # Standardising of importance values in matrix A_final.
  A_final[,1:P] <- A_final[,1:P]/L

  # Threshold computation for determining final
  prediction for a given m.
  LC_t <- matrix(0,ncol=2,nrow=L); colnames(LC_t) <-
  c("LC_t","LC_orig-LC-t")
  for (q in 1:L)
  {
    W2 <- ifelse(W >= q, 1, 0)
    LC_t[q,1] <- mean(apply(W2,1,sum))
  }
  LC_t[,2]  <- abs(LC_t[,1]-LC_orig)
  threshold <- which(LC_t[,2]==min(LC_t[,2]))[1]

  W3 <- ifelse(W >= threshold, 1, 0)
  r <- measures.norank(ylabels=ytest,zlabels=W3)
  W_prob <- W/L
  list(r=r, W_prob=W_prob, Y_true=ytest,
  threshold=threshold)
}
```

Stellenbosch University https://scholar.sun.ac.za

```
##############################
# ENSEMBLE CLASSIFIER CHAINS #
##############################

# The following function performs ECC of ML dataset
ECC_fun_parall <-
      function(P,L,train_data,test_data,notree)
{
  library(ranger)
  library(randomForest)
  xtrain <- train_data[,1:P]; ytrain <-
      train_data[,(P+1):(P+L)]
  xtest <- test_data[,1:P]; ytest <-
      test_data[,(P+1):(P+L)]
  ntrain <- nrow(xtrain)
  ntest <- nrow(xtest)
  B_num <- L

  # Label cardinality of training data
  LC_orig <- mean(apply(ytrain,1,sum))

  # Matrix W for storing votes for each label prediction
  W <- matrix(0,ncol=L,nrow=ntest)

  # The following code implements ECC algorithm
  for (B in 1:B_num)
  {
    ypred <- matrix(1,nrow=ntest,ncol=L)
    sequ <- sample(1:L,L)
    ind <-sample(ntrain,floor(ntrain*0.67),replace=FALSE)
    xtrain_new <- xtrain[ind,]
    ytrain_new <- ytrain[ind,]
    xtest_new <- xtest

    # The following code implements CC algorithm using RF
    for (i in 1:L)
    {
      xytrain <-
      data.frame(xtrain_new,y=as.factor(ytrain_new[,sequ[
      i]]))
      rffit <- ranger(y~.,data=xytrain, write.forest=T,
      num.trees=notree)
      ypred[,sequ[i]] <-
      as.numeric(predictions(predict(rffit, data =
      xtest_new)))-1
      xtest_new[,paste0("label_",sequ[i])] <-
      ypred[,sequ[i]]
      xtrain_new[,paste0("label_",sequ[i])] <-
      ytrain_new[,sequ[i]]
    }
    W <- W + ypred
```

Stellenbosch University https://scholar.sun.ac.za

```
  }
  # Threshold computation for determining final
      prediction for a given m.
  LC_t <- matrix(0,ncol=2,nrow=B_num); colnames(LC_t) <-
      c("LC_t","LC_orig-LC-t")
  for (q in 1:B_num)
  {
    W2 <- ifelse(W >= q, 1, 0)
    LC_t[q,1] <- mean(apply(W2,1,sum))
  }
  LC_t[,2]  <- abs(LC_t[,1]-LC_orig)
  threshold <- which(LC_t[,2]==min(LC_t[,2]))[1]

  W3 <- ifelse(W >= threshold, 1, 0)
  r <- measures.norank(ylabels=ytest,zlabels=W3)
  r
}
```

```
#########################################################
# BENCHMARK DATASETS ANALYSIS USING BR, CC, LCC and ECC #
#########################################################

# The following function fits four classifiers to a given
ML benchmark dataset. The results are based on 30 random
splits of the datasets.
r_bench <- function(Nsim,P,L,dataset,notree)
{
start.time <- Sys.time()

# Initiation of storage matrices.
A_global <- matrix(0,nrow=L,ncol=L+P)
N <- nrow(dataset)

BR_R  <- matrix(0,ncol=7,nrow=Nsim+2);colnames(BR_R) <-
c("HL","CL","PR","RE","F1","F2","AC"); rownames(BR_R) <-
c(1:Nsim,"BR_mean","BR_sd")
CC_R  <- matrix(0,ncol=7,nrow=Nsim+2);colnames(CC_R) <-
c("HL","CL","PR","RE","F1","F2","AC"); rownames(CC_R) <-
c(1:Nsim,"CC_mean","CC_sd")
LCC_R <- matrix(0,ncol=7,nrow=Nsim+2);colnames(LCC_R) <-
c("HL","CL","PR","RE","F1","F2","AC"); rownames(LCC_R) <-
c(1:Nsim,"LCC_mean","LCC_sd")
ECC_R <- matrix(0,ncol=7,nrow=Nsim+2);colnames(ECC_R) <-
c("HL","CL","PR","RE","F1","F2","AC"); rownames(ECC_R) <-
c(1:Nsim,"ECC_mean","ECC_sd")

# Analysis performed over 30 random splits of the
datasets.
for (m in 1:Nsim)
{
  set.seed(m)
```

Stellenbosch University https://scholar.sun.ac.za

```
  train_index <- sample(N,floor(N*0.75),replace=FALSE)
  train_data <- dataset[train_index,]
  test_data <- dataset[-train_index,]

  BR_R[m,] <-
  unlist(BR_fun(P,L,train_data,test_data,notree)[1:7])
  CC_R[m,] <-
  unlist(CC_fun(P,L,train_data,test_data,notree)[1:7])
  ECC_R[m,] <-
  unlist(ECC_fun(P,L,train_data,test_data,B_num=L,notree)
  [1:7])
  LCC_f <- LCC_fun(P,L,train_data,test_data,notree)
  LCC_R[m,] <- unlist(LCC_f$r[1:7])
  A_global <- A_global + LCC_f$A_final
}

# Mean and std.dev. computations.
BR_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(BR_R[1:Nsim,],2,mean),apply(BR_R[1:Nsim,],
2,sd)),ncol=7,byrow=T)
CC_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(CC_R[1:Nsim,],2,mean),apply(CC_R[1:Nsim,],
2,sd)),ncol=7,byrow=T)
LCC_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(LCC_R[1:Nsim,],2,mean),apply(LCC_R[1:Nsim,
],2,sd)),ncol=7,byrow=T)
ECC_R[(Nsim+1):(Nsim+2),] <-
matrix(c(apply(ECC_R[1:Nsim,],2,mean),apply(ECC_R[1:Nsim,
],2,sd)),ncol=7,byrow=T)

# Final presentation of results.
r_mean <-
round(rbind(BR_R[(Nsim+1),],CC_R[(Nsim+1),],LCC_R[(Nsim+1
),],ECC_R[(Nsim+1),]),3)
r_sd <-
round(rbind(BR_R[(Nsim+2),],CC_R[(Nsim+2),],LCC_R[(Nsim+2
),],ECC_R[(Nsim+2),]),3)

rownames(r_mean) <- c("BR","CC","LCC","ECC")
colnames(r_mean) <- c("HL","CL","PR","RE","F1","F2","AC")

rownames(r_sd) <- c("BR","CC","LCC","ECC")
colnames(r_sd) <- c("HL","CL","PR","RE","F1","F2","AC")

# Standardisation of the LCC importance matrix.
A_global2 <- A_global/Nsim
A_global_std <- matrix(0,ncol=P+L,nrow=L)
colnames(A_global_std) <-
c(paste0("X",1:P),paste0("label",1:L));
rownames(A_global_std) <- paste0("label",1:L)
for (k in 1:L)
```

Stellenbosch University https://scholar.sun.ac.za

```
  A_global_std[k,] <-
round((A_global2[k,]/max(A_global2[k,])*100),1)

end.time <- Sys.time()
t.time <- end.time-start.time
list(BR=round(BR_R,3),CC=round(CC_R,3),LCC=round(LCC_R,3)
,ECC=round(ECC_R,3),r_mean=r_mean,r_sd=r_sd,t.time=t.time
,A_global_std=A_global_std)
}
```

B.2 BENCHMARK ANALYSIS: EXPLORATORY ANALYSIS

```
#####################################################
# BENCHMARK DATASETS EXPLORATORY ANALYSIS OF RESULTS #
#####################################################

# The following code splits the results of the benchmark
data analysis, according to the four classifiers.
library(RColorBrewer)
bench_data <- BENCH_emotions
L=6;P=72

bench_data <- BENCH_scene
L=6;P=294

bench_data <- BENCH_yeast
L=14;P=103
BR  <- bench_data$BR[-c(31,32),c(1,2,3,4,6,7)]
CC  <- bench_data$CC[-c(31,32),c(1,2,3,4,6,7)]
LCC <- bench_data$LCC[-c(31,32),c(1,2,3,4,6,7)]
ECC <- bench_data$ECC[-c(31,32),c(1,2,3,4,6,7)]
```

```
############
# BOXPLOTS #
############

# This code creates boxplots of the results for four
classifiers.
par(mfrow=c(3,2),oma = c(0, 0, 2, 0))
lab <- c('Hamming loss (HL)','Classification accuracy
(CL)', 'Precision (PR)','Recall (RE)','F2','Accuracy
(AC)')
lab_main <- c('Hamming loss','Classification accuracy',
'Precision','Recall','F2','Accuracy')
for (i in 1:6)
{
  boxplot(BR[,i],CC[,i],ECC[,i],LCC[,i],names=c('BR','CC'
  ,'ECC','LCC'),ylab=lab[i],main=lab_main[i],col=brewer.p
  al(4,"Greens"))
  grid()
```

Stellenbosch University https://scholar.sun.ac.za

```
}
mtext('Distribution of ML evaluation measures by method:
Yeast data', outer = TRUE, cex = 1.2)
```

```
#################################
# VARIABLE IMPORTANCE DENSITIES #
#################################

# This code plots variable importance distributions for
every label.
VI <- bench_data$A_global_std
all_y <- NULL
for (j in 1:L) all_y <- c(all_y,density(VI[j,])$y)
ymax <- max(all_y)

# For EMOTIONS and SCENE datasets.
par(mfrow=c(3,2),oma = c(0, 0, 2, 0))
for (i in 1:L)
{
  plot(density(VI[i,]), main=paste0('Label
   ',i),xlab="Importance
   index",col="darkcyan",lwd=2,xlim=c(0,100),ylim=c(0,yma
   x))
  arrows(x0=100, y0=0, x1=100, y1=ymax/2, col =
   "orange",lwd=1,length = 0.1)
   text(x=98,y=ymax/1.75,labels=names(VI[i,][VI[i,]==max(
   VI[i,])]),font=2)
}
mtext('Distribution of variable importance values: Scene
data', outer = TRUE, cex = 1.2)

# For YEAST dataset.
par(mfrow=c(2,2),oma = c(0, 0, 2, 0))
for (i in c(1,3,5,7))
{
  plot(density(VI[i,]), main=paste0("Labels ",i,' and
  ',i+1),xlab="Importance
  index",col="darkcyan",lwd=2,xlim=c(0,100),ylim=c(0,ymax
  ))
  lines(density(VI[i+1,]),xlab='Importance
  index',col='darkorange',lwd=2,lty=2)
  legend('bottom', legend=c(paste0('label ',i,'
  '),paste0('label ',i+1)),xpd=T,inset=c(0,-
  0.25),horiz=T,lty=c(1,2),border="white",col=c('darkcyan
  ','darkorange'),bty = "n")
  arrows(x0=100, y0=0, x1=100, y1=ymax/2, col =
  "black",lwd=1,length = 0.1)
   text(x=94,y=ymax/1.5,labels=names(VI[i,][VI[i,]==max(V
   I[i,])]),col="darkcyan",font=2)
   text(x=94,y=ymax/1.75,labels=names(VI[i+1,][VI[i+1,]==
   max(VI[i+1,])]),col="darkorange",font=2)
```

Stellenbosch University https://scholar.sun.ac.za

```
}
mtext('Distribution of variable importance values: Yeast
data', outer = TRUE, cex = 1.2)
```

```
################################
# VARIABLE IMPORTANCE HEATMAPS #
################################

# This code plots heatmaps of variable importance values.
library(gplots)
VI <- bench_data$A_global_std
tiff("Y_heat.tiff", height = 20, width = 20, units =
'cm',compression = "lzw", res = 300)
colo <- colorRampPalette(c("white","gold","red","black"))
VI <- bench_data$A_global_std
nam <- rep("", nrow(t(VI)))
nam[seq(1,nrow(t(VI)), 2)] <-
rownames(t(VI))[seq(1,nrow(t(VI)), 2)]
nam[seq(1,nrow(t(VI)), 7)] <-
rownames(t(VI))[seq(1,nrow(t(VI)), 7)]
nam[seq(1,nrow(t(VI)), 3)] <-
rownames(t(VI))[seq(1,nrow(t(VI)), 3)]
colnames(VI) <- nam
heatmap.2(t(VI),  Rowv=F,
dendrogram="column",keysize=1,main="Yeast data: heatmap
of importance values",trace="none",col = colo(L*(P+L))))
```

Stellenbosch University https://scholar.sun.ac.za

APPENDIX C: Chapter 6

C.1 SIMULATION ANALYSIS: CLASSIFICATION

```
#################################################
# SIMULATION ANALYSIS USING BR, CC, LCC and ECC #
#################################################

# The following function performs ML classification on
# simulated datasets, aggregated over 30 simulation runs.
r_sim_cl <-
function(ntrain,P,L,rho,t,ntest=1000,Nsim=30,densvek=c(re
p(0.3,L)),v=c(0.8,0.2),notree=50)
{
  start.time <- Sys.time()
  # Storage matrices
  A_global <- matrix(0,nrow=L,ncol=L+P)

  BR_R <- matrix(0,ncol=6,nrow=Nsim+2);colnames(BR_R) <-
  c("HL","CL","PR","RE","F2","AC"); rownames(BR_R) <-
  c(1:Nsim,"BR_mean","BR_sd")
  CC_R <- matrix(0,ncol=6,nrow=Nsim+2);colnames(CC_R) <-
  c("HL","CL","PR","RE","F2","AC"); rownames(CC_R) <-
  c(1:Nsim,"CC_mean","CC_sd")
  LCC_R <- matrix(0,ncol=6,nrow=Nsim+2);colnames(LCC_R)
  <- c("HL","CL","PR","RE","F2","AC"); rownames(LCC_R) <-
  c(1:Nsim,"LCC_mean","LCC_sd")
  ECC_R <- matrix(0,ncol=6,nrow=Nsim+2);colnames(ECC_R)
  <- c("HL","CL","PR","RE","F2","AC"); rownames(ECC_R) <-
  c(1:Nsim,"ECC_mean","ECC_sd")

  # True variable importance matrix.
  Amat <- matrix(0,ncol=P,nrow=L);rownames(Amat) <-
  paste0("label",1:L);colnames(Amat) <- paste0("X",1:P)
  for(a in 1:L)   Amat[a,] <-
  sample(c(0,1),size=P,replace=TRUE,prob=impvek)

  # Simulation runs.
  for (m in 1:Nsim)
  {
    train_data <-
    data_gen(N=ntrain,P=P,densvek=densvek,Amat=t(Amat),rh
    o=rho,L=L,t=t)
    test_data  <- data_gen(N=ntest,
    P=P,densvek=densvek,Amat=t(Amat),rho=rho,L=L,t=t)
```

Stellenbosch University https://scholar.sun.ac.za

```
    BR_R[m,] <-
    unlist(BR_fun(P,L,train_data,test_data,notree)[c(1,2,
    3,4,6,7)])
    CC_R[m,] <-
    unlist(CC_fun(P,L,train_data,test_data,notree)[c(1,2,
    3,4,6,7)])
    ECC_R[m,] <-
    unlist(ECC_fun(P,L,train_data,test_data,B_num=L,notre
    e)[c(1,2,3,4,6,7)])
    LCC_f <- LCC_fun(P,L,train_data,test_data,notree)
    LCC_R[m,] <- unlist(LCC_f$r[c(1,2,3,4,6,7)])
    A_global <- A_global + LCC_f$A_final
  }

  # Mean and std.dev. computations.
  BR_R[(Nsim+1):(Nsim+2),] <-
  matrix(c(apply(BR_R[1:Nsim,],2,mean),apply(BR_R[1:Nsim,
  ],2,sd)),ncol=6,byrow=T)
  CC_R[(Nsim+1):(Nsim+2),] <-
  matrix(c(apply(CC_R[1:Nsim,],2,mean),apply(CC_R[1:Nsim,
  ],2,sd)),ncol=6,byrow=T)
  LCC_R[(Nsim+1):(Nsim+2),] <-
  matrix(c(apply(LCC_R[1:Nsim,],2,mean),apply(LCC_R[1:Nsi
  m,],2,sd)),ncol=6,byrow=T)
  ECC_R[(Nsim+1):(Nsim+2),] <-
  matrix(c(apply(ECC_R[1:Nsim,],2,mean),apply(ECC_R[1:Nsi
  m,],2,sd)),ncol=6,byrow=T)

  # Final presentation of results.
  r_mean <-
  round(rbind(BR_R[(Nsim+1),],CC_R[(Nsim+1),],LCC_R[(Nsim
  +1),],ECC_R[(Nsim+1),]),3)
  r_sd <-
  round(rbind(BR_R[(Nsim+2),],CC_R[(Nsim+2),],LCC_R[(Nsim
  +2),],ECC_R[(Nsim+2),]),3)

  rownames(r_mean) <- c("BR","CC","LCC","ECC")
  colnames(r_mean) <- c("HL","CL","PR","RE","F2","AC")

  rownames(r_sd) <- c("BR","CC","LCC","ECC")
  colnames(r_sd) <- c("HL","CL","PR","RE","F2","AC")

  # The following code computes the resulting output from
  the    # analysis.
  A_global2 <- A_global/Nsim
  A_global_std <- matrix(0,ncol=P+L,nrow=L)
  colnames(A_global_std) <-
  c(paste0("X",1:P),paste0("label",1:L));
  rownames(A_global_std) <- paste0("label",1:L)
  for (k in 1:L)
```

Stellenbosch University https://scholar.sun.ac.za

```
    A_global_std[k,] <-
  round(((A_global2[k,]/max(A_global2[k,]))*100),1)

  # Computation of proportion of correctly identified
  relevant variables.
  I1 <-
  apply(A_global_std[,1:P],1,function(x)names(tail(sort(x
  ),0.2*P)))
  I2 <- apply(Amat,1,function(x)names(which(x==1)))
  Icomp <- matrix(0,ncol=L,nrow=0.2*P)
  for (i in 1:L)
    Icomp[,i] <- I1[,i] %in% I2[[i]]
  VIprop <- sum(Icomp)/(0.2*P*L)
  end.time <- Sys.time()
  t.time <- end.time-start.time

  list(ntrain=ntrain,P=P,L=L,rho=rho,t=t,BR=BR_R,CC=CC_R,
  LCC=LCC_R,ECC=ECC_R,
  r_mean=r_mean,r_sd=r_sd,t.time=t.time,Amat=Amat,A_globa
  l_std=A_global_std,VIprop=VIprop)
}
```

C.2 SIMULATION ANALYSIS: EXPLORATORY ANALYSIS

```
############
# BOXPLOTS #
############

library(RColorBrewer)
bench_data <- SIM_13m
bench_data2 <- SIM_14m
BR  <- bench_data$BR[-c(31,32),];  BR2  <-
bench_data2$BR[-c(31,32),]
CC  <- bench_data$CC[-c(31,32),];  CC2  <-
bench_data2$CC[-c(31,32),]
LCC <- bench_data$LCC[-c(31,32),]; LCC2 <-
bench_data2$LCC[-c(31,32),]
ECC <- bench_data$ECC[-c(31,32),]; ECC2 <-
bench_data2$ECC[-c(31,32),]

par(mfrow=c(3,2),oma = c(0, 0, 2, 0))
lab <- c('Hamming loss (HL)','Classification accuracy
(CL)', 'Precision (PR)','Recall (RE)','F2','Accuracy
(AC)')
lab_main <- c('Hamming loss','Classification accuracy',
'Precision','Recall','F2','Accuracy')
for (i in 1:6)
{
```

Stellenbosch University https://scholar.sun.ac.za

```
  boxplot(BR[,i],CC[,i],ECC[,i],LCC[,i],BR2[,i],CC2[,i],E
  CC2[,i],LCC2[,i],

  names=c('BR','CC','ECC','LCC','BR2','CC2','ECC2','LCC2'
  ),
          ylab=lab[i], main=lab_main[i],
  col=brewer.pal(4,"Greens"), ylim=c(0,1))
  grid()
  abline(v=4.5)
}
mtext('Distribution of ML evaluation measures by method
for t: 0.5 vs. 5', outer = TRUE, cex = 1.2)
```

```
#############
# HEATMAPS #
#############

library(gplots)
bench_data <- SIM_1m
L=bench_data$L
P=bench_data$P
VI <- bench_data$A_global_std
colo <- colorRampPalette(c("white","gold","red","black"))
nam <- rep("", nrow(t(VI)))
nam[seq(1,nrow(t(VI)), 2)] <-
rownames(t(VI))[seq(1,nrow(t(VI)), 2)]
colnames(VI) <- nam

heatmap.2(t(VI),  Rowv=F,
dendrogram="column",keysize=1,main="Heatmap of importance
values: t = 5",
          trace="none",col = colo(L*(P+L)))
```

```
##########################################
# RELEVANT VARIABLE CORRECTLY IDENTIFIED #
##########################################

VI_results <- matrix(c(
SIM_1m$VIprop,SIM_2m$VIprop,SIM_3m$VIprop,SIM_4m$VIprop,
SIM_5m$VIprop,SIM_6m$VIprop,SIM_7m$VIprop,SIM_8m$VIprop,
SIM_9m$VIprop,SIM_10m$VIprop,SIM_11m$VIprop,SIM_12m$VIpro
p,
SIM_13m$VIprop,SIM_14m$VIprop,SIM_15m$VIprop,SIM_16m$VIpr
op,
SIM_17m$VIprop,SIM_18m$VIprop,SIM_19m$VIprop,SIM_20m$VIpr
op,
SIM_21m$VIprop,SIM_22m$VIprop,SIM_23m$VIprop,SIM_24m$VIpr
op,
SIM_25m$VIprop,SIM_26m$VIprop,SIM_27m$VIprop,SIM_28m$VIpr
op,
```

Stellenbosch University https://scholar.sun.ac.za

```
SIM_29m$VIprop,SIM_30m$VIprop,SIM_31m$VIprop,SIM_32m$VIpr
op),nrow=8,ncol=4,byrow=T)

VI_results <- cbind(VI_results,apply(VI_results,1,mean))
VI_results <- rbind(VI_results,apply(VI_results,2,mean))
colnames(VI_results) <-
c('rho=0.1,t=0.5','rho=0.1,t=5','rho=0.8,t=0.5','rho=0.8,
t=5','mean VI')
rownames(VI_results) <- c('ntrain=200,p=50,L=5',
                          'ntrain=200,p=50,L=20',
                          'ntrain=200,p=100,L=5',
                          'ntrain=200,p=100,L=20',
                          'ntrain=1000,p=50,L=5',
                          'ntrain=1000,p=50,L=20',
                          'ntrain=1000,p=100,L=5',
                          'ntrain=1000,p=100,L=20',
                          'mean VI')

VI_results <- round(VI_results,3)
VI_results
```

Stellenbosch University https://scholar.sun.ac.za

APPENDIX D: Chapter 7

D.1 DATA PREPARATION AND SPLITTING

```
#########################
# CREDIT BUREAU DATASET #
#########################

library(RColorBrewer)
library(randomForest)

####################
# DATA PREPARATION #
####################

#cs_data_orig <- read.table("MonikaMScData_20160628.txt",
header=T, sep="|")   #87143 obs. of  1431 variables:
cs_data_noid <- cs_data_orig[,-c(1:3)]    # remove IDs
and retro
fact <- sapply(cs_data_noid, function(x)
ifelse(is.factor(x) & length(levels(x)) <= 1, FALSE,
TRUE)) # factors of length 1 are TRUE: 3 predictors
cs_data <- cs_data_noid[,fact]  # remove single-factor
variables, 87143 obs. of  1425 variables:
cs_data_na <- cs_data
L <- 15; P <- ncol(cs_data_na) - L #1410
prop_na <-
sum(apply(cs_data_na,2,function(x)sum(is.na(x))))/(nrow(c
s_data_na)*(L+P)) # all data: proportion of NAs is
1.719538%.
cs_data_clean <- na.roughfix(cs_data_na)
```

```
######################################
# SPLITTING TRAIN, VALIDATION, TEST #
######################################
# The following code splits the data into 50% training,
25% validation and 25% test sets randomly.
N <- nrow(cs_data_clean) # 87143
set.seed(1)
tv_index <- sample(N,floor(N*0.75),replace=FALSE)
#training/validation data
train_val_data <- cs_data_clean[tv_index,] # 65357 (75%)
test_data <- cs_data_clean[-tv_index,] # 21786 (25%)
set.seed(1)
Nt <- nrow(train_val_data) # 65357
train_index <- sample(Nt,floor(Nt*(2/3)),replace=FALSE)
train_data <- train_val_data[train_index,] # 43571 (50%)
val_data <- train_val_data[-train_index,] # 21786 (25%)
```

Stellenbosch University https://scholar.sun.ac.za

```
###############
# EXPLORATORY #
###############

# The following code summarises the ML dataset.
LCard_train <-
mean(apply(train_val_data[,(P+1):(P+L)],1,sum))    #
1.579249
Dens_train <-
mean(apply(train_val_data[,(P+1):(P+L)],2,sum)/nrow(train
_val_data))  # 0.1052833
Uniq_train <- nrow(unique(train_val_data[,(P+1):(P+L)]))
# 759
summary(apply(train_val_data[,(P+1):(P+L)],1,sum))

# Exploratory graphs
tiff("prac_labdist.tiff", height = 25, width = 30, units
= 'cm',compression = "lzw", res = 300)
dens <-
apply(train_val_data[,(P+1):(P+L)],2,sum)/nrow(train_val_
data)
ylim <- c(0, 1.1*max(dens))
barp <- barplot(dens,main=c('Credit bureau dataset:
accounts distribution'),ylab=c('Relative frequency'),
                names.arg=1:L,
cex.main=2,cex.lab=1.5,ylim=ylim,
xlab='Accounts',col=ifelse(dens <
0.1,brewer.pal(3,"Greens")[2],brewer.pal(3,"Greens")[3]))
text(x = barp, y = dens, label = round(dens,3), pos = 3,
cex = 1.5)
dev.off()

tiff("prac_labdist2.tiff", height = 25, width = 30, units
= 'cm',compression = "lzw", res = 300)
dens2 <- NULL
for (i in c(0:8))
  dens2[i+1] <-
round(sum(apply(train_val_data[,(P+1):(P+L)],1,sum)==i)/n
row(train_val_data),4)
ylim <- c(0, 1.1*max(dens2))
barp2 <- barplot(dens2,main=c('Credit bureau dataset:
number of accounts per client'),ylab=c('Relative
frequency'),
                 names.arg=c(0:8),
cex.main=2,cex.lab=1.5,ylim=ylim, xlab='Number of
accounts per client',col=ifelse(dens <
0.1,brewer.pal(3,"Greens")[2],brewer.pal(3,"Greens")[3]))
text(x = barp2, y = dens2, label = round(dens2,4), pos =
3, cex = 1.5)
dev.off()
```

Stellenbosch University https://scholar.sun.ac.za

D.2 SCREENING OF PREDICTORS

```
###################
# Rsq COMPUTATION #
###################

# In this code, the R squared value is computed for all
predictors in order to do a screening of the predictors.
# Rsq based on train data
fact2 <- sapply(train_data, is.factor) # all factors: 174
variables
train_data_num <- train_data
train_data_num[,fact2] <-
sapply(train_data[,fact2],as.numeric) # factors converted
to numeric for multiple R

x <- train_data_num[,1:P]
y <- train_data_num[,(P+1):(P+L)]
Rsq <- matrix(0,nrow=ncol(x),ncol=1); rownames(Rsq) <-
colnames(x);colnames(Rsq) <- c('Mult_R')
for (i in 1:ncol(x))
{
  data_xy <- cbind(x=x[,i],y)
  Rsq[i,] <-
   round(summary(lm(x~.,data=data_xy))$r.squared,3)
}
Rsq_ord <- Rsq[order(Rsq),]

# Graph #
tiff("prac_rsqhist.tiff", height = 25, width = 30, units
= 'cm',compression = "lzw", res = 300)
hist(Rsq,main=expression(paste('Distribution of ', R^2)),
     xlab=expression(paste(R^2, '
values')),cex.main=2,cex.lab=1.5,col=brewer.pal(3,"Greens
")[2])
dev.off()
summary(Rsq)
```

```
####################
# BR VS VALIDATION #
####################

# The following code fits BR to train data using various
number of predictors and obtains the validation errors
based on predicting the validation data.
BR_VS <- matrix(0,ncol=6,nrow=8)
i <- 0
for (t in c(0, 0.25, 0.5, 0.75, 0.8, 0.85, 0.9, 0.95))
{
```

Stellenbosch University https://scholar.sun.ac.za

```
  i <- i+1
  Rsq_rem_temp <- Rsq_ord[Rsq_ord >=
  quantile(Rsq_ord,t)[[1]]] # remove irrelevant variables
  Psub_temp <- length(Rsq_rem_temp)
  train_data_VS_temp <-
  data.frame(train_data[,names(Rsq_rem_temp)],
  train_data[,(P+1):(P+L)])
  val_data_VS_temp   <-
  data.frame(val_data[,names(Rsq_rem_temp)],
  val_data[,(P+1):(P+L)])

  # BR
  cs_BR_VS <-
  BR_fun_parall(P=Psub_temp,L=L,train_data_VS_temp,val_da
  ta_VS_temp,notree=50)
  gc(verbose=F)
  BR_VS[i,] <- unlist(cs_BR_VS$r)[-5]
}

############ Graphs threshold ###############
#library(calibrate)
BR_VS <- t(BR_VS)
colnames(BR_VS) <-
c('0%','25%','50%','75%','80%','85%','90%','95%')
rownames(BR_VS) <- c('HL','CL','PR','RE','F2','AC')
x <- 1:8
tiff("Prac_thresh_all_L.tiff", height = 25, width = 25,
units = 'cm',compression = "lzw", res = 300)
par(mfrow=c(3,2),oma = c(0, 0, 2, 0))
ylimits <-
list(c(0.075,0.135),c(0.2,0.26),c(0.69,0.75),c(0.3,0.36),
c(0.43,0.49),c(0.28,0.34))
lab_main <- c('Hamming loss','Classification accuracy',
'Precision','Recall','F2','Accuracy')
for (i in 1:6)
{
  plot(y=BR_VS[i,],x=x,type="o",xaxt =
  "n",main=lab_main[i],xlab='VS Threshold',

  ylab=colnames(BR_VS)[i],col=brewer.pal(3,"Greens")[3],p
  ch=5,lwd=2, ylim=ylimits[[i]])
  grid()
  abline(v=6,col='darkorange')
  axis(1, at=1:8, labels=colnames(BR_VS))
}
mtext('ML evaluation measures by VS threshold value',
outer = TRUE, cex = 1.2)
dev.off()

# Decision is made based on the graphs produced
t_final <- 0.85
```

Stellenbosch University https://scholar.sun.ac.za

```
Rsq_rem <- Rsq_ord[Rsq_ord >=
quantile(Rsq_ord,t_final)[[1]]] # remove irrelevant
variables
Psub <- length(Rsq_rem) # 214

# Computation of datasets after VS is performed
test_data_VS  <- data.frame(test_data[,names(Rsq_rem)],
test_data[,(P+1):(P+L)]) # 21786 obs. of  229 variables:
train_val_data_VS <-
data.frame(train_val_data[,names(Rsq_rem)],
train_val_data[,(P+1):(P+L)])
```

D.3 COMPARISON OF CLASSIFIERS

```
#############################
# MODELS VALIDATION FITTING #
#############################

# The following code uses training and validation data to
fit the four ML classifiers. Test error rates are
obtained based on predicting the test dataset.

# BR
start.time <- Sys.time()
cs_BR <-
BR_fun_parall(P=Psub,L=L,train_val_data_VS,test_data_VS,n
otree=200)
end.time <- Sys.time()
t.timeBR <- end.time-start.time
gc(verbose=F)

# CC
start.time <- Sys.time()
cs_CC  <-
CC_fun_parall(P=Psub,L=L,train_val_data_VS,test_data_VS,n
otree=200)
end.time <- Sys.time()
t.timeCC <- end.time-start.time
gc(verbose=F)

# ECC
start.time <- Sys.time()
cs_ECC <-
ECC_fun_parall(P=Psub,L=L,train_val_data_VS,test_data_VS,
notree=200)
end.time <- Sys.time()
t.timeECC <- end.time-start.time
gc(verbose=F)
```

Stellenbosch University https://scholar.sun.ac.za

```
# LCC
start.time <- Sys.time()
cs_LCC <-
LCC_fun_parall(P=Psub,L=L,train_val_data_VS,test_data_VS,
notree=200)
end.time <- Sys.time()
t.timeLCC <- end.time-start.time
gc(verbose=F)
```

D.4 ANALYSIS OF RESULTS

```
###########################
# HL, AUC, GINI PER LABEL #
###########################

# This code computes the individual HL, AUC and Gini
coefficient measures per label.

# HL
HL_ind_LCC <- round(cs_LCC$r$HLind,3)
names(HL_ind_LCC) <- paste0('Label ',1:15)
write.csv(t(HL_ind_LCC),file='HL_ind_LCC.csv')


# AUC, Gini
LCC_prob <- cs_LCC$W_prob
LCC_ytrue <- cs_LCC$Y_true
library(ROCR)
auc <- NULL; gini <- NULL
for (l in 1:15)
{
  pred <- prediction(LCC_prob[,l], LCC_ytrue[,l])
  perf <- performance(pred, "tpr", "fpr")
  auc.value <- performance(pred, measure="auc")
  auc[l] <- round(attr(auc.value,'y.values')[[1]],3)
  gini[l] <- abs(1-2*auc[l])
}
res_roc <- cbind(auc,gini)
rownames(res_roc) <- paste0('Label',1:15)
write.csv(t(res_roc),file='res_roc.csv')

# Graph of ROC plots for labels 2 and 3.
tiff("gini.tiff", height = 10, width = 17, units = 'cm',
# this function is used to safe a graph in a better
quality
    compression = "lzw", res = 300)
par(mfrow=c(1,2))
pred <- prediction(LCC_prob[,2], LCC_ytrue[,2])
perf <- performance(pred, "tpr", "fpr")
auc.value <- performance(pred, measure="auc")
```

Stellenbosch University https://scholar.sun.ac.za

```
auc <- round(attr(auc.value,'y.values')[[1]],3)
plot(performance(pred, measure="tpr",
x.measure="fpr"),col="tomato3",
     main="ROC plot for label 2",lwd=3)
abline(a=0, b=1,col="slateblue4",lty=2,lwd=3)
text(0.65,0.1,paste0("AUC value: ",auc),bty = "n")

pred <- prediction(LCC_prob[,3], LCC_ytrue[,3])
perf <- performance(pred, "tpr", "fpr")
auc.value <- performance(pred, measure="auc")
auc <- round(attr(auc.value,'y.values')[[1]],3)
plot(performance(pred, measure="tpr",
x.measure="fpr"),col="tomato3",
     main="ROC plot for label 3",lwd=3)
abline(a=0, b=1,col="slateblue4",lty=2,lwd=3)
text(0.65,0.1,paste0("AUC value: ",auc),bty = "n")
dev.off()
```

```
#############################
# VARIABLE IMPORTANCE PLOTS #
#############################

# The following code plots two heatmaps, VI plot, and
identifies the most globally and locally significant
predictors.

# Standardise LCC A_final
A_global2 <- cs_LCC$A_final
A_global_std <- matrix(0,ncol=Psub+L,nrow=L)
colnames(A_global_std) <-
c(paste0("X",1:Psub),paste0("label",1:L));
rownames(A_global_std) <- paste0("label",1:L)
for (k in 1:L)
  A_global_std[k,] <-
round(((A_global2[k,]/max(A_global2[k,]))*100),1)

tiff("prac_heat_LCC1.tiff", height = 20, width = 25,
units = 'cm',compression = "lzw", res = 300)
library(gplots)
VI <- A_global_std
colo <- colorRampPalette(c("white","gold","red","black"))
nam <- rep("", nrow(t(VI)))
nam[seq(1,nrow(t(VI)), 4)] <-
rownames(t(VI))[seq(1,nrow(t(VI)), 4)]
colnames(VI) <- nam
heatmap.2(t(VI),  Rowv=F,
dendrogram="column",keysize=1,main="Heatmap of credit
bureau data",
          trace="none",col = colo(L*(Psub+L)))
dev.off()
```

Stellenbosch University https://scholar.sun.ac.za

```
tiff("prac_heat_LCC2.tiff", height = 20, width = 25,
units = 'cm',compression = "lzw", res = 300)
VI <- cs_LCC$A_final/max(cs_LCC$A_final)*100
colo <- colorRampPalette(c("white","gold","red","black"))
nam <- rep("", nrow(t(VI)))
nam[seq(1,nrow(t(VI)), 4)] <-
rownames(t(VI))[seq(1,nrow(t(VI)), 4)]
colnames(VI) <- nam
heatmap.2(t(VI),  Rowv=F,
dendrogram="column",keysize=1,main="Heatmap of credit
bureau data",
          trace="none",col = colo(L*(Psub+L)))
dev.off()

# Global VI plot
tiff("VI_global.tiff", height = 20, width = 25, units =
'cm',compression = "lzw", res = 300)
VI_global <- round(apply(cs_LCC$A_final,2,mean),2)
VI_global2 <-
sort((VI_global/max(VI_global))*100,decreasing=T)
b <- barplot(VI_global2,horiz=T,xlab="Variable Importance
Gini Index",
             col='lightgreen',main="Variable Importance
Plot: global importance")
#axis(side=1,at=c(0,.2,.4,.6,.8,1),labels=seq(0,100,by=20
))
dev.off()

# Most important global predictors
VI_global3 <- VI_global2[VI_global2 >
quantile(VI_global2,.9)]
VI_gl_vars <- matrix(0,ncol=2,nrow=length(VI_global3))
VI_gl_vars[,1] <- names(VI_global3)
VI_gl_vars[,2] <- round(VI_global3,2)
write.csv((VI_gl_vars),file='VI_gl_vars.csv')
write.csv(t(VI_gl_vars),file='VI_gl_vars_t.csv')

# Locally important predictors
VI_local <- matrix(0,ncol=L*2,nrow=5)
colnames(VI_local) <- rep(paste0('Label',1:L),each=2)
ncount <- 0
for (i in seq(1,L*2-1,by=2))
{
  ncount <- ncount + 1
  VI_local[,i] <- names(sort(A_global_std[ncount,],
decreasing=T)[1:5])
  VI_local[,i+1] <- sort(A_global_std[ncount,],
decreasing=T)[1:5]
}
write.csv(VI_local,file='VI_local.csv')
write.csv(t(VI_local),file='VI_local_t.csv')
```

Stellenbosch University https://scholar.sun.ac.za

```
############################################
# GRAPH OF PROBABILITIES AND ACTUAL LABELS #
############################################

# The following graph plots the prediction probabilities
as well as the actual labels present per client.

LCC_prob <- cs_LCC$W_prob
LCC_ytrue <- cs_LCC$Y_true
tiff("Percase3.tiff", height = 20, width = 25, units =
'cm',compression = "lzw", res = 300)
par(mfrow=c(2,2))
for (i in sample(1:1000,4))
  {
  LCC_prob[i,]
  LCC_ytrue[i,]
  b2 <- barplot((LCC_prob[i,]),xlab="Response
  variables",cex.main=0.9,
              ylab="Probability",xaxt="n",ylim=c(0,1),
  col = "cornsilk3",
              main=c("Predicted vs. Actual Accounts
  Opened"),
              cex.main=1.5)
  axis(1, at=b2, labels=paste0('L',1:15), tick=T)

  ind_coord <- which(LCC_ytrue[i,] == 1)
  for (j in ind_coord)
  lines(x=rep(b2[j],2),
           y=c(0,95),lwd=3,col="orangered",xpd=F)

  lab2 <- ifelse(LCC_prob[i,]=="0","",
  round(LCC_prob[i,],2))
  text(x=b2,y=LCC_prob[i,]+0.03, labels=lab2,xpd=T)
  legend('topright',bty="n",c("Predicted","Actual"),
  lty=c(1,1),
       lwd=c(4,4), col=c("cornsilk3","orangered"),
  inset=c(.1,0),
       cex=1.5)
}
dev.off()
```