BINARY CLASSIFICATION TREES:

A COMPARISON WITH POPULAR CLASSIFICATION METHODS IN STATISTICS USING DIFFERENT SOFTWARE

by

Morné Michael Connell Lamont

Assignment submitted in partial fulfillment of the requirements for the degree

MASTER OF COMMERCE

in the Department of Statistics and Actuarial Science,
Faculty of Economic and Management Sciences,
Stellenbosch University

Supervisor:
Prof. N. Louw

December 2002

Declaration

I, the undersigned, hereby declare that the work contained in this assignment is my own original work and has not been previously in its entirety or in part submitted at any university for a degree.

Morné M.C. Lamont: Date:

Summary

Consider a data set with a categorical response variable and a set of explanatory variables. The response variable can have two or more categories and the explanatory variables can be numerical or categorical. This is a typical setup for a classification analysis, where we want to model the response based on the explanatory variables.

Traditional statistical methods have been developed under certain assumptions such as: the explanatory variables are numeric only and/ or the data follow a multivariate normal distribution. In practice such assumptions are not always met. Different research fields generate data that have a mixed structure (categorical and numeric) and researchers are often interested using all these data in the analysis. In recent years robust methods such as classification trees have become the substitute for traditional statistical methods when the above assumptions are violated. Classification trees are not only an effective classification method, but offer many other advantages.

The aim of this thesis is to highlight the advantages of classification trees. In the chapters that follow, the theory of and further developments on classification trees are discussed. This forms the foundation for the CART software which is discussed in Chapter 5, as well as other software in which classification tree modeling is possible. We will compare classification trees to parametric-, kernel- and *k*-nearest-neighbour discriminant analyses. A neural network is also compared to classification trees and finally we draw some conclusions on classification trees and its comparisons with other methods.

Opsomming

Beskou 'n datastel met 'n kategoriese respons veranderlike en 'n stel verklarende veranderlikes. Die respons veranderlike kan twee of meer kategorieë hê en die verklarende veranderlikes kan numeries of kategories wees. Hierdie is 'n tipiese opset vir 'n klassifikasie analise, waar ons die respons wil modelleer deur gebruik te maak van die verklarende veranderlikes.

Tradisionele statistiese metodes is ontwikkel onder sekere aannames soos: die verklarende veranderlikes is slegs numeries en/ of dat die data 'n meerveranderlike normaal verdeling het. In die praktyk word daar nie altyd voldoen aan hierdie aannames nie. Verskillende navorsingsvelde genereer data wat 'n gemengde struktuur het (kategories en numeries) en navorsers wil soms al hierdie data gebruik in die analise. In die afgelope jare het robuuste metodes soos klassifikasie bome die alternatief geword vir tradisionele statistiese metodes as daar nie aan bogenoemde aannames voldoen word nie. Klassifikasie bome is nie net 'n effektiewe klassifikasie metode nie, maar bied baie meer voordele.

Die doel van hierdie werkstuk is om die voordele van klassifikasie bome uit te wys. In die hoofstukke wat volg word die teorie en verdere ontwikkelinge van klassifikasie bome bespreek. Hierdie vorm die fondament vir die CART sagteware wat bespreek word in Hoofstuk 5, asook ander sagteware waarin klassifikasie boom modelering moontlik is. Ons sal klassifikasie bome vergelyk met parametriese-, "kernel"- en "*k*-nearest-neighbour" diskriminant analise. 'n Neurale netwerk word ook vergelyk met klassifikasie bome en ten slote word daar gevolgtrekkings gemaak oor klassifikasie bome en hoe dit vergelyk met ander metodes.

To my loving family

Parents: Connell and Thelma

Brother and Sisters: Stanley, Colleen and Julaine (Nita)

Nephew and Nieces: Marcus, Monique and Lucinda

Thank you for all your love, support and encouragement.

Acknowledgements

I would like to express my sincere thanks and appreciation to:

- **God**, for blessing me with many opportunities and the wisdom to pursue my dreams.
- my supervisor, **Prof. Nelmarie Louw**. Through her guidance, inspiration and ideas, she has made this study a pleasant experience. I thank her for her suggestions and contributions in the preparation of this manuscript. I appreciate her encouragement and leadership very much.
- **Prof. Sarel Steel**, for always being helpful and willing to organise funds for my studies, for his invaluable input during the preparation of this manuscript. His encouragement and leadership is much appreciated.
- the **National Research Foundation**, for the funds they have provided for my M.Com studies and for their contribution towards my future and career.
- **Stellenbosch University** for awarding me the Postgraduate Development Programme bursary.
- my **family**, **friends** and **colleagues** who have shown interest in my studies, for their moral support and for listening to my problems.
- the **Institute for Maritime Technology** and **Prof. S. van der Berg**, for the use of their data in this study. This data was used with permission from **Dr. Martin Kidd** and I thank him too for his contribution.
- the **Department of Statistics and Actuarial Science** at Stellenbosch University, for accepting me as a postgraduate student and for making some of my dreams a reality.
- my employer, **ARC-Biometry Unit**, for accommodating my studies by giving me time off from work to attend lectures.

"*The fear of the LORD is the beginning of wisdom, and the knowledge of the Holy One is insight.*" - Proverbs 9:10

Contents

CHAPTER 1

Introduction to Binary Classification Trees

1.1 CLASSIFIER AND CLASSIFICATION

Consider a data set in which the response variable consists of a number of categories, denoted by $j=1, 2, \ldots, J$, each containing a number of observations, and suppose a set of explanatory variables, denoted by $\boldsymbol{X}=(X_1, X_2, \ldots, X_p)$, are given for every observation in category j. Based on this information a model for the response can be built, by systematically summarizing all the information in the explanatory variables. The purpose of such a model is to classify to which category an observation belongs, based on the set of explanatory variables alone. Such a model is known as a classifier or classification rule. Any new observations with unknown categories that come from the same population as the observations that were used to construct the model, can now be classified into a known category using the classifier. Classifiers are based on past data, where observations have known categories and explanatory variables that are of interest in the classification process. This process is called *supervised learning*. The purpose of a classification analysis is to construct a classifier which accurately classifies new cases, and which provides the analyst with more insight into and understanding of the predictive structure of the data. Suppose a data set $\mathcal{L}$ consists of a response variable having J categories, a vector of explanatory variables $\boldsymbol{X}$ and N observations, then a classifier $d(\boldsymbol{X})$ is defined as a function of vector $\boldsymbol{X}$ that can classify any of the N observations into one, and only one of the J categories. Examples of $d(\boldsymbol{X})$ are canonical discriminant functions, linear and quadratic discriminant functions, the kernel discriminants, k-nearest-neighbour classifiers, neural networks, classification trees, logistic regression, etc., which under certain conditions have their own advantages and disadvantages. Because some classification techniques originated from fields outside traditional statistics, the terminology for the techniques may differ as will be seen in this text.

TABLE 1.1: *Some terminology of the different classification methods.*

Classification trees	***Neural Networks***	***Discriminant analysis***
Classes	*Targets/Classes*	*Groups/Populations/Categories*
Cases	*Patterns*	*Observations*
Measurements	*Features*	*Explanatory variables*

Table 1.1 gives some of the terminological names as they are used in these fields, but have the same meanings. Each field uses different notations and this will be clarified in the explanations that follow. Symbols that are written in **bold** in this text represent matrix or vector notation.

1.2 BACKGROUND ON CLASSIFICATION TREES

Tree-based models have become increasingly popular because of some advantages they have over other classical statistical models, especially their robustness and easily interpretable results. Unlike parametric models, trees are not based on any assumptions about the underlying probability distribution. It is a completely non-parametric, nonlinear approach to modeling. The use of these types of models goes back to the 1960's and since then it has been incorporated and popularized in a large number of computer packages. Trees originated as an analytic tool used by social scientists to explore their data.

Starting in 1973, American statisticians from Stanford University (Professors Jerome Friedman and Richard Olshen) and University of California at Berkeley (Professors Leo Breiman and Charles Stone), applied tree methodology to classification and regression problems, thus bringing trees into the world of statistics. Their monograph (Breiman *et al.*, 1984) on classification and regression trees (CART) demonstrates their influence in the development of trees. They have developed new theories for CART as a way of strengthening and extending the original tree methods. All their ideas have been coded into computer software and incorporated into the CART® software. Many analysts have praised the ideas and software developed by these statisticians.

1.3 BINARY CLASSIFICATION TREE: AN EXAMPLE

The purpose of classification trees is to obtain insight into a data set, to explore and identify hidden structures, isolate them and finally to generate a reliable, informative and easily interpretable model with strong and accurate classification power. In a classification tree analysis, we start out with a learning sample containing classes (2 or more) and a number of measurements on which a tree is grown. The beginning of the tree is the root node and at this node classification tree methodology makes use of recursive binary partitioning, by splitting it up into two purer nodes. Those two nodes are split up into further nodes and the recursive splitting process continues in this way, until no more splits are possible and a maximal tree has been grown on the data. This maximal tree is then pruned upwards to obtain the best optimal tree which is ultimately the final classification tree. The process is easily explainable and understandable, but it is a very involved and computer intensive exercise. It has been incorporated into special-purpose software called CART, as well as other software packages, to make the technique user-friendly.

To describe the entire classification tree modeling process, we start out by giving an example of such a tree. Then we proceed to explain what happens at each stage of the tree growing process. Figure 1.1 is a hypothetical hierarchical classification tree example of patients to be classified into heart-attack prone [1] and not heart-attack prone [2] on the basis of their age, weight and regular exercise. The example is taken from Johnson and Wichern (1992). In this problem we have 2 classes (1 or 2), and 3 measurements namely age (numerical), weight (categorical in this case; either overweight or not) and exercise (categorical; either regular or not). Thus we have a mixture of data structures (categorical and numerical). The first split, Split 1, starts out at root node t_1 and asks the question "is the patient over 45 years of age?". If yes, that case is thrown in non-terminal node t_2 else it goes to terminal node t_3. Non-terminal node t_2 is split into two by Split 2, resulting in nodes t_5 and t_4. Then finally, node t_4 is split by Split 3 into t_6 and t_7.

At each split a different question is asked and the cases are then split accordingly. At each stage a case will fall into one of two nodes. This node can either be terminal (like t_3, t_5, t_6 and t_7) or non-terminal (like t_1, t_2 and t_4). When all nodes become terminal, the tree growing process has stopped and classes need to be assigned to the nodes. Class 1

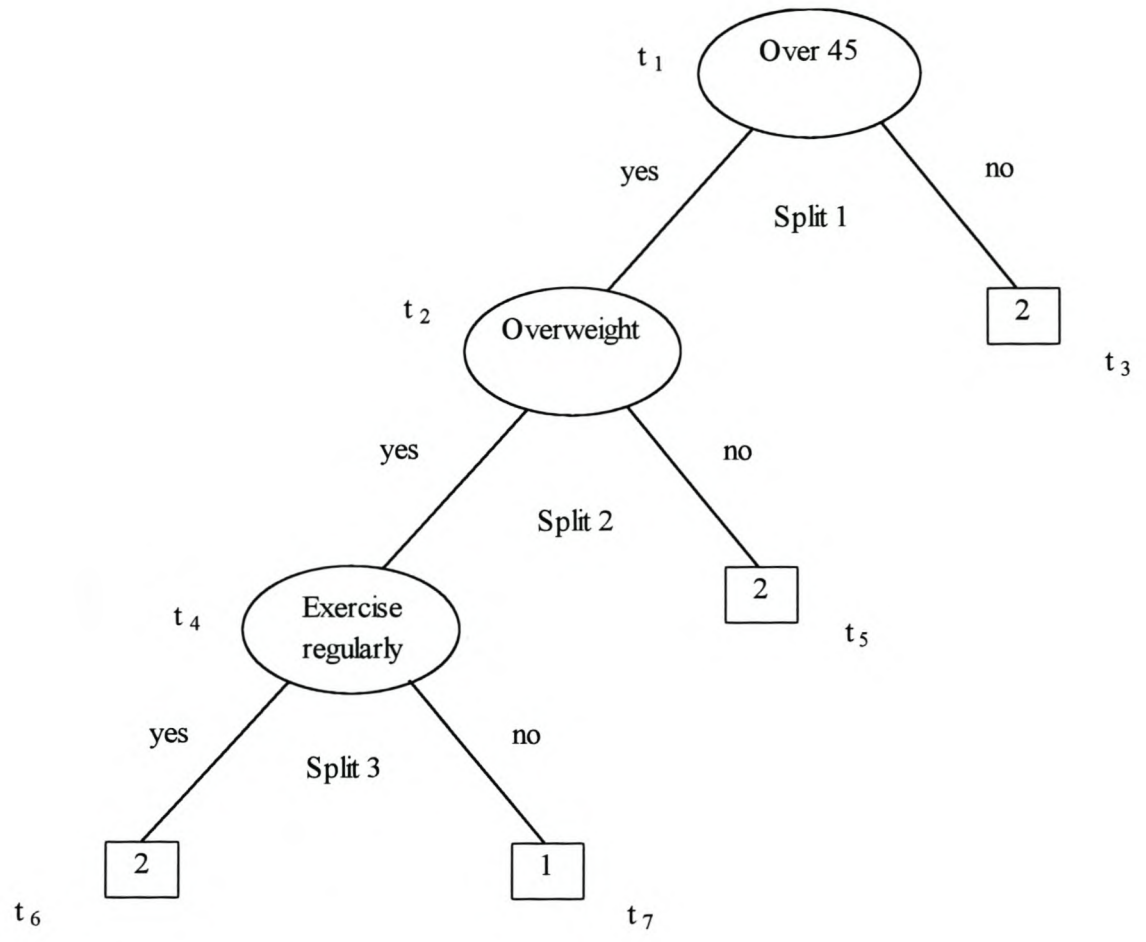

FIGURE 1.1: *An example of the hierarchical classification tree.*

has been assigned to node t_7, and class 2 has been assigned to t_6, t_5 and t_3. When interpreting this tree, we can see that patients who are over 45, overweight and do not exercise regularly fall into the heart-attack prone class. The non heart-attack prone cases have a similar interpretation. For example if a patient is not over 45, he/ she will be classified in the non heart-attack prone class. So will a patient over 45 but not overweight, as well as a patient over 45, overweight but exercises regularly, be classified in the non heart-attack prone class. The entire construction of a classification tree revolves around three basic elements:

- How to select the splits.
- How to decide when a node is declared terminal.
- How to assign a class to each terminal node.

1.4 HOW TO SELECT THE SPLITS

In Figure 1.1, t_1 is known as a parent/ ancestor node and t_2 and t_3 are child/ descendant nodes of t_1. Node t_2 is the ancestor of t_4 and t_5, therefore a non-terminal node is an ancestor as well as a descendant node. Splits are formed by a set of questions and the idea behind every split is to obtain a purer subset of cases in the descendant nodes than in the ancestor nodes. Purer means that the split is trying to isolate the classes in the ancestor node by directing each case into a separate descendant node. The following explain how this objective is achieved:

- let $p(j|t)$, $j=1,\ldots, J$ be the node proportions (the proportions of the cases in node t that belongs to class j). Therefore $p(1|t)+ p(2|t)+ p(3|t)+ \ldots + p(J|t)=1$.
- Let the measure $i(t)$ be defined by the impurity function ϕ of node t:

$$i(t) = \phi\,(p(1|t), p(2|t), p(3|t), \ldots, p(J|t)).$$

The function ϕ is defined such that the impurity will be a maximum if each class is represented by the same number of cases in t, i.e. $p(1|t)= p(2|t)= p(3|t)= \ldots = p(J|t)$. Impurity will be a minimum if only one of the J classes are present in node t, i.e. $p(j|t)=1$, $p(k|t)=0$, such that $k,j \in \{1,\ldots, J\}$ and $j \neq k$. With each binary split in the classification tree the aim is to minimize $i(t)$. Thus the impurity of the descendant nodes must be less than the impurity in the ancestor node. For a given split s the decrease in impurity will determine the goodness of the split.

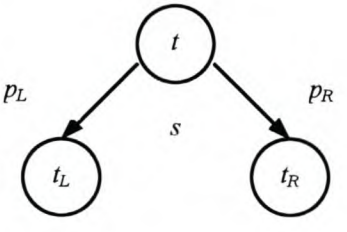

FIGURE 1.2: *Splitting node t into t_R and t_L.*

Say split s from a number of splits S directs a proportion p_R of the data cases in t to the first descendant node t_R and the proportion p_L of data cases to the second descendant node t_L, as in Figure 1.2. Then the decrease in impurity of node t is defined as

$$\Delta i(s,t) = i(t) - p_R i(t_R) - p_L i(t_L).$$

The split s that maximizes this function will be the best split. The process is done for every node until the maximal tree is obtained. It can be shown that by splitting each node according to this rule, the whole tree is actually becoming more pure. This proof will be shown in the more detailed discussion of a splitting rule. A number of splitting criteria are used in classification trees. These include the commonly used Gini criterion and the Twoing criterion.

1.5 HOW TO DECIDE WHEN A NODE IS DECLARED TERMINAL

An initial proposal to stop the splitting, was to define a threshold $\beta > 0$, and if the maximum decrease in node impurity was less than β, the splitting would be stopped and the node declared terminal. However, this is not satisfactory in the sense that some structures in the tree remain hidden beyond that point and valuable splits can be missed. Pruning the maximal tree became the better option. By pruning the tree upwards according to the minimal cost-complexity measure of pruning, a sequence of finite subtrees is generated. Each tree is evaluated by estimating its error rate using either the test sample estimate or V-fold cross-validation estimate. Depending on the error rate estimate chosen by the analyst, the optimal tree is selected by choosing the tree for which the estimated error is a minimum from the sequence of subtrees. In this way a best tree is selected and the terminal nodes are determined automatically as a result of this process.

1.6 HOW TO ASSIGN A CLASS TO EACH TERMINAL NODE

The class that is the most represented in a node, is the class that will be assigned to that node. For example if class 1 has the highest proportion of cases in node t, class 1 will be assigned to node t. The class assignment is the easiest of the three elements and is a logical process.

1.7 ADVANTAGES OF CLASSIFICATION TREES

1. It can be applied to any data structure. Unlike for example discriminant analysis (Chapter 6), that can handle only numerical data, classification trees can handle both categorical and numerical variables as explanatory variables.
2. Classification trees can handle large data sets of high dimensionality and can effectively model two or more classes.
3. The final classification tree has a simple hierarchical form and has an easy, logical interpretation.
4. It provides an automatic stepwise variable selection. By considering all possible splits and selecting the best split, classification trees has a built-in variable selection. It will only split on those variables that are found to be the most important according to the goodness-of-split criterion.
5. It provides the classification as well as an estimate of the misclassification probability of a case.
6. In a standard data structure of the independent variables, it is invariant under all monotone transformations. For some transformations of variables such as the logarithm, there will be no change in the classification tree. Thus we could work with the original variables.
7. It is a robust technique and is not affected by outliers/ noise.
8. It is a non-parametric technique and makes no assumptions about the underlying distribution of the data. No model parameters need to be estimated and the data "speak for themselves".
9. It assumes no functional relationship between the response and the measurements.
10. It can use univariate and combinations of variables to determine a split.
11. If certain classes are over-represented, classification trees will handle this by automatic reweighting, thus giving all classes an equal chance in the analysis.
12. It provides surrogate splits (Section 4.2) in the case of missing values. The analyst does not need to throw an observation containing a missing value out or try to estimate it. Classification trees provide a surrogate split and the surrogate split also determines the importance of a variable in a tree.

13. The application of priors and misclassification costs can reduce misclassification of some cases, making the tree a better classifier.
14. The classification power of trees can be increased by the use of resampling techniques (Bagging and Boosting in Chapter 3). You can use a committee of experts instead of a single tree to classify a case. Random forests can also be used (Section 4.3).
15. Classification trees can be used for preliminary modeling. Like ordinary statistical variable selection, trees can be used likewise to reduce dimensionality, enabling the analyst to choose the most important variables in the trees and then use these variables in a regression, discriminant, neural network analysis, etc.
16. You can also predict the class probabilities[1] with trees. Using a similar Bayesian approach as discriminant analysis, you can predict the posterior probability that a case belongs to a class.
17. Although computationally very intensive, classification trees are mathematically much simpler than other methods.

1.8 SUMMARY

In statistical research problems mixtures of data types (messy data) are often gathered, including nominal, ordinal and numerical data. The applied statistician/ analyst will always aim to investigate all data in his analyses. Because of the strict assumptions many classical statistical models are based on, certain data types are not suitable for analysis by means of some modeling techniques. Only numerical data (clean data) can be used. For reasons like these, the analyst will be eager to explore a more robust technique as a modeling tool, which can incorporate as many data types as possible. Robust techniques such as classification trees, regression trees and neural networks have become quite popular nowadays in statistical modeling, are included in many statistics textbooks and courses, used in scientific publications and have been incorporated into a number of statistical computing software. They have become the substitute for classical statistical

[1] See Breiman *et al.* (1984) and CART® User's Guide for more reading on class probability trees.

modeling techniques such as least squares regression, discriminant analysis, non-linear regression, logistic regression, cluster analysis, etc.

In the subsequent chapters we will focus mainly on binary classification trees as an alternative classification method to discriminant analysis. Because of its lack of assumptions, classification trees may perform well in cases where discriminant analysis may not be so effective. As seen in Section 1.7, classification trees offer many advantages to the analyst and more theoretical developments have made it quite a flexible and powerful technique. **We will also compare classification trees to a popular neural network technique.** All practical comparisons will be done by applying these techniques to different data sets. **For the purpose of this study different software was used:** CART® version 4.0 for the classification trees analyses, SAS® Base version 8.2 for discriminant analyses and SAS® Enterprise Miner version 4.1 for neural network analyses.

The rest of this text is set out as follows: In Chapter 2 we explain the theory and philosophy of single classification trees. Being dissatisfied with single tree results, Breiman *et al.* (1984) implemented and developed more techniques to improve tree performance, as will be seen in Chapters 3 and 4. In Chapter 3 we illustrate the use of resampling techniques in classification trees. Chapter 4 contains more developments on the basic classification tree, portraying more of its flexibility. In Chapter 5 we discuss the CART software. In Chapter 6 we shortly explain some theory of neural network, parametric- and non-parametric discriminant analysis. In Chapter 7 we look at an example of a classification tree analysis and by using multiple data sets, we will compare the classification results of the classification methods discussed earlier. Chapter 8 contains general conclusions on the classification methods. The notation used in the explanations of classification trees is adopted from Breiman *et al.* (1984). Most discussions will revolve around highlighting the advantages mentioned in this chapter, in comparison with alternative classification methods. We will also discuss some disadvantages of classification trees.

CHAPTER 2

Theory of Binary Classification Trees

2.1 THE SETUP

A learning sample ($\mathcal{L}$) consists of a data matrix $[(j_1, \mathbf{X}_1), (j_2, \mathbf{X}_2), \ldots, (j_N, \mathbf{X}_N)]$ of N cases, where $\mathbf{X}_n$ is the vector of measurements taken on the N cases and j_n is the classes in $\mathcal{L}$, such that $j_n \in \{1, \ldots, J\}$ and $n=1, \ldots, N$. Thus $\mathcal{L}$ is a data set with a response variable of J classes and a set of explanatory variables $\mathbf{X}_n$. Let N_j be the number of cases in class j and let $\pi(j)$ be the prior probabilities of the classes. If $\mathcal{L}$ reflects the expected proportions of the classes, $\pi(j)$ is often estimated by N_j/N. Alternatively the analyst gives $\pi(j)$-values. Let $N(t)$ be the number of cases in any node t and $N_j(t)$ be the number of class j cases in node t, so that $N_j(t)/N(t)$ is the proportion of the cases in node t falling in class j. The probability that a case will be in class j and fall into node t, is then given by

$$p(j,t) = \pi(j)\ \{N_j(t)/N_j\}$$

or

$$p(j,t) = \{N_j/N\}\ \{N_j(t)/N_j\} = N_j(t)/N.$$

The probability $p(t)$ that any case will fall into node t is the sum over all j classes

$$p(t) = \sum_j p(j,t).$$

Then the probability that a case belongs to class j given that it falls in node t is

$$p(j|t) = p(j,t)/p(t),$$

which is equal to $N_j(t)/N(t)$ when $\pi(j) = N_j/N$, and is therefore the proportion of node t, that belongs to class j.

We also introduce the calculation of the resubstitution error rate, which involves growing a classification tree on all the data in $\mathcal{L}$. The same data is sifted through that tree and classified. The proportion of cases that have been classified as a class other than the original class, is called the resubstitution error rate estimate and is defined as

$$R(T) = \frac{1}{N}\sum_{i,j} N_{ij} ,$$

where N_{ij} is the number of cases belonging to class j that have been classified as class i $(i \neq j)$. Although this estimate is easily computable, it is not recommended as an error rate estimate when evaluating the final classification tree, since it under-estimates the true error rate. It does however play a vital role in the minimal cost-complexity measure as a method of obtaining the optimal tree.

A classification tree needs to be grown on $\mathcal{L}$. How is the tree growing process going to take place? The following sections give us an understanding and explanation of the theory underlying the basic tree growing methodology for a single tree.

2.2 SPLITTING RULES

A finite number of splits exist for a tree and each of these possibilities is considered. To give an example, let measurement x_i be a numerical random variable on N cases. For this variable, there would be N different splits. If x_i is a categorical random variable with K categories, then there is $2^{K-1} - 1$ different splits for this variable. In a high dimensional data set with many cases, the number of possible splits becomes enormous.

As mentioned before, the purpose of splitting is to decrease the impurity in the ancestor nodes. The impurity function $i(t)$ and decrease in impurity $\Delta i(s,t)$ was defined earlier as

$$i(t) = \phi(p(1|t), p(2|t), p(3|t), \ldots, p(J|t))$$
$$\Delta i(s,t) = i(t) - p_R i(t_R) - p_L i(t_L).$$

If you consider a grown tree T having a set of terminal nodes $\tilde{T}$, the impurity for T is the sum of the node impurities

$$I(T) = \sum_{t \in \tilde{T}} I(t) ,$$

where $I(t) = i(t)p(t)$ denotes the impurity in node t (Breiman *et al.*, 1984). By selecting the split s that maximizes $\Delta i(s,t)$ we are also increasing the purity in T. To explain this, let T be a tree and split node t in T into t_L and t_R (as in Figure 1.2). Thus we have a new bigger tree, say T_1. Then T_1 has impurity

$$I(T_1) = \sum_{t \in \tilde{T}} I(t) + I(t_L) + I(t_R)$$

and the decrease in impurity at node t is

$$\Delta I(s,t) = I(t) - I(t_R) - I(t_L).$$

Let

$$p_L = p(t_L)/p(t) \text{ and } p_R = p(t_R)/p(t)$$

then

$$\begin{aligned}\Delta I(s,t) &= i(t)p(t) - i(t_R)p(t_R) - i(t_L)p(t_L) \\ &= i(t)p(t) - i(t_R)p_R\,p(t) - i(t_L)p_L\,p(t) \\ &= \{i(t) - i(t_R)p_R - i(t_L)p_L\}\,p(t) \\ &= \Delta i(s,t)p(t).\end{aligned}$$

Now

$$\Delta I(T) = \sum_{t \in \tilde{T}} \Delta I(s,t) = \sum_{t \in \tilde{T}} \Delta i(s,t)p(t),$$

which means that if $\Delta I(s,t)$ is maximized, the decrease in impurity for the whole tree T will be maximized. It is therefore clear that a tree will become purer if a node becomes purer for a split s, $s \in S$ (S is the set of all possible splits).

2.3 GOODNESS-OF-SPLIT CRITERION

The goodness-of-split criterion is a measure used in the splitting process at each non-terminal node. This criterion is responsible for decreasing the node impurity as defined above. Vast literature on different goodness-of-split criteria exists, such as the Entropy criterion, Chi-square measure, G-square measure, Deviance functions, Gini criterion, Symmetric Gini, Twoing criterion and ordered Twoing. These criteria all differ, but have one common goal and that is to isolate the classes in $\mathcal{L}$. Only the Gini criterion, Symmetric Gini, Twoing criterion and ordered Twoing will be discussed. These goodness-of-split criteria have been formulated so that $\Delta i(s,t) \geq 0$.

2.3.1 The Gini and Symmetric Gini criterion

The ***Gini criterion*** is a measure of node impurity. It takes the biggest class in the data set and tries to isolate it from the rest. Once this is done, the Gini proceeds by pulling out the second biggest class and then the third biggest etc., until each class appears in a separate node. Thus the Gini criterion tries to separate one class at a time in its effort to obtain the purest terminal nodes. The Gini criterion is defined as follows:

$$i(t) = 2p(1 \mid t)p(2 \mid t)$$

in case of two classes (1 and 2), but is generalized to

$$i(t) = \sum_{j \neq i} p(j \mid t)p(i \mid t) = 1 - \sum_{j} p^2(j \mid t)$$

for more than 2 classes. By minimizing this impurity function, the Gini criterion will cause the descendant nodes of t to be purer than t. The Gini criterion is the estimated probability of misclassification and if it is small, the purity in the descendant nodes is maximized.

The ***Symmetric Gini criterion*** works similar to the Gini criterion, but incorporates the misclassification costs, whereas the Gini criterion works best with unit misclassification costs. When misclassification costs are used, the decrease in impurity for the Gini criterion may be negative for some splits (Breiman *et al.*, 1984). Since this is an undesirable property, it is better to choose the symmetric Gini option.

2.3.2 The Twoing and Ordered Twoing criterion

The ***Twoing criterion*** uses a different approach from the Gini. It segments the classes into two groups, called superclasses. In these two superclasses, the Twoing criterion groups together classes with similar characteristics. The two superclasses are most dissimilar, but within in each group the classes are very similar. This process is carried out at each non-terminal node and the split is done as if there were only two classes in the whole data set. The following explains the splitting process more clearly. Let C denote the classes of the response in a data set, such that $C=\{1, \ldots, J\}$, which has no natural order (such as gender for example). At each node, separate the classes into two superclasses C_1 and C_2, where

$$C_1=\{1, \ \ldots, j_n\} \text{ and } C_2=C\text{-}C_1$$

C_1 and C_2 are dissimilar, but the classes $(1, \ldots, j_n)$ in C_1 are similar as well as the classes in C_2. Now for any split s, compute $\Delta i(s,t)$ as though there are only two classes. Find the split for which $\Delta i(s,t)$ is a maximum. Because there are a number of possible superclasses of two, we need to find the superclasses that maximize $\Delta i(s,t)$. So this process is an iterative process where we start out with two superclasses, compute its decrease in impurity, select two other superclasses, compute its decrease in impurity etc., until all possible superclasses have been evaluated. The choice of superclasses is finally those that minimize $i(t)$ or equivalently maximize the $\Delta i(s,t)$.

It is clear that the Twoing criterion attempts to isolate each class as the tree is growing so that in the end we have the purest nodes. Each node is divided into two superclasses that are similar within and if the process continues that way, we will end up with only one class in a node, which is then declared terminal. When there are J classes, the number of finite superclasses of two is 2^{J-1}, which increases as J increases. Because we are splitting as if we have two classes, the Twoing criterion formula is defined as follows. For any node t and a given split s of t into t_L and t_R, the Twoing criterion has the formula

$$\Delta i(s,t) = \frac{p_L p_R}{4}\{\sum_j | p(j \mid t_L) - p(j \mid t_R) |\}^2 ,$$

and the best split is obtained by maximizing this formula. p_L and p_R are the proportions of cases in node t that goes into nodes t_L and t_R respectively, $p(j|t_L)$ and $p(j|t_R)$ are the proportions of cases in class j in descendant nodes t_L and t_R. A more formal explanation of this formula can be found in Breiman *et al.* (1984). The superclass C_1 for a split s that maximizes $\Delta i(s,t)$ is

$$C_1(s) = \{j: p(j|t_L) \geq p(j|t_R)\} \quad \text{and} \quad C_2(s) = C - C_1(s)$$

The Twoing criterion is a measure of the decrease in node impurity $[\Delta i(s,t,C_1)]$.

The ***Ordered Twoing criterion*** works in a similar fashion, but in this case the categories of C has a natural order. An example of such a case is where the classes are low risk, moderate risk and high risk clients applying for a home loan. The restriction in this case is on the choice of C_1 and C_2 such that $C_1=(1, \ldots, j)$ and $C_2=(j+1, \ldots, J)$. The ordered Twoing has no impurity function.

2.4 MINIMAL COST-COMPLEXITY PRUNING

Minimal cost-complexity pruning is due to Breiman *et al.* (1984) who suggested that it is better to grow an overly large tree on the data and to prune it upwards to obtain the best[1] subtree. Let T_{max} denote the maximal grown tree. The minimal cost-complexity measure $(R_\alpha(T))$ for a subtree $T \leq T_{max}$ is given by the linear combination

$$R_\alpha(T) = R(T) + \alpha\widetilde{T},$$

where $R(T)$ is the resubstitution error estimate for T, $\widetilde{T}$ is the complexity (the number of terminal nodes in T), and α is called the complexity parameter (α is any real number ≥ 0). It is clear from the above, that $R_\alpha(T)$ is obtained by adding a cost penalty for complexity $(\alpha\widetilde{T})$ to $R(T)$. For different values of α, the subtree $T \leq T_{max}$ need to be found that minimizes the cost-complexity measure. Therefore, we need to find

$$R_\alpha(T(\alpha)) = \min_{T \leq T_{max}} R_\alpha(T).$$

Since T_{max} has a finite number of terminal nodes, it also has a finite number of subtrees and by increasing the value of α, $T(\alpha)$ becomes the minimizing subtree. Thus, at a certain value of α, some splits make no difference in the error rate, and might as well be ignored and be pruned away. As α increases, another minimization point is reached at say α' and now $T(\alpha')$ becomes the new subtree that minimizes the complexity measure. So the process continues by increasing α, until a number of subtrees are obtained and eventually one is left with the root node $\{t\}$ of T_{max}. Although the pruning may sound relatively straightforward, minimal cost-complexity asks the following questions during the pruning process.

i. Is there a unique subtree $T \leq T_{max}$ that will minimize $R_\alpha(T)$?
ii. By increasing α, a number of subtrees, $T_1, T_2, T_3, \ldots, \{t\}$ are obtained. Is each tree derived by pruning upward from the previous subtree? Thus, do the nested sequence of subtrees $T_1 > T_2 > T_3 > \ldots > \{t\}$ hold?

In their explanation of the minimal cost-complexity pruning, Breiman *et al.* (1984)

[1] Another method of obtaining the desired tree is through recursive shrinking (Chambers and Hastie, 1997).

defined the smallest minimizing subtree $T(\alpha)$ for complexity parameter α by the following two conditions:

a) The best subtree $T(\alpha)$ for α is the one that minimizes $R_\alpha(T)$, i.e.

$$R_\alpha(T(\alpha)) = \min_{T \leq T_{\max}} R_\alpha(T).$$

b) If $T(\alpha)$ is a subtree that minimizes $R_\alpha(T)$, then $T(\alpha)$ is a subtree of T, i.e.

$$\text{if } R_\alpha(T) = R_\alpha(T(\alpha)), \text{ then } T(\alpha) \leq T.$$

If a subtree such as $T(\alpha)$ exists, then it must be unique and the subtree sequence $T_1 > T_2 > T_3 > \ldots > \{t\}$ will hold.

The starting point for proving that for every value of α there exists a smallest minimizing subtree, is subtree T_1, which is the subtree of T_{max} that satisfies the condition $R(T_1)=R(T_{max})$. Thus the subtree T_1 has the same resubstitution error rate as T_{max}. A hypothetical tree is created in Figures 2.1a and 2.1b for illustration purposes. Consider Figure 2.1a. To get subtree T_1 from T_{max}, let t_L and t_R be any two terminal nodes from immediate ancestor t. For such a split to be worthwhile, the resubstitution estimate at t $\{R(t)\}$ must be greater than or equal to the sum of the resubstitution estimates at t_L and t_R $\{R(t_L)+R(t_R)\}$. Should $R(t)=R(t_L)+R(t_R)$, the split is unnecessary, hence we could prune off t_L and t_R. If all such terminal nodes are pruned off, the resulting tree is called T_1. Figure 2.1b is subtree T_1. Consider the branch T_t (Figure 2.1b) and define its resubstitution error rate as

$$R(T_t) = \sum_{t' \in \tilde{T}_t} R(t'),$$

where $\tilde{T}$ is the set of terminal nodes in T_t and $R(t')$ is the resubstitution error rate in all the terminal nodes in that branch. Since $R(T_1)=R(T_{max})$ it is clear that for a non-terminal node t in Figure 2.1b, $R(t)>R(T_t)$ otherwise branch T_t would have been pruned off in the establishment of T_1. Let $\{t\}$ be the root node for branch T_t and set

$$R_\alpha(\{t\}) = R(t) + \alpha. \qquad \text{(note that } \tilde{T} = 1 \text{ because } \{t\} \text{ is the root node)}$$

For the branch T_t let

$$R_\alpha(T_t) = R(T_t) + \alpha\tilde{T}_t$$

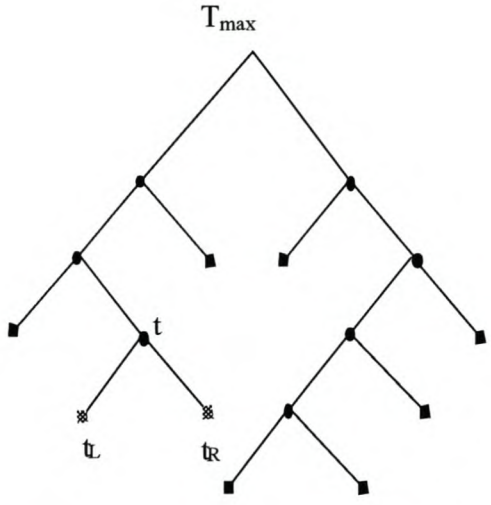

FIGURE 2.1a: *T_{max} with split at node t.*

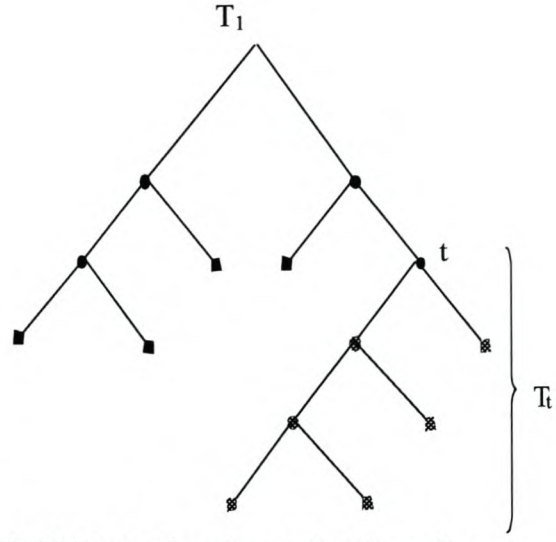

FIGURE 2.1b: *T_1 is the result of T_{max} after pruning away t_R and t_L.*

As long as $R_\alpha(T_t) < R_\alpha(\{t\})$, it is worthwhile keeping the branch T_t, but when $R_\alpha(T_t)=R_\alpha(\{t\})$ that branch is no longer needed and can be pruned off. Minimal cost-complexity pruning thus proceeds by cutting off the weakest link which is the cases where $R_\alpha(T_t)\rightarrow R_\alpha(\{t\})$. At some critical value of α these two rates are equal and this value can be determined by solving the inequality $R_\alpha(T_t) < R_\alpha(\{t\})$. Thus $R(T_t)+\alpha\widetilde{T}_t < R(t)+\alpha$, which becomes

$$\alpha < \frac{R(t)-R(T_t)}{\widetilde{T}_t - 1}$$

Now define the function $g_1(t)$, $t \in \widetilde{T}$, by

$$g_1(t) = \begin{cases} \dfrac{R(t)-R(T_t)}{\widetilde{T}_t - 1}, t \notin \widetilde{T}_1 \\ +\infty, t \in \widetilde{T}_1 \end{cases}$$

Using this function, the weakest link (defined as node $\bar{t}$, a non-terminal node) of all the non-terminal nodes in T_1 is the node that minimizes this function $g_1(t)$, thus

$$g_1(\bar{t}) = \min_{t \in \widetilde{T}_1} g_1(t)$$

This now becomes the case where minimal cost-complexity pruning takes place, since the node $\bar{t}$ becomes preferable to branch $T_{\bar{t}}$, which has $\bar{t}$ as the root node. Pruning this

branch results in a new tree which is T_2 and $T_2 = T_1 - T_t$. From T_2 the next weakest link can be pruned off in the same fashion. Pruning off all the weakest links will eventually leave the root node of T_{max}. Thus the objective of obtaining a unique subtree that minimizes $T(\alpha)$ and obtaining the decreasing sequence of subtrees $T_1 > T_2 > T_3 > \ldots > \{t_1\}$ where each tree is a subtree of the previous, has been obtained by calculating $g(t)$ and pruning away the link where $\min(g(t))$ is found.

2.5 SELECTING THE BEST PRUNED OR OPTIMUM–SIZED SUBTREE

The method of minimal cost-complexity pruning results in a sequence of subtrees $T_1 > T_2 > T_3 > \ldots > \{t_1\}$. The problem the analyst now faces, is selecting the best tree from these subtrees. To solve this problem, the error rate of each subtree is estimated to test how accurate the subtrees are in terms of their classification power. By using the estimated error rate, all the subtrees can be compared to find the optimal one, and the optimal/ best one is defined as the subtree that minimizes the error rate. A number of cross-validation methods exist in the literature but this text will only explain the derivation of V-fold and test sample cross validation error rate estimation. Since these error rates will eventually determine the final tree, it can be seen as an aid in pruning the maximal grown tree.

General definitions for estimation of error rates:

i. $Q^*(i \mid j) = P(d(\boldsymbol{X}) = i \mid class = j)$

so that $Q^*(i|j)$ is the probability that a case in class j is classified into class i by classifier d.

ii. $R^*(j) = \sum_i C(i \mid j) Q^*(i \mid j)$

so that $R^*(j)$ is the expected cost of misclassification for class j cases.

iii. $R^*(d) = \sum_j R^*(j)\pi(j)$

as the expected misclassification cost for the classifier and $\pi(j)$ are the prior probabilities.

2.5.1 Derivation of test sample (ts) error estimate, $R^{ts}(T_k)$

When the data set is large enough, the better procedure to estimate the error rate is to use an independent test sample. To obtain such an estimate, the data set $\mathcal{L}$ is partitioned into two subsets, $\mathcal{L}_1$ and $\mathcal{L}_2$ which respectively forms the learning sample and the test sample.

$\mathcal{L}_1$ = Learning sample

$\mathcal{L}_2$ = Test sample having number of cases equal to $N^{(2)}$

An overly large tree T_{max} is grown on $\mathcal{L}_1$ and pruned upward, using minimal cost-complexity pruning, giving the sequence of trees $T_1 > T_2 > T_3 > \ldots > \{t_1\}$ for different α values. This is called the $\{T_k\}$ sequence of subtrees meaning that there are k subtrees to be evaluated. For every tree in $\{T_k\}$, calculate the estimated error rate, $R^{ts}(T_k(\alpha))$. This is done by sifting $\mathcal{L}_2$ through each of the subtrees and obtaining the number of misclassifications for each tree. Let $N_j^{(2)}$ be the number of class j cases in $\mathcal{L}_2$. For any one of the trees $T_1, T_2, T_3, \ldots, T_{(k)}$ take $N_{ij}^{(2)}$ to be the number of class j cases in $\mathcal{L}_2$ that has been classified as i. This implies that cases of class j that have been misclassified, are counted. Then the estimate

$$Q^{ts}(i \mid j) = N_{ij}^{(2)} \Big/ N_j^{(2)}$$

is the test sample estimate of $Q^*(i|j)$, which is the proportion of cases in class j in the test sample that has been classified as class i. The *ts* estimate of $R^*(j)$ is given by the sum of the cost of misclassification multiplied by the proportion of cases in class j classified as i, i.e.

$$R^{ts}(j) = \sum_i C(i \mid j) Q^{ts}(i \mid j).$$

Finally, the *ts* estimate for $R^*(d)$ is

$$R^{ts}(d) = \sum_j R^{ts}(j)\pi(j).$$

If the proportion of class j in $\mathcal{L}_2$ is used to represent $\pi(j)$, then $\pi(j) = N_j^{(2)} \Big/ N^{(2)}$. Since we are using a tree (T) as a classifier, $R^{ts}(d)$ can by replaced by $R^{ts}(T(\alpha))$, calculated from the minimal cost-complexity tree $T(\alpha)$. Using $\pi(j)$ as above, the *ts* error rate estimate becomes

$$R^{ts}(T(\alpha)) = \frac{1}{N^{(2)}} \sum_{i,j} C(i \mid j) N_{ij}^{(2)},$$

which is the misclassification cost if every case in $\mathcal{L}_2$ is dropped through tree $T(\alpha)$. If the analyst gives the value of $\pi(j)$, the above formula will change slightly to take into account the effect thereof.

Selecting the best tree is then accomplished by means of the following rule:

$$R^{ts}(T_{ko}(\alpha)) = \min_{k} R^{ts}(T_k(\alpha)),$$

meaning that the subtree $(T_{ko}(\alpha))$ that minimizes the error rate for all the minimal cost-complexity subtrees $T_1 > T_2 > T_3 > \ldots > \{t_1\}$ is the best.

2.5.2 Derivation of V-fold cross-validation (CV) error estimate, $R^{cv}(T_k)$

In cases where the data set is too small to be divided into a learning and test sample, V-fold cross validation becomes the preferred way of estimating the error rate. V-fold cross-validation is computationally very intensive, but it makes effective use of all the cases in the data set. With V-fold cross-validation, the learning sample $\mathcal{L}$ is divided into V approximately equal sized subsets, $\mathcal{L}_v$, $v=1, \ldots, V$. The new learning sample now becomes $\mathcal{L}^{(v)}$, where

$$\mathcal{L}^{(v)} = \mathcal{L} - \mathcal{L}_v.$$

This is called the v-th learning sample and $\mathcal{L}_v$ the test sample. V auxiliary trees are grown using $\mathcal{L}^{(v)}$, $v=1, \ldots, V$. Starting by growing V overly large trees, $T^{(v)}_{max}$, on $\mathcal{L}^{(v)}$ and simultaneously growing T_{max} on $\mathcal{L}$, the following procedure describes the error rate estimation of V-fold cross validation, denoted by $R^{CV}(T_k)$.

Let $T^{(v)}(\alpha)$ be the minimal cost-complexity subtree of $T^{(v)}_{max}$ grown on $\mathcal{L}^{(v)}$ and let $T(\alpha)$ be the minimal cost-complexity subtree of T_{max} grown on $\mathcal{L}$. Remember that the cases in $\mathcal{L}_v$ have not been used in growing the v-th tree. Now sifting $\mathcal{L}_v$ through $T^{(v)}(\alpha)$, we can obtain the number of cases misclassified in subtree v and define it as:

$N^{(v)}_{ij}$ = the number of class j cases in $\mathcal{L}_v$ classified as i by $T^{(v)}(\alpha)$ and set

$N_{ij} = \sum_{v} N^{(v)}_{ij}$, which is the total number of class j cases classified as i

The total number of class j cases in all the $\mathcal{L}_v$ test samples is N_j, and is also the number of class j cases in $\mathcal{L}$. For $\mathcal{L}^{(v)}$ large $T^{(v)}(\alpha)$ and $T(\alpha)$ should have more or less the same classification accuracy and the estimate of $Q^*(i|j)$ for $T(\alpha)$ is

$$Q^{CV}(i \mid j) = N_{ij} / N_j$$

Furthermore, the estimate of $R^*(j)$ for $T(\alpha)$ is

$$R^{CV}(j) = \sum_i C(i \mid j) Q^{CV}(i \mid j) \text{ and } R^{CV}(T(\alpha)) = \sum_j R^{CV}(j)\pi(j),$$

where $\pi(j) = N_j / N$ is estimated from $\mathcal{L}$. For this case, the error rate for V-fold cross-validation now becomes

$$R^{CV}(T(\alpha)) = \frac{1}{N} \sum_{i,j} C(i \mid j) N_{ij}.$$

Selecting the best tree is then obtained where

$$R^{CV}(T_{ko}(\alpha)) = \min_k R^{CV}(T_k(\alpha)),$$

meaning that the minimal cost-complexity subtree that minimizes this error rate is the best. The calculation of this error rate is very complex and only N_{ij} is derived from using $\mathcal{L}_v$ while the other values N and N_j are obtained from $\mathcal{L}$. The simulation studies done by Breiman *et al.* (1984) show that cross validation estimates seem to fall close to optimal in terms of minimizing the error rate. If V is too small, the estimate $R^{CV}(T)$ tends to be less accurate and also tends to be overestimating the true error rate. When V=N, the situation becomes the N-fold or leave-one-out cross validation with one case left out every time as a test sample. Breiman *et al.* (1984) found that V=10 gives adequate accuracy of the error rate. Smaller values of V have also done very well in estimating the error rate.

The assumption that is made when doing V-fold cross-validation is that the process is stable. This means that the trees grown on $\mathcal{L}^{(v)}$, are constructed using almost all of $\mathcal{L}$ and thus have a reasonably unbiased error rate $R^{CV}(T^{(v)}(\alpha))$ almost equal to $R^{CV}(T(\alpha))$, $R^{CV}(T^{(v)}(\alpha))$ being the error rate from the minimal cost-complexity subtree $T^{(v)}(\alpha)$ and $R^{CV}(T(\alpha))$ being the error rate for the minimal cost-complexity subtree $T(\alpha)$. Therefore the assumption is that $T^{(v)}(\alpha)$ and $T(\alpha)$ behave in the same fashion. This assumption becomes valid when V is large or alternatively $\mathcal{L}^{(v)}$ is large.

2.6 THE STANDARD ERROR RULE FOR TREE SELECTION

The standard error (SE) rule is an alternative way of selecting the desired classification tree. It may be the case that the optimal tree is not acceptable. For example, it might be too large for practical purposes. To overcome this problem the SE rule can be very helpful. Using the frequentist approach to calculating the SE of any random variable, we need to make an assumption about its probability distribution. Using a classification tree T, a case can be classified correctly or misclassified. Let this binary classification be the random variable. Therefore, we can now make the assumption that classification has a binomial distribution[2] and we let the misclassifications represent the successes. In the case where we have a learning sample $\mathcal{L}_1$ and a test sample $\mathcal{L}_2$, we can now drop the $N^{(2)}$ cases in $\mathcal{L}_2$ through T. The probability that any single case will be misclassified, is $R^{ts}(T)$. We now have a binomial distribution consisting of $N^{(2)}$ independent trials with probability $R^{ts}(T)$ of a success. Using $R^{ts}(T)$ as an unbiased estimate for the probability of a success, the SE for $R^{ts}(T)$ can be derived as:

$$SE(R^{ts}(T)) = \sqrt{R^{ts}(T)\{1-R^{ts}(T)\}/N^{(2)}}\,.$$

Using the SE rule, the analyst can search for a more preferable tree, which is the one where

$$R^*(T_k) \le R^*(T_{ko}) + \beta SE(R(T_{ko}))\,.$$

$R^*(T_k)$ can be either $R^{ts}(T_k)$ or $R^{CV}(T_k)$ depending on the error rate estimator that have been chosen. $R^*(T_{ko}) = \min_k R^*(T_k)$ and β (usually 1 or 2) is a constant. The newly selected best tree now becomes the tree with the next smallest error rate that falls within for example 1 SE from the optimal tree error rate, derived by minimal cost-complexity pruning. Using this rule, the analyst will end up with a less complicated tree but with a bigger error rate.

[2] See Mendenhall *et al.* (1990), (Chapter 3) as a reference on how a binomial distribution is derived.

2.7 PRIORS [π(j)] AND COST OF MISCLASSIFICATION [C(i|j)]

During our previous discussions, priors and cost of misclassification was mentioned but it was not fully explained what real purpose they serve in the selection of the tree. However for practical applications these two factors play an important role in obtaining a desired classification tree and changing them will affect the tree growing dramatically. Fundamentally, the main purpose of a tree is to classify a case as accurately as possible. In practice, classification may not be as accurate as expected, thus we would want to minimize costs if such a case should occur. For this purpose $C(i|j)$ plays a vital role in selecting the best tree. For example if the cost of classifying a high risk client as a low risk client is high it can be taken into account by specifying $C(i|j)$ and the final classifications may be less catastrophic. When the number of cases in classes are unequal, the tree may tend to classify cases in the biggest class better and cases in the smallest class worse. Therefore, the tree will give us more detail on the big class and almost none on the smaller class. By specifying $\pi(j)$ appropriately, this effect can be diminished. We could thus see $\pi(j)$ as a way of adjusting the importance of misclassifications in the classes. By giving the priors, you are actually specifying how likely it is that a case will fall into a class. For example if high risk clients have higher priors than low- and moderate risk, a case are more likely to be classified as high risk. Priors can be estimated from the learning sample, test sample or if prior knowledge based on past experience exists, the analyst can specify them. The priors are the assumed class distribution before the analysis.

2.8 CONCLUSION

Selecting the best tree using minimal cost-complexity alone may not always result in a tree desired by the analyst. Firstly, the tree might be too large and complex and secondly, the tree may not do very well in classifying a data set containing completely new cases. A tree is grown on a learning sample and does not guarantee that the best tree on that sample will be the best tree for future classification. The theory of trees may sound simple in selecting the best tree, but the bias-variance trade-off phenomenon plays an important role as well. A tree should not classify the learning sample cases very well, but

new cases are poorly classified. This is the case when the tree is too big. A tree should also not be too small so that it contains fairly no structure and is too simple. Thus a classification tree should be big enough to have a good structure but also small enough to accommodate the variance in a new set of cases in order to make accurate classifications.

In calculating the test sample error rate estimate, the test sample is obtained by splitting the data into two groups. The cases in these two groups have an equal chance of being in either group. The proportion of the classes in each group plays an important role in growing the tree as well as obtaining an unbiased estimate of the error rate. If these class proportions are not well represented in each of the groups, we will have some bias. For a small number of classes in a large data set, this should not be such a serious problem. But as the number of classes gets bigger and the data set gets smaller it will create a problem. For V-fold cross-validation this is also true. Some common questions that analysts frequently ask are:

- How big should V be, so that $\mathcal{L}^{(v)}$ can be just as good as $\mathcal{L}$ and the error estimate still accurate enough for different data set sizes?
- How big should $\mathcal{L}$ be before a tree analysis should be considered?
- How big should $\mathcal{L}_1$ and $\mathcal{L}_2$ be when estimating the test sample error rate?
- How can we ensure that all the classes are well represented in the learning and test sample?

It seems that the decision for the best/ optimal tree, still depends on the experience and knowledge of the analyst, as well as the specific environment in which the classification research is being done.

CHAPTER 3

Resampling techniques used in Trees

3.1 INTRODUCTION

In Chapter 2 we explained single tree methodology. We also demonstrated how the optimal single tree as a classifier can be obtained. In the following sections we move away from this approach and introduce multiple trees as the new classifier, instead of one tree. Each of the multiple trees is still grown separately according to the methodology in Chapter 2, but together they form the classifier. Using multiple trees no longer results in a nice visually appealing classifier, but it exhibits a kind of "black-box" phenomena (Figure 3.1) where one enters the data, grow the trees and gets the results of the classification. However, it is possible to view the individual trees.

Classification trees have been proven to yield unstable results (Breiman, 1998). Taking sub-samples from $\mathcal{L}$ and growing trees on them may give completely different trees although these samples come from the same population. This phenomenon of high variance creates confusion as to which single tree is the right one. Consequently, further developments on the basic classification tree were done to give the tree a more stable performance. The use of resampling techniques was introduced where a single tree is no longer used, but a committee of experts (multiple trees). Since we only have one learning sample, each tree in the committee is grown on data resampled from $\mathcal{L}$ in a bootstrap[1] fashion with replacement of cases. Figure 3.1 provides an illustration of how the resampling process works. Let B denote the number of bootstrap samples. Firstly, B bootstrap samples are drawn from $\mathcal{L}$. Secondly, B trees are grown on them and all B trees vote simultaneously for the group membership of a case.

[1] Efron and Tibshirani (1993) is a good reference for a discussion on the bootstrap theory. The bootstrap method is due to Stanford University Professor, Bradley Efron.

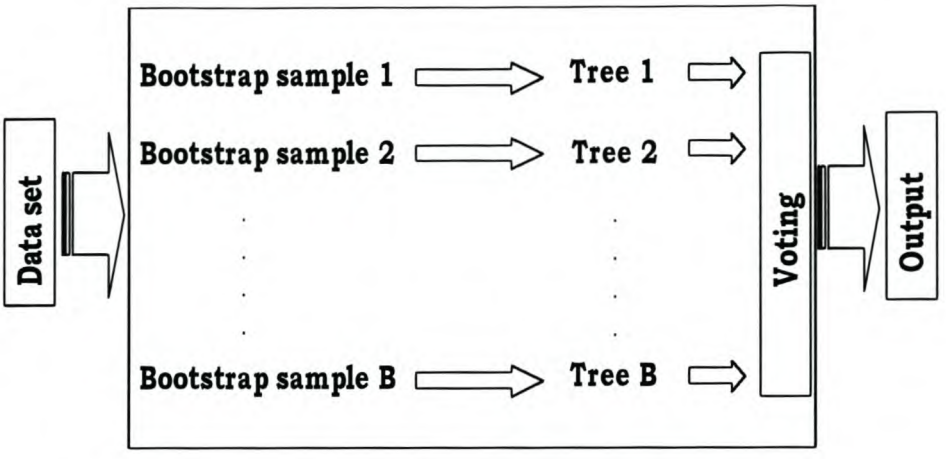

FIGURE 3.1: *The "black-box" of multiple classification trees.*

Finally, the membership that has the majority vote of the committee of experts is selected as the output.

The remainder of this chapter is a discussion revolving around two resampling methods found in the CART software. Bagging (**B**ootstrap **agg**regat**ing**) is a resampling technique proposed by Breiman (1996). ARCing (**A**daptive **R**esampling and **C**ombin**ing**) is another resampling technique proposed by Breiman (1998). ARCing is a term introduced by Breiman for his form of Boosting[2] incorporated in the CART software. However, Boosting was first popularized by Yoav Freund and Robert Schapire with the introduction of their "AdaBoost.M1" algorithm (Hastie *et al.*, 2001). We will simply refer to AdaBoost.M1 as AdaBoost. Bagging and ARCing are similar, except that in ARCing, higher weights are applied to the misclassified cases prior to each successive bootstrap sample.

Both techniques may yield better results than a single tree classifier but improvement is not guaranteed. One of its main purposes is to reduce the high variance of unstable trees. These techniques can be used with other unstable classification methods or learning algorithms, such as neural networks. They could also be used with regression problems, such as regression trees. The use of resampling techniques increases the

[2] The motivation for Boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee"-Hastie *et al.* (2001). From this concept, Bagging and ARCing originated.

computation time of classification methods and is a time consuming process with large data sets. Using them also sacrifices the tree interpretability.

3.2 BAGGING TREES

Bagging refers to the creation of a committee of experts using the same algorithm on bootstrap samples, sampled with replacement from a single learning sample. Let each bootstrap sample generated from $\mathcal{L}$ be denoted by $\mathcal{L}^{[b]}$, where $b=1, \ldots, B$. In Bagging, each $\mathcal{L}^{[b]}$ is drawn from $\mathcal{L}$ in an identical way with replacement. Let $\mathcal{L}$ have N cases ($n=1, \ldots, N$). $\mathcal{L}^{[b]}$, also having N cases, is drawn randomly with each case in $\mathcal{L}$ having equal probability $p(n) = \frac{1}{N}$ of being in a bootstrap sample. Let $T^{[b]}$ denote the tree grown on $\mathcal{L}^{[b]}$. If B bootstrap samples ($\mathcal{L}^{[1]}, \mathcal{L}^{[2]}, \ldots, \mathcal{L}^{[B]}$) are drawn, then a set of B trees of the form $T^{[1]}, T^{[2]}, \ldots, T^{[B]}$ are grown on them. This series of trees are called a committee of experts and by letting each tree vote for a specific case, the class membership of that case is obtained.

Breiman (1996) demonstrated that by using this form of classification, better results are obtained than for a single classification tree. Especially when single trees are unstable or otherwise put, when data sets are messy. Having a mixture of data types surely affects the final tree when the data set is slightly perturbed. For example, if a few cases are left out from the data, the classification tree may be completely different. Since resampling are perturbations in the learning sample, each tree in the committee of experts will differ, but by using their majority vote the effect of the difference in trees are eliminated. Therefore, stability is achieved. Hastie *et al.* (2001) illustrates the effect of Bagging and shows how the perturbations change the tree results. Breiman (1998) showed that with clean data sets perturbations in the learning sample do not affect the classification results significantly. Thus clean data produce stable classifiers, therefore Bagging and ARCing (Section 3.3) is not necessary.

Because modeling mixtures of data types is possible with trees, Bagging can help a lot in improving the classification results. Improvement will depend on how dramatic the perturbations in each bootstrap sample are. In his paper *"Bagging predictors"*, Breiman (1996) demonstrates, by using examples, why this procedure reduces instability in trees. Bauer and Kohavi (1999) show the Bagging algorithm.

3.3 BOOSTING (ARCING) TREES

In ARCing, each bootstrap sample drawn from $\mathcal{L}$ depends on the previous sample. The motivation for this is to increase the probability of classifying cases correctly that have been misclassified in the previous sample. To achieve this, higher weights are applied on cases that are more frequently misclassified and this process is repeated at each successive sample. Let $\mathcal{L}$ have N cases ($n=1,\ldots, N$) and put the probability of each case $p(n) = \frac{1}{N}$ equal to weight w. Now sample one case, N times with replacement from $\mathcal{L}$, resulting in bootstrap sample $\mathcal{L}^{[1]}$. Grow tree $T^{[1]}$ on $\mathcal{L}^{[1]}$ and obtain the misclassified cases by sifting $\mathcal{L}^{[1]}$ through $T^{[1]}$.

Define a function $\psi > 1$ which depends on the misclassification rate of the tree. If the n-th case in $\mathcal{L}^{[1]}$ is misclassified, then put $w=\psi^*[p(n)]$ on that case and all correctly classified cases has $w= p(n)$. Now obtain the sum of w for the N cases $\{\sum_{n=1}^{N} w_n\}$ and divide each w by this sum to get the updated probability of a case being in the next bootstrap sample from $\mathcal{L}$. The misclassified cases thereby obtain a higher probability of being in the next sample and the correctly classified cases have a lower probability. We can now draw sample $\mathcal{L}^{[2]}$ from $\mathcal{L}$ using this updated probability information and grow tree $T^{[2]}$ on $\mathcal{L}^{[2]}$. Obtain the misclassified cases in $\mathcal{L}^{[2]}$ using $T^{[2]}$ and update w for the n-th misclassified case in $\mathcal{L}^{[2]}$. Calculate $\sum_{n=1}^{N} w_n$ again, and dividing each w by that sum, gives the new updated probability of a case being in the next bootstrap sample. The sequence continues until the last bootstrap sample is generated. Thus we have B bootstrap samples $\mathcal{L}^{[1]}, \mathcal{L}^{[2]},\ldots, \mathcal{L}^{[B]}$ of equal size denoted by $\mathcal{L}^{[b]}$ ($b=1,\ldots, B$) and B classification trees $T^{[b]}$ from these samples. Each successive tree is forced to concentrate more on the observations that are misclassified by the previous tree.

The resulting set of trees is the committee of experts and is used to classify a new case by voting. Each case will be assigned to the class receiving the majority of the committee votes. ARCing results are equivalent to that of Bagging. In some instances, ARCing might do better than Bagging and vice versa. Opitz and Maclin (1999) did empirical comparisons of AdaBoost, ARCing and Bagging using a neural network and classification tree. They showed that these methods do reduce the error rates and that a

neural network may do better then a classification tree. In his paper *"ARCing classifiers"*, Breiman (1998) demonstrates ARCing and uses examples to show why this procedure works. Using simulated data, he showed that the variance of trees decreases a lot with the use of ARCing and Bagging. This paper also contains the ARCing algorithm and explains how the updating function works. For the interested reader, Hastie *et al.* (2001) and Freund and Schapire (1999) give more detail on how the AdaBoost algorithm works.

3.4 DOES BAGGING AND BOOSTING INCREASE CLASSIFICATION POWER?

In this section, we use some sample data sets in the CART® software to demonstrate the effect of Bagging and Boosting in classification. Pruning with a committee of experts is not advised, but can be done. The reason for no pruning is that the multiple trees now vote for a class membership. In the case of the single tree, the bias-variance trade-off plays an important role and pruning is necessary to get a good unbiased classifier with reduced variance. Since multiple trees reduce variance, the effect of pruning is faded out. The default settings of CART was used in this experiment: the Gini splitting criterion, no pruning, unit misclassification costs, equal priors, 10% of the cases in each data set was randomly selected and left out as a test sample. The trees were grown on the other 90% of the data. For ARCing, the weights were updated by applying exponent =4 in the function ψ. The use of the exponent and its role in ψ will become clear later.

Figure 3.2 demonstrates the effect of Bagging and ARCing on four data sets (see Appendix B and Chapter 7 for more detail on the data sets). The initial tree in each bar chart is the single tree (when no Bagging or ARCing has been applied). It is clear that the overall error rate is the highest using a single tree. Resampling was introduced using a different number of trees, based on different numbers of bootstrap samples. The effect of Bagging and ARCing on the error rate is clearly visible and it increases the classification accuracy substantially. Bagging uses fewer trees than ARCing to reach its minimum error rate. As the number of trees increases, ARCing performs better than Bagging. However there is no certainty on how many trees will suffice for this type of analysis. Cases may also occur where these techniques perform worse than the single tree. Cases might occur

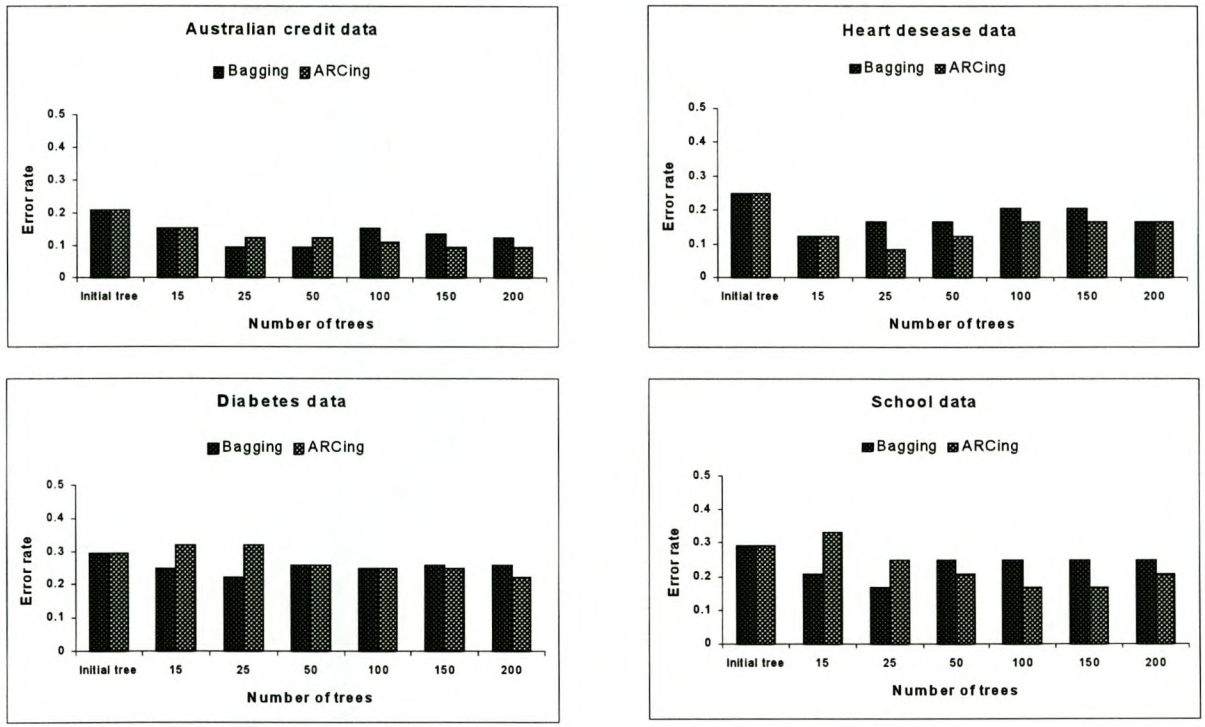

FIGURE 3.2: *The effect of Bagging and Boosting (ARCing) on the miclassification error rate at different number of classification trees. The error rate is equal to the number of misclassified cases divided by the total number of cases in the 10% test sample.*

when both perform equally better, or the same as the single tree. Although analysts such as Breiman (1998) and Opitz and Maclin (1999) have empirically shown that these techniques do work, it may not always be the case. The result is very much dependent on the nature of the data set.

In Breiman's variant ARCing, he modified Freund and Schapire's AdaBoost algorithm (Freund and Schapire, 1999) and introduced an exponent in the function ψ when updating the probability of a misclassified case being included in the next sample. Having B bootstrap samples and B trees, the updated probabilities in successive samples are defined in the ARCing algorithm by

$$p_n^{b+1} = \frac{(1+m_n^h)}{\sum_{n=1}^{N}(1+m_n^h)}.$$

For the n-th case in $\mathfrak{L}$, m_n refers to the number of times that case was misclassified by the previous b trees and p_n^{b+1} is the probability for selecting n to be part of next sample, $b+1$. Thus the form $1+m_n^h$ relates to function ψ, and h is the exponent emphasizing the weight placed on the cases more frequently misclassified. Increasing this exponent increases the weight, but does not guarantee decrease in the number of misclassifications (error rate). Based on empirical evidence, Breiman (1998) found h=4 to work very well, and this was the reason for putting the exponent=4 when the effect of ARCing was studied.

To study the effect of the exponent, five data sets were used at committee of experts of one hundred trees with varying numbers for the exponent (h). The results are displayed in Figure 3.3. The lines in Figure 3.3 are fairly horizontal, indicating that the error rate fluctuates around a constant equal to the average error rate for a specific data set. In some cases, increasing the exponent did decrease the error rate, but it appears to be less significant. However, the same conclusion is made here, that the effect of the exponent depends on the nature of the data set. The analyst should also experiment with different tree numbers in the committee of experts, since the exponent-trees interaction may have an effect on the results.

During this study, it was found that neither Bagging nor ARCing works in some cases when the learning sample is too small. With small learning samples, each randomly drawn bootstrap sample may not include some classes and some of the trees in the

committee of experts may not be able to classify a case correctly due to that. If there are only two classes and the learning sample is relatively small, some bootstrap samples may not contain the one class, thus cannot grow a tree or use a committee of experts. The same can happen for more than two classes. This finding demonstrates that trees are very data intensive classification methods.

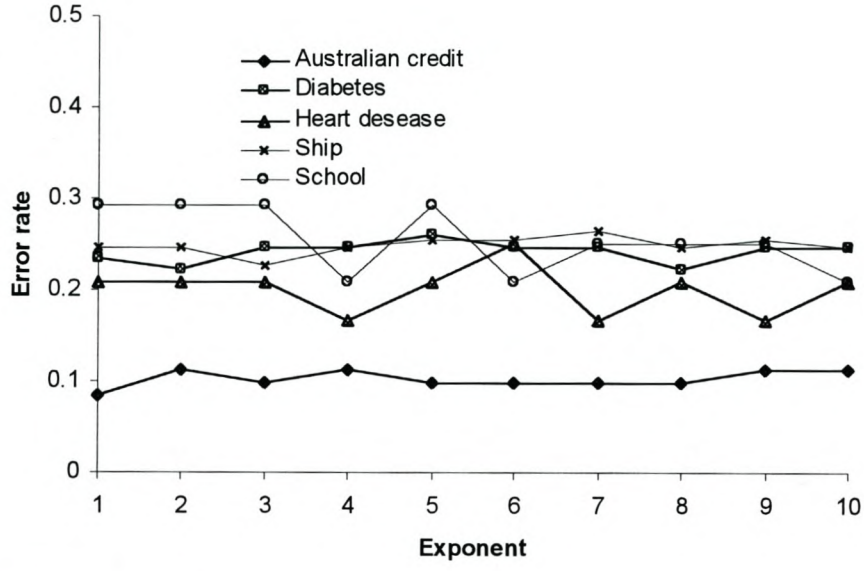

FIGURE 3.3: *The effect of the exponent on the error rate using 100 trees as the committee of experts. Error rate is the number of misclassifications divided by the total number of cases.*

CHAPTER 4

More developments on Trees

The following sections contain more developments on classification and regression trees introduced by Breiman (1996, 1998 and 2001) and Breiman *et al.* (1984). Combinations of variables are used to improve classification/ prediction. These techniques introduce further complications when interpreting the results. Surrogate splits are a very useful tool and provide more flexibility to the analysis. In Section 4.3 we discuss random forests, which are used similarly to Bagging and ARCing, but a different approach is used in establishing a classifier. These techniques will be briefly explained in the context of classification trees.

4.1 COMBINED VARIABLES AS SPLITTING RULES

Using multiple variables in a split is sometimes better than using a univariate split as described in Chapter 2. To increase the power of trees, combined variables can be used which may separate the classes better than a single variable. Breiman *et al.* (1984) found three useful combination rules to split classes: linear combination splits, Boolean combinations and through the addition of ad hoc combinations of variables (features).

When using ***linear combination splits***, only numerical variables that contain no missing values are used. Let x_m ($m = 1, \ldots, M$) be such a variable, and M the number of variables satisfying this condition. Then at every non-terminal node t, a set of coefficients $(a_1, a_2, \ldots, a_M)$ is calculated such that $\sum_{m=1}^{M} a_m^2 = 1$ and a split $s^*(a)$ of the form

$$\sum_{m=1}^{M} a_m x_m \leq c$$

needs to be found, where c ranges over all possible values. For $s^*(a)$, we denote the corresponding decrease in impurity as $\Delta i(s^*(a),t)$ and we define the best coefficients $(a^*_1, a^*_2, \ldots, a^*_M)$ as those $a_1, a_2, \ldots, a_M$ which maximize the decrease in impurity. Thus $\Delta i(s^*(a^*),t) = \max(\Delta i(s^*(a),t)$ and the best split $s^*(a^*)$ is of the form $\sum_{m=1}^{M} a^*_m x_m \leq c^*$. The form of such a split is very complex and as Breiman *et al.* (1984) states, such a search algorithm might get trapped in local maxima. This type of split gives a complicated tree structure with many variables. However, you can cut out some variables that contribute very little to the effectiveness of the split, by backward deletion[1] of variables. The use of linear combinations introduces a few disadvantages for trees. With single variable trees we have stated that monotone transformations of variables do not affect the structure of the tree. However, with linear combinations, this is no longer true. Combining variables creates trees that are difficult to interpret in the presence of high dimensionality. But if the data possess some linear structure, using linear combinations is better than univariate splits. Handling missing values in linear combinations is also a problem. An example of a linear combination split is $0.5height + 0.7weight + 0.51length \leq 25$ so that the squared coefficients add to 1.

A ***Boolean combination split*** of variables is very different from linear combinations. No linearity is involved, but rather a logical combination of the form: Does the case have property A and B? Does the case have property A and C? For example, *does a heart disease patient exercise regularly and is he over 45? Does a bank loan applicant have a high income and a permanent job?* These structures are called Boolean combinations and a large number of such splits exist when dealing with high dimensional data. Again here, only the variables containing no missing cases are considered. However this type of combinations maintain the invariance of the tree under monotone transformations of variables and its interpretation is also much simpler than using linear combination splits.

Using ***features*** of variables requires intense knowledge of the variables in the data set. The analyst can use variables and create new features that need not be linear or Boolean of nature. In the previous two cases, the splits were done using the given

[1] For further reading on this deletion process see Breiman *et al.* (1984) and CART® User's Guide.

variables. In the case of creating features, completely new variables are created and then the splits are done on those variables. Creating features is a powerful technique that can lead to better classification and better understanding of the data. The number of features is almost unlimited and thus requires innovative skills from the analyst. For example the analyst may decide that some non-linear index formula of weight and height $\{\frac{Weight}{Height}\}$ may produce a better split when classifying a person as over- or underweight. The interpretation of trees using features for combinations should not be so difficult, since the analyst would understand the features before the analysis. For more reading on combinations of variables as splitting rules, see Breiman *et al.* (1984).

4.2 SURROGATE SPLITTERS

Surrogate splits are an innovation due to Breiman *et al.* (1984), which provide alternative univariate splits in the presence of missing values. Surrogate splits are also a way of determining the importance of variables in a classification analysis. Classification trees provide these splits at every non-terminal node for all the variables other than the primary split. These splits are then utilized to classify a case that contains a missing value and to determine the importance of a variable.

A surrogate split is a unique split that most accurately mimics the primary split. We define s^* as the best (primary) split at node t into t_L and t_R. Suppose we have variable x_m. Let S_m be all possible splits on this variable and S'_m be all the splits on any other variable, complementary to S_m. Thus for any split s_m ($s_m \in S_m \cup S'_m$) of node t into t'_L and t'_R, let $N_j(LL)$ be the number of cases in class j in node t that both s^* and s_m send left (into $t_L \cap t'_L$) and $N_j(RR)$ be the number of cases in class j in node t that both s^* and s_m send right (into $t_R \cap t'_R$). Then the probability that a case will fall into $t_L \cap t'_L$ or $t_R \cap t'_R$ is

$$p(t_L \cap t'_L) = \sum_j \pi(j) N_j(LL)/N_j \quad \text{or} \quad p(t_R \cap t'_R) = \sum_j \pi(j) N_j(RR)/N_j,$$

where N_j is the number of cases in class j in node t. Now we define the probability that both s^* and s_m send a case in node t left or right as

$$p_{LL}(s^*, s_m) = p(t_L \cap t'_L)/p(t) \quad \text{or} \quad p_{RR}(s^*, s_m) = p(t_R \cap t'_R)/p(t),$$

with $p(t)$ as defined in Chapter 2. Suppose s^* is unknown and we try to predict s^* using s_m, then the probability that s_m predicts s^* correctly is estimated by

$$p(s^*, s_m) = p_{LL}(s^*, s_m) + p_{RR}(s^*, s_m).$$

Where this probability is a maximum, s_m is the best surrogate split for the primary split in the presence of a missing value. Thus we can define the best surrogate split on x_m as $\hat{s}_m$ where

$$p(s^*, \hat{s}_m) = \max_{s_m} p(s^*, s_m)$$

and $\hat{s}_m \in S_m \cup S'_m$. Surrogate splits are used for three applications: Handling missing values, variable importance and detecting masked variables.

Handling missing values in any analysis is very important since it influences your conclusion. Surrogate splits have been implemented in classification trees for two reasons: to make effective use of all cases in the data set and to classify any case that is dropped through the final classification tree. When a variable has a missing value, then s^* does not exist for that case. Then choose among all non-missing variables in that case, one (say x_m) with $\hat{s}_m$ having the highest measure of predictive association with s^*. The measure of predictive association is calculated as

$$\lambda(s^* \mid \hat{s}_m) = \frac{\min(p_L, p_R) - (1 - (p(s^*, \hat{s}_m))}{\min(p_L, p_R)},$$

where p_L and p_R are the proportions of cases that s^* sends to t_L and t_R. When $\lambda(s^* \mid \hat{s}_m)$ is the highest, then $\hat{s}_m$ is the best surrogate split for s^*. When explanatory variables are highly correlated, then $\lambda(s^* \mid \hat{s}_m)$ will be high and the surrogate split will be the best alternative for the primary split. The worst situation occurs when the highest $\lambda(s^* \mid \hat{s}_m)$ is relatively small, so that the best surrogate split, is still a "not-so-good" alternative for the primary split. New cases that contain missing values in the primary splitting variable are now easily classified using a surrogate split.

Researchers are often interested in ascertaining which variables are the more important ones. ***Variable importance*** is obtained by ranking the variables. This process is similar to stepwise discriminant analysis. For classification trees, variable importance does not only consider the variables that are used in the splitting of a node, but also variables that are masked. The masked one will play a prominent role in the tree, when

another variable is removed. For this reason, surrogate splits play an important role in defining the variable importance through the decrease in impurity of a node. Let T be the best tree selected and let t be a node such that $t \in T$. For the Gini criterion, the decrease in impurity using a surrogate split is defined as $\Delta I(\hat{s}_m, t)$ and for the Twoing criterion it is defined as $\Delta I(\hat{s}_m, t) = \max_{C_1} \Delta I(\hat{s}_m, t, C_1)$. Then the measure of importance of a variable (say x_m) is defined as

$$M(x_m) = \sum_{t \in T} \Delta I(\hat{s}_m, t),$$

which is the decrease in impurity from using the surrogate split, summed over all the nodes in T. By calculating the normalized quantities $100 * \frac{M(x_m)}{\max M(x_m)}$, the most important variable has the score of 100 and the other variables have scores of 0 to 100.

4.3 RANDOM FORESTS

Breiman (2001) wrote on random forests as *"a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest"*. Thus this procedure uses $\mathcal{L}$ and some injected random element, in the construction of multiple trees. In this paper (Breiman, 2001), Breiman studies forests by using randomly selected variables or combinations of variables at each node to grow the trees. In his experiments, he utilizes Bagging for several reasons. Therefore resampling is also used for random forests, but not in the same manner as Bagging and ARCing described in Chapter 3.

The simplest random forest is formed by selecting randomly a small subset of variables to split on, at each node. Grow the tree by the methods explained in Chapter 2, but keep the maximal grown tree (do not prune). Breiman calls this procedure Forest-RI and he uses a single randomly selected variable and a group of randomly selected variables in his experiments. His results show that random forests do improve classification accuracy, which compare favourably to AdaBoost. Using a random element with linear combinations of variables also compares favourably to AdaBoost. The latter procedure Breiman calls Forest-RC.

According to Breiman (2001), forests proof to be an effective classification tool when the right kind of randomness is injected. Forest-RI and Forest-RC are just two kinds of procedures of injected random elements, but other random features have also been suggested. The use of random forests is fairly new in trees, its computation time is faster than other resampling procedures since randomly selected variables are used and not all. Its accuracy is comparable to other resampling techniques. Random forests can do the following:

- Classification
- Variable importance
- Computes proximity measures between cases
- Relatively robust to outliers and noise

More reading and software on random forests for classification and regression trees can be found on Breiman's website: *(http://oz.berkeley.edu/users/breiman/)*

4.4 CONCLUSION

The sections discussed in Chapters 2, 3 and 4 form part of the theory underlying the CART software discussed in Chapter 5. With all these options, there is no definite parameter setting for classification trees and the choice is a very experimental process. The analyst can exercise various settings for the trees to study its performance before settling for the final classifier. The use of resampling techniques has dramatic effects on the classification power and should be explored when the analyst wants to steer the analysis to better classifications. Identifying important variables from surrogate splits is also a fundamental addition to trees. It provides so much more insight into the data and the analyst can study and learn more from the data. It also provides an automatic basis before continuing with other analyses, similar to selecting the important variables in a stepwise[2] discriminant analysis. Combining variables is also effective, but requires understanding the data, which variables to combine and this surely affects the

[2] Klecka (1980) discusses a stepwise selection of variables. SAS/ STAT® User's Guide, Version 8, Volume 3 is also a good reference on how this procedure works.

interpretation which becomes more difficult for practical purposes. Random forests however are not part of the options in the version of CART discussed in this text.

CHAPTER 5

The CART Software (Version 4.0)

5.1 INTRODUCING CART

Classification **A**nd **R**egression **T**rees (CART[1]) is an easy-to-use decision tree tool, used to search for and isolate patterns and relationships within large and complex data sets. The patterns and relationships obtained can then be used by the analyst to make predictions or classifications. The analyst can also explore and understand more of the structures in the data set. With the code originally developed by Breiman, Friedman, Olshen and Stone, CART has become Salford Systems' new generation data mining software with many successful applications in practice. CART supports a variety of operating systems, which include Windows 95/98/ME, Windows NT/2000/XP, UNIX, LINUX and IBM MVS and CMS. Using a Windows based interface, it can easily be used by the technical and non-technical analysts. It is available in different memory sizes, thus can easily accommodate a variety of data set sizes. Figure 5.1 gives an example of CART 's windows based graphical interface.

Analysts in many industries including telecommunications, transportation, banking, financial services, insurance, health care, manufacturing, retail, direct mailing, government, marketing, as well as the academic environment have used CART. Being used for different purposes in each industry, the following user quotes taken from Salford Sytems' website (*http://www.salford-systems.com/*) give us a brief idea as to the success of CART applications.

[1] CART® is a registered trademark of California Statistical Software, Inc. and is exclusively licensed to Salford Systems.
Salford Systems * 8880 Rio San Diego Drive * Suite 1045 * San Diego, CA 92108 * Tel 619.543.8880 * Fax 619.543.8888 * mail to:info@salford-systems.com * http://www.salford-systems.com/

RETAIL

"CART is an important statistical analysis tool that we use to segment our databases and predict risk factors for the Sears Card. The advantage of the decision tree format is that our results are easy to interpret; especially with CART, we are able to see a great deal of detail about each of the nodes, such as the node's misclassification costs, the count of data assigned to that node, and a display of the surrogate values substituted for the node."

Steven Li, Senior Manager, Risk Technology, Sears, Roebuck and Co

TELECOMMUNICATIONS

"When we purchased CART, it was the only comprehensive classification and segmentation software available that could handle the large data sets we use for credit card risk management. In addition, CART provides us with a great deal of flexibility by allowing us, for example, to specify a higher penalty for misclassifying a certain data value."

Feng Xu, Senior Manager, AT&T Universal Card Services

BANKING/FINANCE

"CART offers two distinctive advantages that other database segmentation tools do not. First, it allows the analyst to identify the smallest target segment possible, such as 10 out of tens of thousands, with exceptional precision. In addition, CART allows us to specify a higher penalty for misclassifying a potentially poor prospect than for rejecting a good one; this makes us more confident that, for products with very thin margins, our segmentation models avoid prospects who would likely be non profitable. CART is an invaluable data mining and modeling tool for Fleet Financial Group."

Terence Mak, VP, Lead Analytic Consultant, Fleet Financial Group

ACADEMIC

"As a research scientist in both academic and professional environments, I work with databases too large and complex to process manually. CART, unlike multiple linear programming and other methods that are constrained by functional forms, shows me truer characterizations of interrelationships between the data. CART is also a robust program that can support a diverse set of applications ranging, in my case, from food security analyses to pattern recognition and remote sensing problems."

Eric Weiss, Ph.D., Consultant; Arid Lands Resource Sciences, University of Arizona

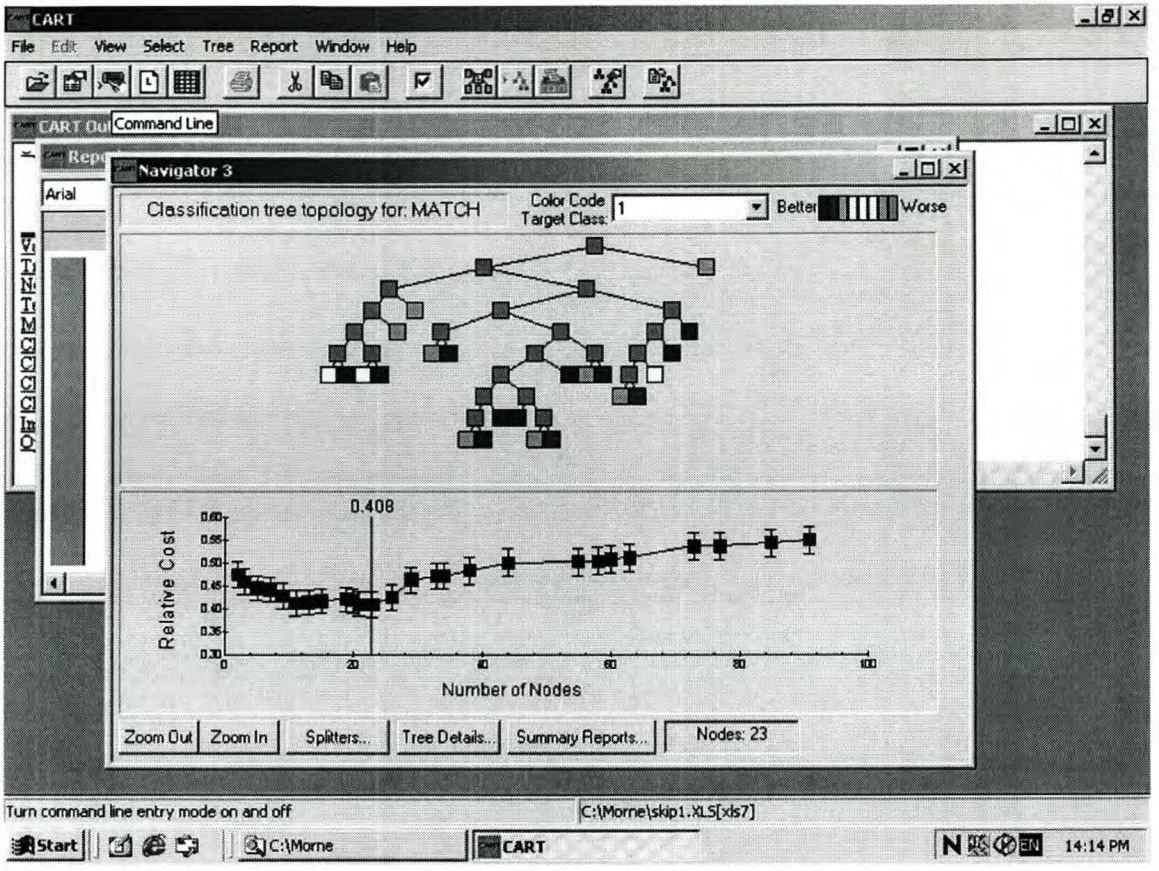

FIGURE 5.1: *CART displaying the optimal binary tree in the navigator window. This is the minimal cost-complexity tree with minimum misclassification rate. The dark blue box-plot-like graphs, represent the T_k sequence of subtrees.*

5.2 ANALYZING DATA IN CART

CART data are in a spreadsheet format and can import data files generated from a number of computer packages, such as ASCII files, Excel, Quatro pro, Lotus, NCSS, SAS, SPSS, S-Plus, etc. Once the data have been imported into CART the analyst can proceed with the classification or regression tree analysis. The first step in the analysis is to specify the response (target) variable and the number of categories/levels (when using classification trees), the measurements (variables) and which are categorical. This is done in the model setup window displayed in Figure 5.2. From a number of possible variables, the analyst can select only those of interest. CART has default settings, but the analyst can specify his own splitting criterion, best tree criterion, testing criterion, priors and misclassification costs. This can be done for a single tree, but if the analyst wants to use a committee of experts in classification, he could use the combining options.

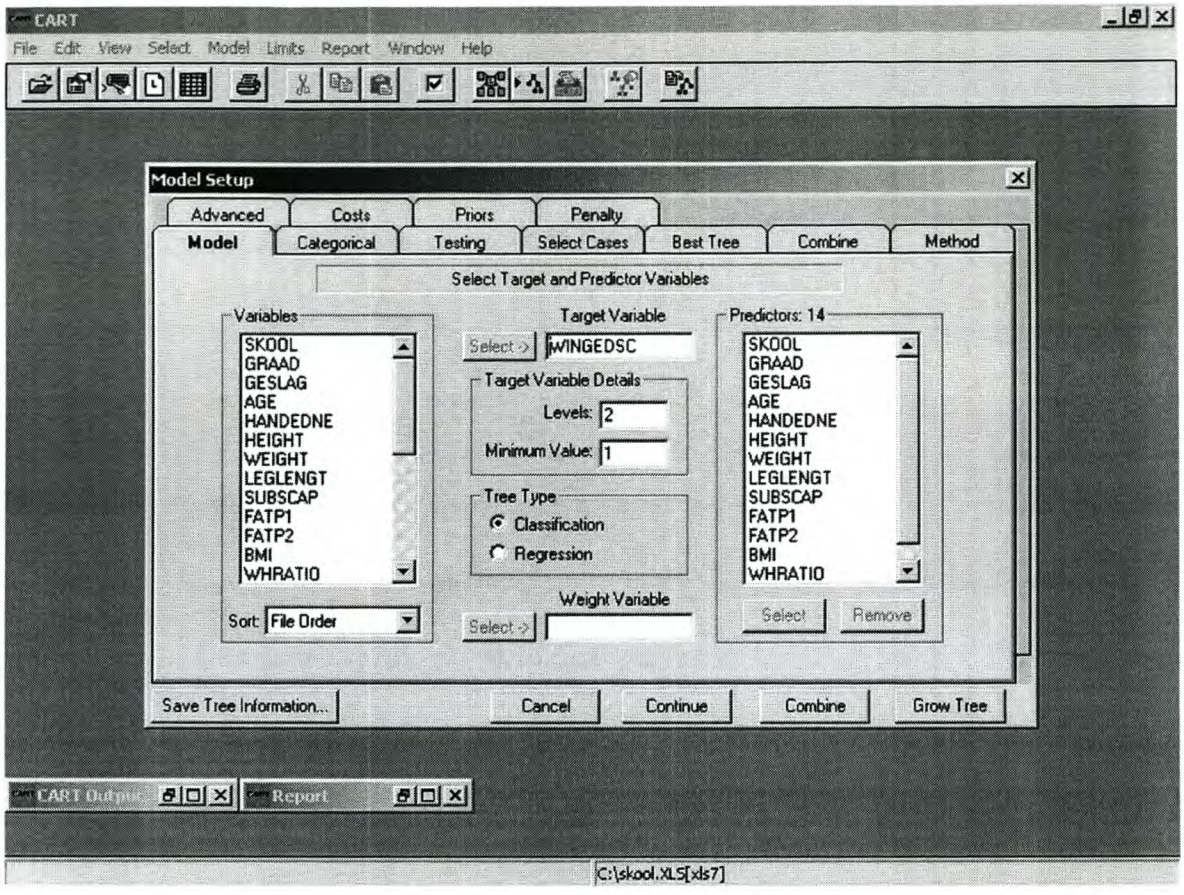

FIGURE 5.2: *Setting up the model for the CART analysis.*

Once all the options have been specified for the CART software, the tree or committee of experts can be grown. CART reports on different results and here the analyst can also exercise his options. The tree navigator displays the best tree and the relative cost or error rate (see Figure 5.1). The analyst can select any one of the sequences of subtrees generated by minimal cost-complexity. By hovering over the tree, you can view the node detail such as the number of cases, the splitting variable and the class representation. You could also study different sections of the tree. CART gives you the option to view surrogate splits, competitor splits and primary splits. Thus, you could see which split is second best to the primary split and how much improvement it could have in the misclassification error rate.

CART generates a report on the classification or regression analysis, which can be printed or copied to another application such as a Word document to write a more detailed report. It generates text output on misclassifications, variable importance,

sequence of trees and standard errors and much more information about the tree. When the analyst is finished with the CART analysis, it is good practice to save the tree information for future use, such as for classifying or predicting new cases. This can be done by saving the tree information as a .tr1 file in CART. Once the tree or committee of experts have been saved, classifying a new data set is easy. By dropping the opened file containing the new data through the tree, you would obtain the classifications of that data set. These classifications can be saved to another application (such as Excel) and the information implemented for whatever purposes it might be useful.

5.3 THE FEATURES OF CART 4.0

CART is a robust tool that provides stable performance and reliable results. It uses binary recursive partitioning to do its splits. It splits each node into two and can do it recursively for the ancestor node and each following descendant node. Each descendant node can also be an ancestor node. The fundamental elements of the CART methodology can be summarized into these categories: (1) *How to split each node of a tree*. (2) *Deciding when a tree is complete or when a node is terminal*. (3) *How to assign a class to each terminal node*.

Given a large data set with many cases and many variables, it becomes a difficult task to split the nodes, since there are so many possibilities. For example, if there are 1000 cases and 15 numerical variables, CART will consider all 15000 possible splits. In choosing the right split by considering all possibilities, CART becomes an automatic variable selection process. CART will therefore only split on the variables which are of significance in obtaining the best classification or regression tree.

The number of splits increases tremendously as the data set gets bigger and the question that now arises, is: which is the best splitting rule given all those possibilities? A goodness-of-split criterion needs to be specified, which is then used as the basis for choosing the best split. CART provides the analyst with quite a number of single variable splitting criteria (Figure 5.3) namely *Gini*, *symmetric Gini*, *twoing*, *ordered twoing* and *class probability* for classification trees. For regression trees, it has *least squares* and *least absolute deviation*. For combined variables, it has the *linear method splitting criterion*. When the analyst has decided upon a splitting criterion, each split is evaluated

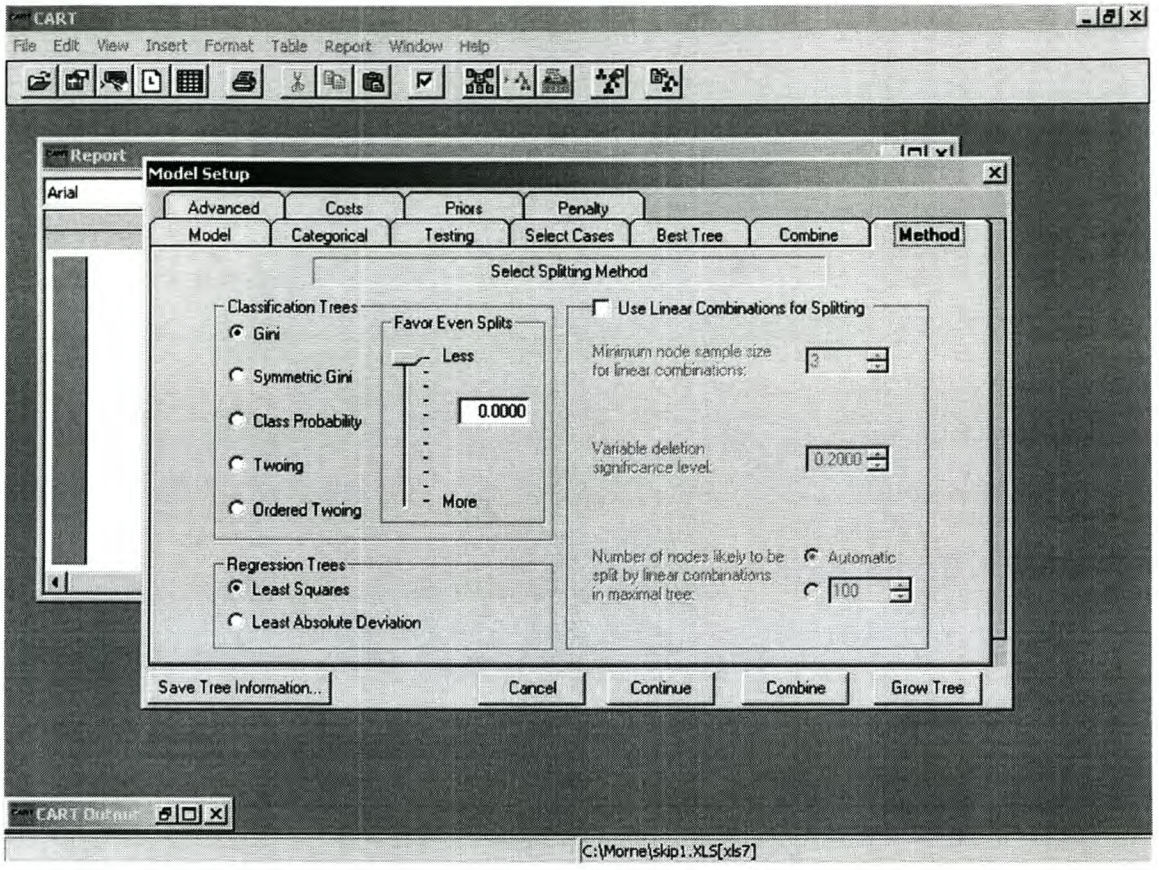

FIGURE 5.3: *The splitting criterion options in CART in the model setup menu. Five criteria for classification trees and two for regression trees. It also shows the options of combining variables for splitting.*

and ordered according to rank. The split that optimizes that criterion is considered the best split and the tree nodes will be partitioned accordingly. The default criterion used by CART is the Gini criterion, which is regarded as the better one when choosing a splitting rule. In summary, the binary partitioning of CART asks the questions "yes" or "no" based on the splitting rule. For example, consider the splitting rule: weight ≤75 kg. For each case, CART will ask the question: Is the weight for a particular case ≤75? If yes, that case will go to the left descendant node. If no, the case will go to the right descendant node.

The splitting process continues until no further splitting is possible, or the splitting process is stopped. Splits can continue until only one case is left in a node. CART makes sure that no important structure or split is overlooked in the process. It is impossible to split further, once a terminal node consists of one case. It is also possible to stop the splitting by specifying the lowest number of cases a terminal node should have.

After a tree is fully-grown, the analyst can now prune the tree to obtain the best optimal tree. The fully-grown tree is referred to as the maximal tree and by pruning away some of its branches, a set of subtrees can be explored.

Pruning is a way of avoiding over-fitting or making sure that a tree can be applied to new cases and not only the cases on which the tree was grown. Pruning in CART can be done using the *minimal cost-complexity* pruning. Selecting the optimal tree involves testing which subtree is the best and this task is an integral part of CART. Once the analyst has obtained all subtrees of interest, they can be compared to each other by calculating their error rates, which is obtained from the proportion of misclassified cases with known classes. With a large enough data set, CART allows you to divide the data into subsets. Perhaps an easy estimate of the error rate in case of large data sets is to divide the data into a *learning sample* and a smaller *test sample*. In CART you can either use a fraction of the data set or import the test sample from a different file. The learning sample is used to grow the maximal tree. Subtrees from this learning sample tree are then derived by pruning.

The test sample is classified by each of the subtrees and the misclassification error rate for each tree can be calculated. When data sets are small, which in practice is often the case, CART allows the analyst to do *V-fold cross-validation*. In this case the entire data set is divided into smaller homogeneous subsets. A tree is grown on most subsets leaving one subset out to calculate an error rate from. This is done giving all subsets a chance to be left out. The error rate from all subsets is then averaged to obtain an estimate of the true error rate. V-fold cross-validation performs better when V is large. Resubstitution can also be used to estimate the error rate, but it is generally too optimistic since the data on which the tree is grown, is also used to calculate it. The tree with the lowest error rate will be the best/optimal tree. Another possibility of choosing the best tree is by using the standard error rule.

Probably the easiest aspect of all is assigning a class to each terminal node. The simplest criterion is called the plurality rule meaning that if

$$p(j_0 \mid t) = \max_j p(j \mid t)$$

holds, class j_0 will be assigned to terminal node t. This means that with J classes, the class with the biggest representation in node t, will be the class (in this case j_0) assigned to

node *t*. This simple criterion can be modified to accommodate the *costs of* making a mistake in *misclassification* and also to adjust (using *priors*) for over- or under-sampling from certain classes. CART gives the analyst the options of doing these modifications in order to obtain the right class assignment.

CART also allows the analyst to use two different resampling techniques namely *Bagging* and *Boosting (ARCing)* in which cases separate CART trees are combined into one classifier or predictor. Bagging: in which each resample is drawn in an identical way. Boosting: in which the way a sample is drawn for the next tree depends on the performance of previous trees. It has been shown that these techniques increase the predictive power of a tree and in cases where subtrees provide unstable results, majority voting or averaging over those trees can only be of benefit to the construction of a final classifier. Figure 5.4 shows the CART window with options for Bagging or Boosting as well as other options. These techniques do not generate a visually unique tree, but it generates a committee of experts with which the analyst can do the classification or prediction. Other options can also be exercised when doing resampling.

Often cases in the data set have missing values. If a variable is selected on which the split is done and a case has a missing value for that variable, that case cannot be classified using that variable. However the use of *surrogate splits* makes this task possible and CART allows you to use surrogate splitters as a substitute for the primary split. *Variable importance* ranks the variables in the tree from the most important to least important. The importance of variables can be viewed considering its contribution to the primary splits, as well as its contribution to the surrogate splits.

Another nice feature of CART is that it can use *weights* to penalize individual cases higher where misclassification of those cases could have disastrous effects. By applying a penalty to certain cases, CART will try to steer the tree away from these cases being misclassified and thus avoid making costly errors. However, CART does not guarantee that the classification will be correct. In practice this type of penalizing can be very helpful.

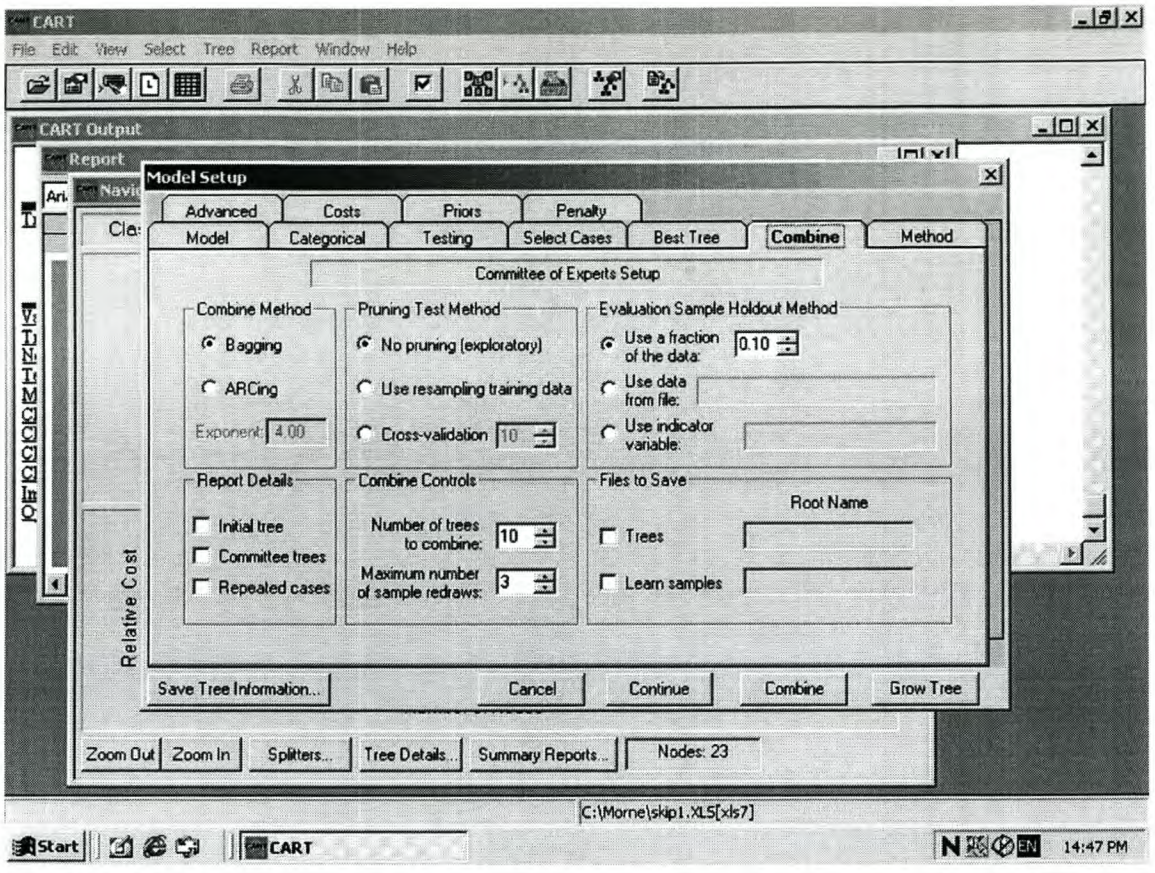

FIGURE 5.4: *Options for Bagging and Boosting (ARCing) with CART default settings. It also displays the fraction of data held out for evaluation (other options as well), the number of bootstrap trees to combine and whether pruning should be done.*

5.4 CONCLUSION

CART is a very sophisticated analysis tool and is based on the theory explained in the previous chapters. It is a flexible tool and gives the analyst a lot of options to explore the data, to try different settings, which eventually will lead to the selection of the best tree or committee of experts. It is very easy to use given that the analyst knows what he wants to do. The visual displays are very nice and it generates output that is understandable and easily interpretable. Some researchers use CART not only as an analysis tool, but also as a preliminary analysis before the main analysis. Having data of high dimensionality that need reduction, CART can be utilized in ranking the variables from most important to least important and choose the best variables from that in another analysis such as regression, discriminant analysis, principal component analysis, etc. Viewing the tree becomes a problem when it is very large. CART allows the analyst to view sections of

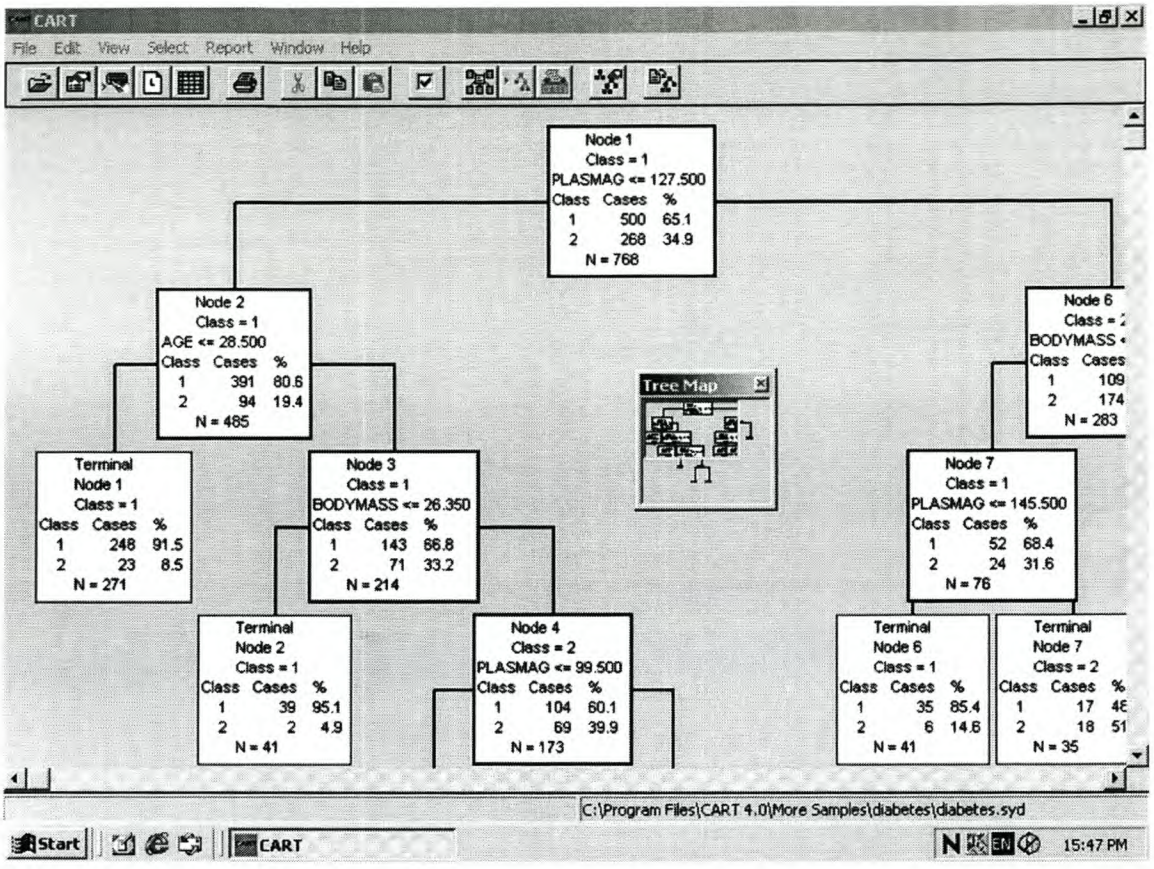

FIGURE 5.5: *The detailed CART tree grown on data in the CART software samples library. The little tree map window guides the analyst through sections of the tree.*

the tree and with a tree map (see Figure 5.5) window it guides you through the tree. When the tree is constructed in CART, the analyst can do different investigations. The sequence of minimal cost-complexity trees can be viewed one at a time by selecting any one in the navigator window as shown in Figure 5.1. CART displays the nodes in different colours with the non-terminal nodes in green and the class representation is displayed by the colour red for better and blue for worse.

Other existing software in which tree modeling is also possible includes ASSISTANT, AID (**A**utomatic **I**nteraction **D**etection), C4.5, QUEST (**Q**uick, **U**nbiased, **E**fficient **S**tatistical **T**rees), CHAID (**Chi**-square **A**utomatic **I**nteraction **D**etector), S-Plus, SAS Enterprise Miner, STATISTICA, etc. For more information about other software packages the interested reader can visit the website *www.recursive-partitioning.com/ Classification_Trees*. These software packages are similar but differ in several ways, such as splitting and pruning criteria, stopping rules, some are menu driven others not,

they differ in output, flexibility, computation time and segmentation (some binary -, others multiway splits).

CHAPTER 6

Other Classification Methods

Let a data set with N observations have a response consisting of J groups ($j=1,\ldots, J$), a vector $\boldsymbol{X}=(X_1,\ldots, X_p)$ of p variables, and let $\boldsymbol{x}=(x_1, \ldots, x_p)$ be the observed measurements on $\boldsymbol{X}$. This is the same data setup as for classification trees. In the sections that follow, we will briefly discuss classifiers based on this setup using a parametric approach, a non-parametric approach as well as a neural network procedure as alternative methods for classification trees. Both parametric and non-parametric approaches make use of distance matrices. The parametric method makes use of the generalized squared distance matrix (obtained from the Mahalanobis distance) and the non-parametric methods make use of the Mahalanobis, Euclidean or Diagonal distance matrix as a metric. A neural network uses an iterative procedure to calculate weights to minimize the performance metric. In all the discussions that follow, we assume that costs of misclassification is 1.

6.1 PARAMETRIC DISCRIMINANT ANALYSIS

The English born statistician, Sir R.A. Fisher first introduced discriminant[1] analysis (in the 1930 's) to separate groups by finding functions based on a number of explanatory variables. The discriminant functions are not only used for separation purposes, but can also be used to classify new observations from the populations on which the functions are based. With parametric discriminant analysis we assume that the explanatory variables $\boldsymbol{X}$ in each group j comes from the multivariate normal distribution

$$f_j(\boldsymbol{x}) = \frac{\exp[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)'\Sigma_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)]}{(2\pi)^{p/2}\,|\Sigma_j|^{1/2}} \qquad -\infty < x_1, \ldots, x_p < \infty; \quad j=1,\ldots, J,$$

where $\boldsymbol{\mu}_j$ represents the mean vector and Σ_j the covariance matrix for population j. These

[1] Terminology due to Sir Ronald Fisher. See Johnson and Wichern (1992) for reference.

parameters are unknown and need to be estimated from the sample data. The sample data mentioned here is the same learning sample ($\mathcal{L}$) discussed in classification trees. In determining the group to which an observation belongs, lets define p_j as the prior probability of group j. Parametric as well as non-parametric discriminant analysis (Section 6.2) try, in different ways, to approximate the well known Bayes rule by using $\mathcal{L}$ to get an estimate of the density function $f_j(\boldsymbol{x})$. With the group memberships known, these methods also fall under supervised learning. Using Bayes rule, we can calculate the posterior probability $P(j|\boldsymbol{X})$ as

$$
\begin{aligned}
P(j|\boldsymbol{X}) &= \text{the probability that an observation comes from group } j \\
&\quad \text{given explanatory variables } \boldsymbol{X} \text{ was observed} \\
&= \frac{(prior) \times (likelihood)}{\sum_{groups} (prior) \times (likelihood)} \\
&= \frac{p_j f_j(\boldsymbol{x})}{\sum_{g=1}^{J} p_g f_g(\boldsymbol{x})}.
\end{aligned}
$$

The likelihood $f_j(\boldsymbol{x})$ for the parametric approach is estimated by the multivariate normal density estimates $\hat{f}_j(\boldsymbol{x})$ for each group j and the prior probability p_j is estimated as $\hat{p}_j$, either from the data or provided by the analyst. All new observations are assigned to the group j for which the posterior probability is a maximum. The posterior probability $P(j|\boldsymbol{X})$ is therefore the likelihood that an observation belongs to a group, j.

When the costs[2] of misclassification equal 1, the analyst can derive certain functions, which can be used to classify an observation. For these functions, $\boldsymbol{\mu}_j$ and Σ_j are estimated from the learning sample as $\bar{\boldsymbol{x}}_j$ and $\boldsymbol{S}_j$, which is the sample mean vector and sample covariance matrix respectively. Discriminant analysis uses the pooled covariance matrix or the with-in group covariance matrices. The with-in group matrices are the individual covariance matrices of the groups in the sample, denoted by $\boldsymbol{S}_j$, $j=1, \ldots, J$. When the with-in group population covariance matrices are equal, the sample with-in group covariance matrices may be pooled (denoted by $\boldsymbol{S}_{pooled}$). Given multivariate normal

[2] Johnson and Wichern (1992), (Chapter 11) gives an example when misclassification costs are not 1.

distributions and applying the natural logarithm to this distribution function, two important discriminant functions can be derived (Johnson and Wichern, 1992; Hastie *et al.*, 2001).

1. The classifier for multivariate normal populations with unit costs and unequal Σ_j's is the *quadratic discriminant function*

$$d_j^Q(\boldsymbol{X}=\boldsymbol{x}) = -\tfrac{1}{2}\ln|\boldsymbol{S}_j| - \tfrac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}}_j)'\boldsymbol{S}_j^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}}_j) + \ln(\hat{p}_j)$$

and

2. the classifier for multivariate normal populations with unit costs and equal Σ_j's is the *linear discriminant function*

$$d_j^L(\boldsymbol{X}=\boldsymbol{x}) = \bar{\boldsymbol{x}}_j'\boldsymbol{S}_{pooled}^{-1}\boldsymbol{x} - \tfrac{1}{2}\bar{\boldsymbol{x}}_j'\boldsymbol{S}_{pooled}^{-1}\bar{\boldsymbol{x}}_j + \ln(\hat{p}_j),$$

where $\boldsymbol{S}_{pooled} = \dfrac{(N_1-1)\boldsymbol{S}_1 + \ldots + (N_J-1)\boldsymbol{S}_J}{N_1+N_2+\ldots+N_J-J}$, N_j = the number of observations in group j, $\hat{p}_j$ is the prior estimate for group j and $|\boldsymbol{S}_j|$ is the determinant of $\boldsymbol{S}_j$. For each of these functions, an observation is assigned to group j if the discriminant score $d_j^*(\boldsymbol{X})$ is the maximum of $d_1^*(\boldsymbol{X})$, $d_2^*(\boldsymbol{X}), \ldots, d_J^*(\boldsymbol{X})$. These functions provide the analyst with more information, over and above the classifications. The signs and magnitudes of the coefficients of the explanatory variables tell a little more about the influence of the explanatory variables. A variable with a large coefficient means that the explanatory variable contributes more to the discriminant score, than one with a small coefficient. A negative coefficient will lead to a smaller discriminant score and vice versa, which in turn will influence the choice of the group membership of an observation.

This type of discriminant analysis has been coded in the powerful and flexible SAS Base, giving the analyst a number of options. For *parametric* discriminant analysis, SAS uses the generalized squared distance matrices. For linear discriminant analysis this distance is

$$D_j^2(\boldsymbol{x}) = (\boldsymbol{x}-\bar{\boldsymbol{x}}_j)'\boldsymbol{S}_{pooled}^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}}_j) - 2\ln(p_j)$$

and for quadratic discriminant analysis it is

$$D_j^2(\boldsymbol{x}) = (\boldsymbol{x}-\bar{\boldsymbol{x}}_j)'\boldsymbol{S}_j^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}}_j) + \ln|\boldsymbol{S}_j| - 2\ln(p_j).$$

The posterior probability $P(j|\boldsymbol{x})$ is calculated from the multivariate normal density as

$$P(j \mid \boldsymbol{x}) = \frac{\exp[-\frac{1}{2}D_j^2(\boldsymbol{x})]}{\sum_{g=1}^{J}\exp[-\frac{1}{2}D_g^2(\boldsymbol{x})]},$$

where J is the number of groups and j refers to a specific group. The prior probabilities are taken into account in the distance matrix. The procedure used is PROC DISCRIM. SAS uses Bartlett's modification of the likelihood ratio test (SAS, 1999) for the homogeneity of the with-in group covariance matrices to test if the Σ_j's are equal. The test statistic has an approximate chi-square distribution and if the test shows that there is enough evidence to accept $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_J$, SAS will use $\boldsymbol{S}_{pooled}$ and perform linear discriminant analysis. Otherwise SAS will use the individual $\boldsymbol{S}_j$'s if POOL=NO and calculate the quadratic discriminant function. The default in SAS is POOL=YES. SAS also calculates the linear or quadratic discriminant functions with which future observations can be classified. With two or three dimensions, parametric discriminant functions can be plotted graphically for further exploration. Although multivariate normality is assumed here, some authors (Manly, 1986; Klecka, 1980) state that some of the discriminant functions from a non-normal population may be worthwhile when used carefully.

The misclassification rate for discriminant analysis can be estimated in the same way as for classification trees. Again, the resubstitution estimate will be a biased, optimistic error rate, but a test sample or a cross-validation estimate can be used to obtain a less biased estimate of the true error rate. Cross-validation in SAS is done using *N-1* observations as the learning sample to derive the discriminant functions. One of the *N* observations is left out each time and classified into a group using the discriminant function obtained from the other *N-1* observations. This is termed the leave-one-out procedure. An independent test sample, to obtain the error rate estimate, can be classified by the discriminant functions when the original data set is big enough to be divided into two sets. SAS uses what is known as the calibration information to classify the observations. Another procedure for classification available in SAS is the canonical discriminant procedure (*cf.* Bennett and Bowers, 1976).

6.2 NON-PARAMETRIC DISCRIMINANT ANALYSIS

Non-parametric methods of discriminant analysis are not based on multivariate normal densities for $\boldsymbol{X}$, but rather on non-parametric estimates of group-specific probability densities. Non-parametric methods can use either the kernel method or k-nearest-neighbour method to obtain the posterior probability estimates in each group to produce a classification criterion. The ***kernel*** method uses a fixed radius (r) and a kernel to estimate the smoothed density at each observation. The Uniform, Normal, Epanechnicov, Biweight or Triweight kernels are all possibilities that can be used to estimate the densities. The posterior probability here is still calculated as

$$P(j \mid \boldsymbol{X}) = \frac{p_j f_j(\boldsymbol{x})}{\sum_{g=1}^{J} p_g f_g(\boldsymbol{x})},$$

but $f_j(\boldsymbol{x})$ is now estimated using any one of the Mahalanobis, Euclidean or Diagonal distance metrics and kernel functions mentioned above. A kernel function should satisfy the condition

$$\int_{-\infty}^{\infty} K_j(\| \boldsymbol{x} \|) d\boldsymbol{x} = 1,$$

where $\| \boldsymbol{x} \|$ defines the distance metric and $K_j(\| \boldsymbol{x} \|)$ the kernel density function. Therefore we can estimate $f_j(\boldsymbol{x})$ by the group-specific density

$$\hat{f}_j(\boldsymbol{x}) = \frac{1}{N_j} \sum K_j(\| \boldsymbol{x} - \boldsymbol{y} \|),$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the vectors of explanatory variables for two different observations ($\boldsymbol{x}$ and $\boldsymbol{y}$). Thus, $\| \boldsymbol{x}\text{-}\boldsymbol{y} \|$ is the distance between them. The summation is over all N_j observations in group j.

Non-parametric discriminant analyses are also coded in PROC DISCRIM of SAS. For non-parametric discriminant analysis using the kernel density approach, the Mahalanobis distance matrix

$$D^2(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})' \boldsymbol{S}_{pooled}^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

and the popular normal kernel[3] (Hastie *et al.*, 2001; Ogden, 1997; SAS, 1999)

$$K(\boldsymbol{x}-\boldsymbol{y}) = \frac{\exp[-\frac{1}{2r^2}D^2(\boldsymbol{x},\boldsymbol{y})]}{(2\pi)^{p/2} r^p \mid \boldsymbol{S}_{pooled} \mid^{1/2}}$$

can be used. The posterior probability using this distance matrix and normal kernel becomes

$$P(j \mid \boldsymbol{x}) = \frac{p_j f(\boldsymbol{x} \mid j)}{\sum_{g=1}^{J} p_g f(\boldsymbol{x} \mid g)},$$

where

$$f(\boldsymbol{x} \mid j) = \frac{1}{N_j} \sum_{i=1}^{N_j} \exp[-\frac{1}{2r^2} D^2(\boldsymbol{x}, \boldsymbol{y}_i)].$$

Notice that the denominator in $K_j(\boldsymbol{x}-\boldsymbol{y})$ are cancelled out as a common factor when calculating $P(j|\boldsymbol{x})$ using $\boldsymbol{S}_{pooled}$. The radius (r) in the above formulas is used by the kernel method to obtain better classifications. It is referred to as the smoothing parameter. The larger the value of r, the smoother the estimated kernel density curve. For smaller values, the density curve may be jagged.

According to Hastie *et al.* (2001) and Breiman *et al.* (1984), the ***k-nearest-neighbour*** method goes back at least to 1951 and is due to Fix and Hodges. The *k*-nearest-neighbour method fixes the number of training set points (k) for each observation. For $k>0$ and the i_n -th observation with explanatory variables vector $\boldsymbol{x}$, find the k nearest observations (neighbours) to i_n in $\mathfrak{L}$ using a distance metric. Then classify observation i_n as group j if more of the k nearest neighbours are in group j than in the other groups. It is a much simpler method than the kernel method.

With the *k*-nearest-neighbours approach the Mahalanobis distance matrix

$$D^2(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})' \boldsymbol{S}^{-1}_{pooled} (\boldsymbol{x}-\boldsymbol{y})$$

can be used in SAS. Of the k distances, the number of distances associated with group j are obtained with posterior probability of membership in each group is calculated as

[3] See SAS/STAT ® User's Guide, Version 8, Volume 2 for other kernel density functions.

$$P(j \mid \boldsymbol{x}) = \frac{p_j m_j(\boldsymbol{x})}{\sum_{g=1}^{J} p_g m_g(\boldsymbol{x})},$$

where $m_g(\boldsymbol{x})$ = proportion of observations in group g in k nearest neighbours of $\boldsymbol{x}$. For non-parametric discriminant analysis, $\boldsymbol{y}$ and $\boldsymbol{x}$ are the observed vectors of explanatory variables for two different observations ($\boldsymbol{y} \neq \boldsymbol{x}$). $D^2(\boldsymbol{x},\boldsymbol{y})$ is the distance between these two vectors. The k that produces the best classification should be chosen.

These are some of the mechanics of non-parametric discriminant analysis underlying the calibration information stored by SAS, which is used for classifying new observations. For each observation, a posterior probability is calculated and when this probability is the highest at a specific group j, that observation receives group j membership. Non-parametric discriminant analyses calculates the same error rates as discussed in the parametric method. However, the non-parametric approach does not give a nice classification function like the parametric approach. They are called *memory-based* classifiers (Hastie *et al.*, 2001) and when a new data set needs to be classified, SAS uses the calibration information to classify these new observations. This is done by providing the TESTDATA= option in the discriminant procedure. For both non-parametric and parametric methods, SAS calculates the posterior probability of an observation given explanatory variable vector $\boldsymbol{x}$. These probabilities are part of the calibration information, which SAS utilizes to classify new cases. SAS allows the analyst to explore the kernel density estimated curves as well.

6.3 NEURAL NETWORK ANALYSIS

Neural networks are well-known analysis tools from the artificial intelligence field. It has been applied to classification problems as an alternative to traditional statistical classification techniques. Neural networks originated from the desire to mathematically model the neural interconnections in the human brain, which has the ability to recognize patterns and uses these patterns and past experiences to make future decisions. This type of characteristic gives neural networks the potential to be used in a classification analysis. Neural network analysis can be traced back to Warren McCulloch and Walter Pitts who started working on it in the late 1930's (Garson, 1998). Just like classification trees, it is a

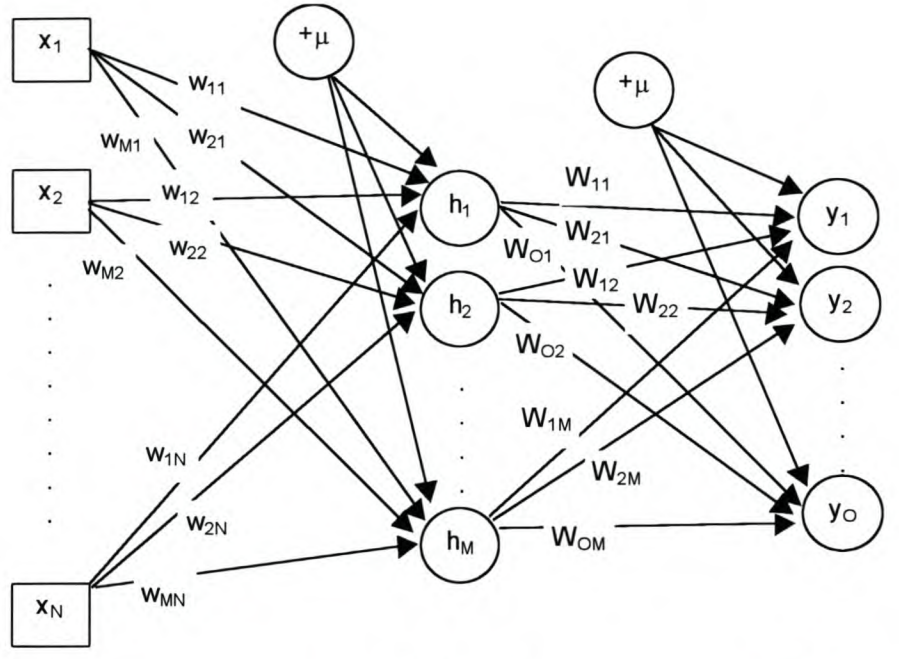

FIGURE 6.1: *A two-layered feedforward neural network. The units indicated by circles represent the two layers. $+\mu$ Represents the biases (intercepts) in each linear combination function.*

non-parametric approach, very robust, can handle categorical and numerical data, and it can model linear and nonlinear relationships in the data. Developments on neural networks have received great attention and its applications have become popular among many statisticians.

This section explains the mechanics of how a neural network works, as published in a paper by Warner and Misra (1996). Many kinds of neural networks exist. One popular type of neural network model uses a multilayered feedforward setup to train the network on a learning sample. By using supervised learning, this network learns the learning sample and produces a classification rule with which new cases (patterns) can be classified. When a new data set with only the explanatory variables (features) is given, the trained network can classify it into one of the known classes. Calculating weights from the input units to hidden units and from hidden units to output units, the neural network recognizes patterns in the data, which ultimately leads to a strong classifier. The algorithm developed to achieve this objective, is called the backpropagation algorithm, which is the most commonly used in neural network modeling (Garson, 1998). Figure 6.1

is a schematic representation of a multilayered (two-layered in this case) feedforward neural network. The backpropagation algorithm derives the weights in the multilayered feedforward neural network. This process utilizes the chain rule of differentiation to calculate the weight adjustments. The weights are iteratively calculated with the objective of minimizing the performance metric (error function). Many error functions are based on the maximum likelihood principle, although in computations the negative log likelihood is minimized (SAS online help). The likelihood is based on a family of error distributions or objective functions, which have optimization features. For illustrative purpose, we will consider the normal distribution, alternatively called the sum of squares of errors (SSE), as our error function. Other distributions are Gamma, Poisson, Bernoulli, Multiple Bernoulli etc., and their error functions differ as well. The choice of the error function has a profound effect on the performance of the network and depends on the nature of the response variable.

Neural network researchers use a number of activation functions (such as the Gaussian, Softmax, Exponential, Identity, Square, etc.), but we will use the *S*-shaped *logistic function* to obtain the weights. The input into a unit is a weighted sum of outputs from previous units in the feedforward network, which can simply be expressed as the *linear combination function*

$$netinput_i = \sum_j w_{ij} \times output_j + \mu_i \ .$$

Here μ_i is a constant added to the weighted sum of outputs also called the bias or intercept of the i -th unit. Other combination functions e.g. Additive, EQSlopes, Xradial, etc. also exist. At each unit in the network the above calculation is performed and the logistic activation function

$$g(netinput_i) = [1 + \exp(-netinput_i)]^{-1}$$

is then applied before sending this as a netinput to the next unit in the network, continuing until the final output unit is reached. Let's examine the case of a two-layered feedforward network. Let w be the weights from the input units to the hidden units and W be the weights from the hidden units to the output units. For simplicity, assume that the bias is zero.

Let

i refer to the number of input units having a total of N,

p refer to the number of cases having a total of n,

k refer to the number of output units having a total of O,

j refer to the number of hidden units having a total of M.

At the beginning of the network the input units (x_{pi}) provide data to the hidden units (h_{ij}) in the form of a netinput, which is equal to

$$h_{ij} = \sum_{i=1}^{N} w_{ji} x_{pi} .$$

Applying the activation function, let

$$v_{pj} = [1 + \exp(-h_{pj})]^{-1}$$

be the output of the hidden units, which is sent as netinput to the output units in the form of f_{pk}, where

$$f_{pk} = \sum_{j=1}^{M} W_{kj} v_{pj} .$$

When the activation function is applied, the final output ($\hat{y}_{pk}$) is

$$\hat{y}_{pk} = [1 + \exp(-f_{pk})]^{-1} .$$

All this information can be substituted in SSE (performance metric) to obtain the equation as shown below. Let E denote SSE and define E as

$$E = SSE = \frac{1}{2} \sum_{p=1}^{n} \sum_{k=1}^{O} \{y_{pk} - \hat{y}_{pk}\}^2$$

with

$$\hat{y}_{pk} = [1 + \exp(-\sum_{j=1}^{M} W_{kj} \{1 + \exp(-\sum_{i=1}^{N} w_{ji} x_{pi})\}^{-1})]^{-1} .$$

Then E can be written as

$$E = \frac{1}{2} \sum_{p=1}^{n} \sum_{k=1}^{O} \{y_{pk} - [1 + \exp(-\sum_{j=1}^{M} W_{kj} \{1 + \exp(-\sum_{i=1}^{N} w_{ji} x_{pi})\}^{-1})]^{-1}\}^2 .$$

The adjustments in the weights ΔW_{kj} from the hidden units to the output units are derived, using the chain rule, as

$$\begin{aligned}\Delta W_{kj} &= -\eta \frac{\partial E}{\partial W_{kj}} \\ &= -\eta \frac{\partial E}{\partial \hat{y}_{pk}} \cdot \frac{\partial \hat{y}_{pk}}{\partial f_{pk}} \cdot \frac{\partial f_{pk}}{\partial W_{kj}} \\ &= \frac{-\eta[(-1)(y_{pk} - \hat{y}_{pk})]\hat{y}_{pk}(1-\hat{y}_{pk})}{1+\exp(-\sum_{i=1}^{N} w_{ji} x_{pi})}.\end{aligned}$$

The adjustments in the weights Δw_{ji} from the input units to the hidden units are derived as

$$\begin{aligned}\Delta w_{ji} &= -\eta \frac{\partial E}{\partial w_{ji}} \\ &= -\eta \sum_{k=1}^{O} \frac{\partial E}{\partial \hat{y}_{pk}} \cdot \frac{\partial \hat{y}_{pk}}{\partial f_{pk}} \cdot \frac{\partial f_{pk}}{\partial v_{pj}} \cdot \frac{\partial v_{pj}}{\partial h_{pj}} \cdot \frac{\partial h_{pj}}{\partial w_{ji}} \\ &= \frac{\eta \sum_{k=1}^{O} (y_{pk} - \hat{y}_{pk})\hat{y}_{pk}(1-\hat{y}_{pk}) W_{kj} x_{pi} \exp(-\sum_{i=1}^{N} w_{ji} x_{pi})}{[1+\exp(-\sum_{i=1}^{N} w_{ji} x_{pi})]^2},\end{aligned}$$

where η is the learning rate (see Warner and Misra, 1996 for further reading). η is also called a control parameter and it controls the size of the iteration steps (the rate at which the network learns the data) when weights are calculated. These weight adjustments are calculated and previous weights updated. The weight adjustments are non-linear functions and cannot be solved mathematically, but numerically by using the iterative backpropagation algorithm, until E is a minimum and no longer changes. Then the process converges and the results are obtained. If η is large the process converges quickly, since fewer iterations are needed and vice versa. The correct setting for η is chosen by experimentation. For a network consisting of more than two layers, the process becomes more complicated, but follows the same recipe. Neural network modeling is a process whereby explanatory variables are provided and outputs are given. Between these

two stages, it is not clear which variables actually played an important role in the separation of the classes, a phenomena which is referred to as a "black-box". Although a neural network is an accurate modeling tool, statements such as these make it less attractive when more insight is needed about the data.

Neural networks[4] are also memory-based classifiers and are one of the data mining tools coded in SAS Enterprise Miner. The Sub-sections 7.6.1 and 7.6.2 in Chapter 7, give an illustration of the setup of a neural network using this software. Neural networks have been developed quite extensively, and although Warner and Misra (1996) explain it in a regression framework, we can extend it to a classification framework. This section should be viewed merely as a basic illustration of the mechanics of this technique.

[4] Vast literature exists on the subject of neural networks. References provided in the SAS online help facility are:

a) Bishop, C.M., *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press, 1995.
b) Tao, K.M., "A closer look at the radial basis function (RBF) networks," *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers* (Singh, A., ed.), **1** (1993), 401-405, Los Alamitos, CA: IEEE Comput. Soc. Press.
c) Werntges, H.W., "Partitions of unity improve neural function approximation," *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, **2** (1993), 914-918.

CHAPTER 7

Applying and Comparing Classification Methods

7.1 INTRODUCTION

The previous chapters described theoretical and philosophical ideas innovated by renowned statisticians and mathematicians. This chapter brings their ideas into practice. Evaluating the performance and robustness of newly developed classification methods are often done by using simulated data from theoretical probability distributions. In the uncertain practical world, where these methods will eventually be applied, data are not always in a structure that the analyst would expect. Given such a condition, the analyst has the option of using only the numerical subset of data applicable to a statistical method and work with the method to analyze those data. An alternative is to use a more robust technique, which can accommodate any data type. Using a subset of a data set does not always give as good results as when using the whole data set, or the whole data set may give worse results than the subset. Therefore, any analyst should be careful when deciding upon a technique for his analysis, since the results obtained from the different techniques may differ tremendously.

As seen thus far in this text, classical statistical methods such as parametric and non-parametric discriminant analysis are based on certain assumptions and in practice, these assumptions are not always met. The robust classification tree and neural network methods make virtually no assumptions at all. In this Chapter, we explore the application and comparison of these methods. The data sets used come from textbooks, are used as sample data in software packages, obtained from websites and some were obtained from the Centre for Statistical Consultation at Stellenbosch University. The learning algorithms of CART, SAS Base and SAS Enterprise Miner were used to analyze the data sets. In Section 7.2, a short overview of the software is given. For simplicity, all

misclassification costs in the analyses were assumed to be 1 and prior probabilities were assumed to be equal.

7.2 A SHORT OVERVIEW OF SOFTWARE

As seen in Chapter 5, CART version 4.0 was developed as a Data Mining tool. This tool was designed to handle large amounts of data. Its use is limited only to Classification and Regression Trees. Performing other statistical analyses is not possible. It is completely menu-driven and requires no programming. However, CART has a built-in programming language.

SAS Base version 8.2 is a flexible statistical analysis tool and can perform a large number of statistical analyses. Most of the modern statistical methods have been coded in this software. It requires some programming to perform an analysis. It is a powerful tool and popular among many statisticians. Its flexibility allows the analyst to manipulate data in a manner that is impossible with CART or SAS Enterprise Miner.

SAS Enterprise Miner version 4.1 is also a very powerful Data Mining tool. Neural networks, Trees and Logistic regression are some of the methods coded in this software. It has also been designed to handle large amounts of data. It is a menu-driven software package and was found to be very flexible for the neural network analyses in this study.

From this overview is it clear that these software packages have been designed for special purposes. Each one has some advantages and disadvantages over the other. For specific analyses, some are more powerful. For example, CART is more powerful in analyzing data using trees. SAS Base contains discriminant analysis procedures and others not. SAS Enterprise Miner has neural networks and CART does not. There are many differences among these software packages. In the analyses, we chose the most flexible software that suits the nature of the data the best.

7.3 APPLICATIONS OF CLASSIFICATION METHODS

Some authors (Garson, 1998) have stated that traditional statistical techniques may outperform robust techniques such as classification trees and neural networks when the

data are clean and assumptions are not violated. However, when data are messy, complex and mixed types, the robust techniques become superior. The areas of application of classification methods are very broad. It can be applied in agriculture, medicine and health, business and economics, biology, psychology, environmental studies, astronomy, chemistry, government and many other areas. The following are some examples of the problems that can be solved using classification methods (found at *http://www.stats.ox.ac.uk/~northrop/teaching/PAN/*):

- Diagnosing diseases
- Recognizing dangerous driving conditions
- Identifying types of car, airplane, etc.
- Identifying suspected criminals by fingerprints and DNA profiles
- Classifying galaxies by shape
- Identifying incoming missiles from radar or sonar signals
- Detecting shoals of fish by sonar
- Deciding which customer will be good credit risks
- Spotting good opportunities on the financial markets

In a study by Latorre *et al.* (1994), both *k*-nearest-neighbour and linear discriminant analyses were applied to differentiate among wines from different brands, which can be used as possible substrates for falsification due to similar organoleptic properties and colour. The application was successful for the aim of their study. Kowalski and Kwan (1978) applied *k*-nearest-neighbour analysis to classify wine samples according to vintage year and wine region. Kowalski and Bender (1972) also applied *k*-nearest-neighbour analysis to chemical data. Debeljak *et al.* (2001) used classification trees in their habitat modeling of red deer in South-central Slovenia. Meléndez *et al.* (2001) used classification trees to study the discriminating capacity of variables. Gorman and Sejnowski (1988) used neural networks as a classification method in identifying underwater sonar contacts.

7.4 A CART EXAMPLE: THE SHIP DATA SET

We will now discuss a practical example and investigate the output of a classification tree analysis using the CART software. We will start by giving a description of the data set

and then proceed to the results from CART. In the analysis that follows, a selected mixed set of variables from the original data set was used. The Ship data set was used with permission from Dr. M. Kidd.

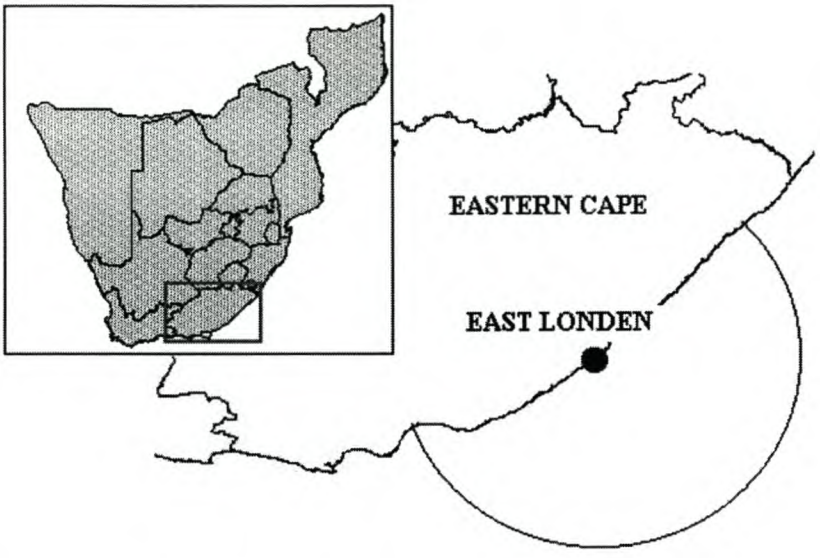

FIGURE 7.1: *East Londen (South Africa) in the Eastern Cape with radius of the radar station.*

Description of the data: In a study, ship information was used to match data from a radar station at the coast of East Londen (COASTRAD) and data from a Shipping Information System (SIS) database. These two sources gather ship data in two different ways and in the study the two databases were integrated by matching COASTRAD data with entries in the SIS database. However, for some SIS data points no corresponding matches could be found in the COASTRAD database. A match response variable was constructed and was binary coded as: 0=non-match and 1=match. Figure 7.1 shows the coast of East Londen and the half circle is the radar area covered by the radar station. When a ship enters the radar zone, position/ time information is recorded from the time the ship is detected by the radar until it disappears from the radar screen. This position/ time information will be referred to as the track of the ship.

Reporting/ sighting agencies (such as airplane and ship agents) identified all ships they see along the coast and was responsible for the ship data in the SIS database. They reported on the different characteristics of the ships and their positions at the time that

they were sited. When the two databases were integrated a search was done for each SIS record (whose position was within the detection range of the radar) to find a matching track from COASTRAD. Depending on the outcome of the search the SIS entry was marked as matched or non-matched. The data set consisted of 1005 cases and the distribution of the binary response was as follows

Class Distribution:

Class 0 (Non-match): 343 cases (34.1%)

Class 1 (Match) : 662 cases (65.9%)

There were 19 explanatory variables (5 Categorical and 14 Numerical).

Aim of study: The purpose is to establish a model to classify cases as matches or non-matches based on the information gathered from the two databases. Insight as to why there are non-matches is also of prime importance.

Results from CART analysis: The ship data were analyzed using classification trees, because of the power of this method in terms of flexibility, classification and data exploration. All 19 variables were used in the analysis. Due to the nature of the data, only univariate splits were used. The Gini and the Twoing splitting criteria yielded the same classification results. Since there was no natural ordering in the classes, the settings were left as the CART default. The rest of this section is the results from the analysis. Table 7.1 is the CART rankings of the six important class separation variables from most important to least important, in terms of the magnitude of their scores. All other variables had zero scores and did not play a significant role in the separation of the two classes. The importance of variables in Table 7.1 is calculated in terms of both primary and surrogate splitters. In terms of the primary splits, *Distance from radar* is the only important variable. Figure 7.2 is the minimal cost-complexity tree with the lowest misclassification rate. This tree was created by CART using 10-fold cross-validation to prune the tree. This is the smallest single tree and gives the best classification. However, the sequence of subtrees in CART can be explored to see what effect the other variables had in splitting the classes. The other subtrees have more structure, more terminal nodes but bigger misclassification rates. The biggest tree was quite complicated and had 74 terminal nodes. Studying Figure 7.2, it becomes clear that when *Distance from radar*

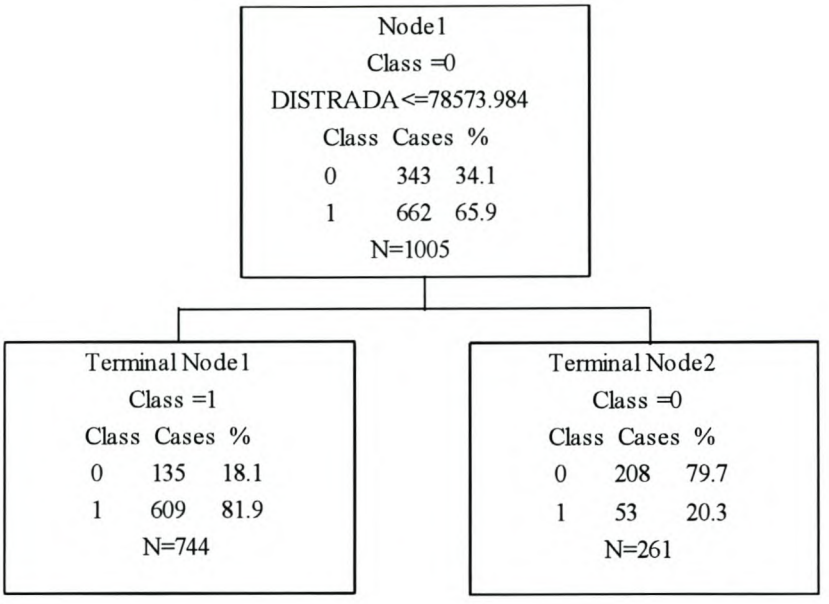

FIGURE 7.2: *The optimal/ best tree obtained from applying CART to the ship data. The left descendant node has majority of class 1. The right descendant node has majority of class 0. Each node contains the class, number of cases and percentage of each class. The ancestor node contains the splitting rule. The value 78573.984 is known as a threshold.*

TABLE 7.1: *Variable importance rankings of ship data in CART.*

Variable	***Score***	
Distance from radar	100.00	
Latitude	30.72	
Longitude	30.53	
Direction from radar	16.65	
Distance from coast	13.78	
Draft	2.07	

TABLE 7.2: *Confusion matrix of ship data obtained from 10-fold cross-validation.*

Actual Class	***Number of cases***	***Predicted Class***	
		0	***1***
0	343	208	135
1	662	53	609

(DISTRADA) <= 78573.984, a ship is more likely to be a match in the two databases. For *Distance from radar* > 78573.984, it will more likely be a non-match. With other classification methods, this type of information will be hidden in the analysis and unknown to the analyst. Table 7.2 is a confusion matrix created from the 10-fold cross-validation results. The overall error rate is (53+135)/1005=0.1871. Thus 19% of all the cases in the data was misclassified and 81% was correctly classified. Classifying a new data set from the same populations would have this expected error rate if the tree in Figure 7.2 were chosen as the classifier. Should the classification results not be satisfactory, the next step is to use Bagging or ARCing. From Table 7.2, it is clear that the tree classifies the matches most accurately. Improvement on the non-matches might still be desirable. Bagging or ARCing can achieve that, if possible. Although classification in these cases might be better, interpretability of the single tree analysis will be sacrificed.

7.5 CLASSIFICATION TREES VERSUS CLASSICAL STATISTICS

For single trees the Gini criterion and 10-fold cross-validation were used to choose the optimal tree. For the committee of experts, no pruning was used. The choice for the number of trees, varied from 50 to 250 trees for both Bagging and ARCing. The exponent in ARCing was kept at 4. In summary, the settings in CART were kept at their default settings. Single trees or committees were saved as .tr1 files in CART, and the test samples were classified by dropping them down the trees. This process was followed for the classification tree results in Tables 7.3b and 7.4b.

When classification trees and classical statistical methods were compared, the best circumstances were created to meet the assumptions for the latter methods. For this reason, the data sets used in this section contain only numerical explanatory. (Categorical variables have been left out in some data sets, see Table 7.3a). The classical statistical methods that were used in this section are the parametric and non-parametric methods discussed in Chapter 6. The analysis using these methods was done using the discriminant analysis procedure of SAS base. Appendix A contains an example of the SAS code for performing the discriminant analyses. CART was used for classification tree analyses.

7.5.1 Choosing the parameter settings for Discriminant analyses

Choosing the settings for the analysis in SAS Base was an experimental process. The classifiers with the lowest overall error rate and best classification for all classes were selected. It should be noted that the data for discriminant analysis were not subjected to a stepwise discriminant analysis, thus classification results are based on all variables. For parametric discriminant analyses, the generalized squared matrices mentioned in Chapter 6 were used. We experimented with linear (LDA) and quadratic discriminant analyses (QDA). With the kernel density approach, the Mahalanobis distance matrix was used. The normal kernel was used and the radius (r) was chosen by trying various settings. For k-nearest-neighbours the Mahalanobis distance matrix was also used. Different values for the training set point (k) were used. For these classical statistical methods, the test samples were classified by specifying the TESTDATA option in PROC DISCRIM. It is important to note that parametric, kernel and k-nearest-neighbour methods were designed to work only on numerical explanatory variables. Applying these methods to categorical explanatory variables may have optimistic results, but may be bad for future classifications.

7.5.2 Classification results of Trees and Discriminant analyses

As an introduction, consider Figure 7.3. It is a plot of the hemophilia data taken from Johnson and Wichern (1992) (see Appendix B for more detail). It is a straightforward scatter plot representing two groups (Noncarriers and obligatory carriers of hemophilia A). A set of new cases was provided to be classified. The circles represent the new cases. Through visual inspection, any analyst would classify all the new cases as noncarriers. However, this simple example was used in parametric, non- parametric and classification tree analysis to ascertain group membership. All these techniques confirmed what is concluded from the visual inspection, and classified all the new cases correctly as noncarriers (error rate=0). The same methods were applied to complex data sets given in Table 7.3a. The misclassification results are presented in Table 7.3b.

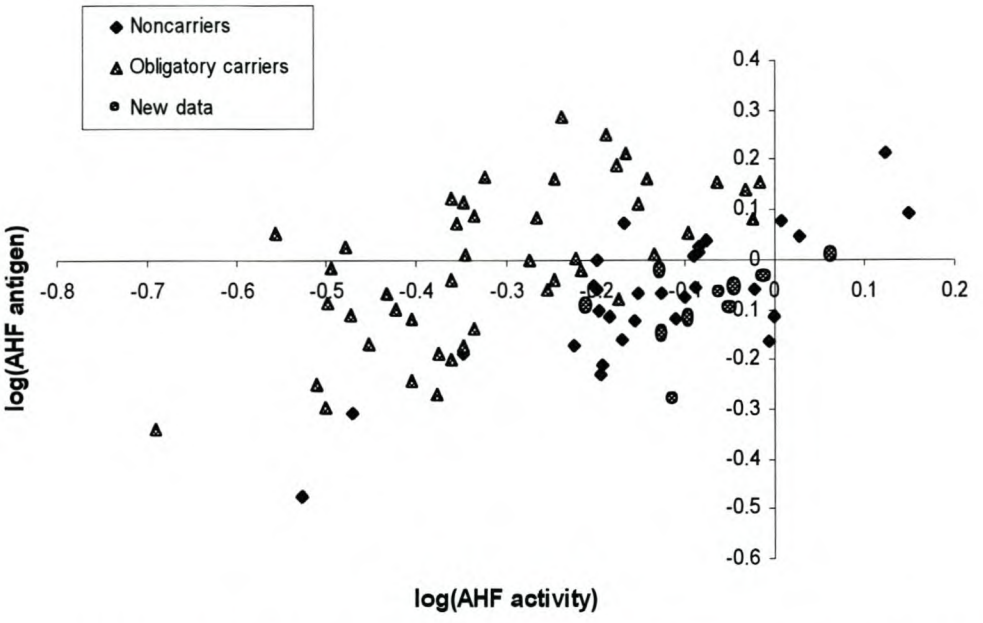

FIGURE 7.3: *Scatter plot of the Hemophilia data. The triangles represent the Obligatory carriers, the diamonds represent the noncarriers and the circles represent the new cases.*

TABLE 7.3a: *The data sets containing only numerical variables.*

Data set	*# of cases in $\mathcal{L}_1$*	*# of cases in $\mathcal{L}_2$*	*Numerical variables*	*Categorical variables*	*# of classes*
School	216	72	10	-	2
Satellite image	4435	2000	36	-	6
Glass	161	53	9	-	6
German credit	750	250	7	-	2
SA heart	347	115	8	-	2

TABLE 7.3b: *The error rates of the test samples, using trees and discriminant analyses.*

Data set	*Classification Trees*			*Discriminant analyses*		
	Single Tree	*Bagging*	*ARCing*	*LDA or QDA*	*Kernel Density*	*k-nearest-neighbour*
School	0.3611	0.375	0.3611	0.4861	0.3889	0.3333
Satellite image	0.157	0.102	0.0925	0.1605	0.181	0.209
Glass	0.4339	0.3584	0.283	0.3774	0.283	0.3019
German credit	0.46	0.356	0.312	0.316	0.432	0.396
SA heart	0.3913	0.3565	0.3478	0.3391	0.3391	0.3739

Preparing the data: For each data set in Table 7.3a a learning sample and a test sample were kept separately to ensure that the error rates of the techniques are compared using the same data. The error rate is calculated as the number of misclassifications divided by the total number of cases in the test sample. The test sample was randomly chosen from each data set ($\mathcal{L}$) using a random number generator in Microsoft® Excel. Random numbers equal to the number of cases in $\mathcal{L}$ were generated. The first 25% of the randomized $\mathcal{L}$ were chosen as the test sample ($\mathcal{L}_2$) and the rest (75%) were used as the learning sample ($\mathcal{L}_1$) to build the classifier. The satellite image data had a separate test sample. See Appendix B for the number of cases in classes. The results of Table 7.3b will be discussed in Section 7.7 together with the results of classification trees versus a neural network that will follow in the next section.

7.6 CLASSIFICATION TREES VERSUS NEURAL NETWORKS

These methods are very robust and assume no functional form of the relationship between the response and explanatory variables. They have the ability to deal with categorical data, where classical statistical techniques are less adequate. The selection of the right settings for these methods was done by trying different values. It becomes a time consuming process, thus some background knowledge of the techniques is vital. For the comparison of classification tree and neural network results, CART and SAS Enterprise Miner were used.

7.6.1 Choosing the parameter settings for a Neural Network

In Section 6.3, we gave a simple illustration of how a neural network process works with certain settings. Since setting up the neural network is very flexible, the documentation in SAS ' online help was used as guidance to set up the neural network framework for each data set. SAS Enterprise Miner has different Tabs and the neural network settings are done within these Tabs. For example: The neural network architecture (means Model in statistical terms) and the number of hidden neurons are selected in the Basic Tab; The model selection criteria is selected in the General Tab; Setting the variable status, variable type and variable role is done in the Variables Tab. Figure 7.4 and Figure 7.5

show an example of a SAS Enterprise Miner setup with these different Tabs. In designing a neural network, one can exercise three options:

- accept the default network architecture settings of SAS Enterprise Miner,
- use the basic templates in the Basic Tab to configure the network or
- specify the appropriate network architecture in the Advanced Tab, where one can define or modify input, hidden, and output layers. One can also add or remove connections among these layers.

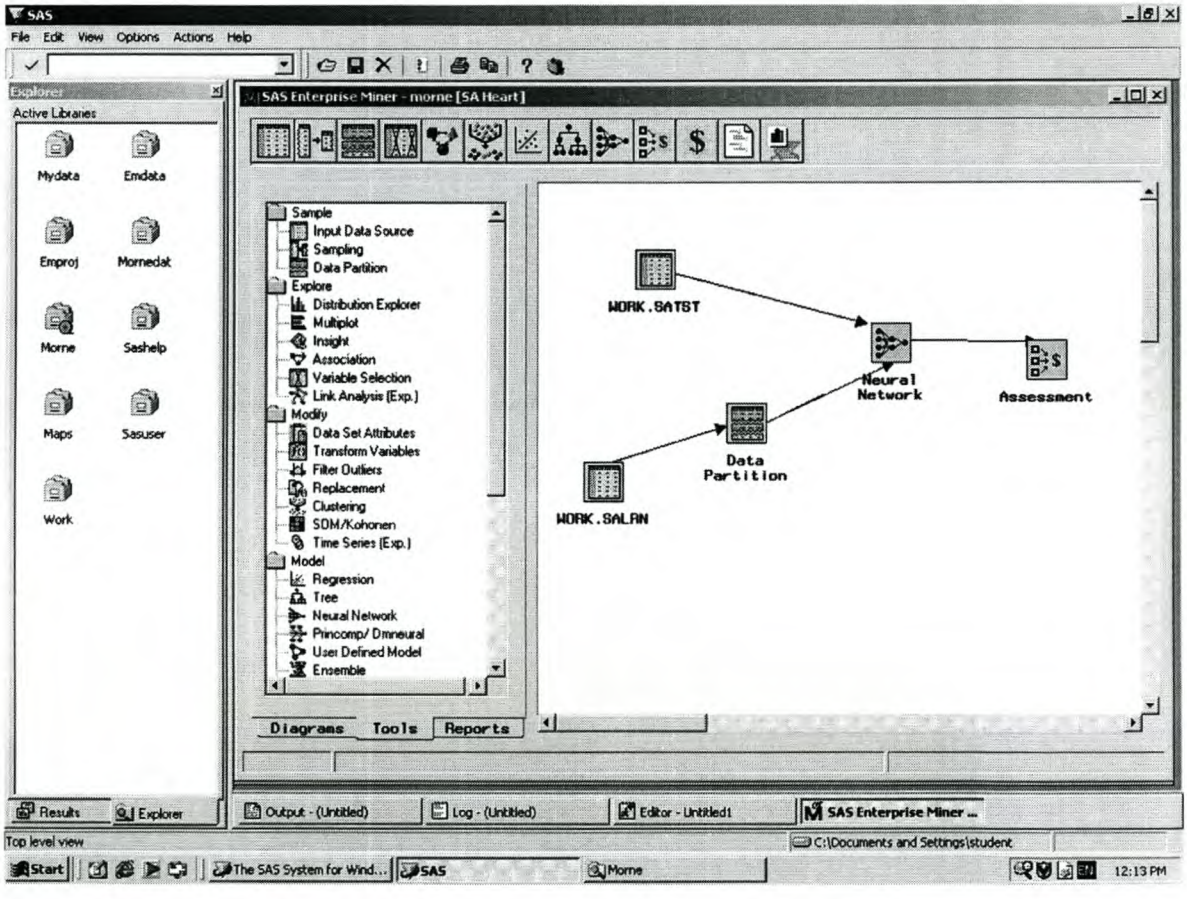

FIGURE 7.4: *This picture illustrates a neural network setup using the graphical interface of SAS Enterprise Miner. Two Input Data Source nodes are used to access the learning sample and the test sample. The Data Partitioning Node can be used to split the learning sample into a training set and a validation set. These nodes are connected to the Neural Network node. The Neural Network node is used for neural network modeling. When this node is opened, the Tabs as shown in Figure 7.5 can be viewed. The Assessment node provides a common framework for comparing models and predictions from the neural network modeling node.*

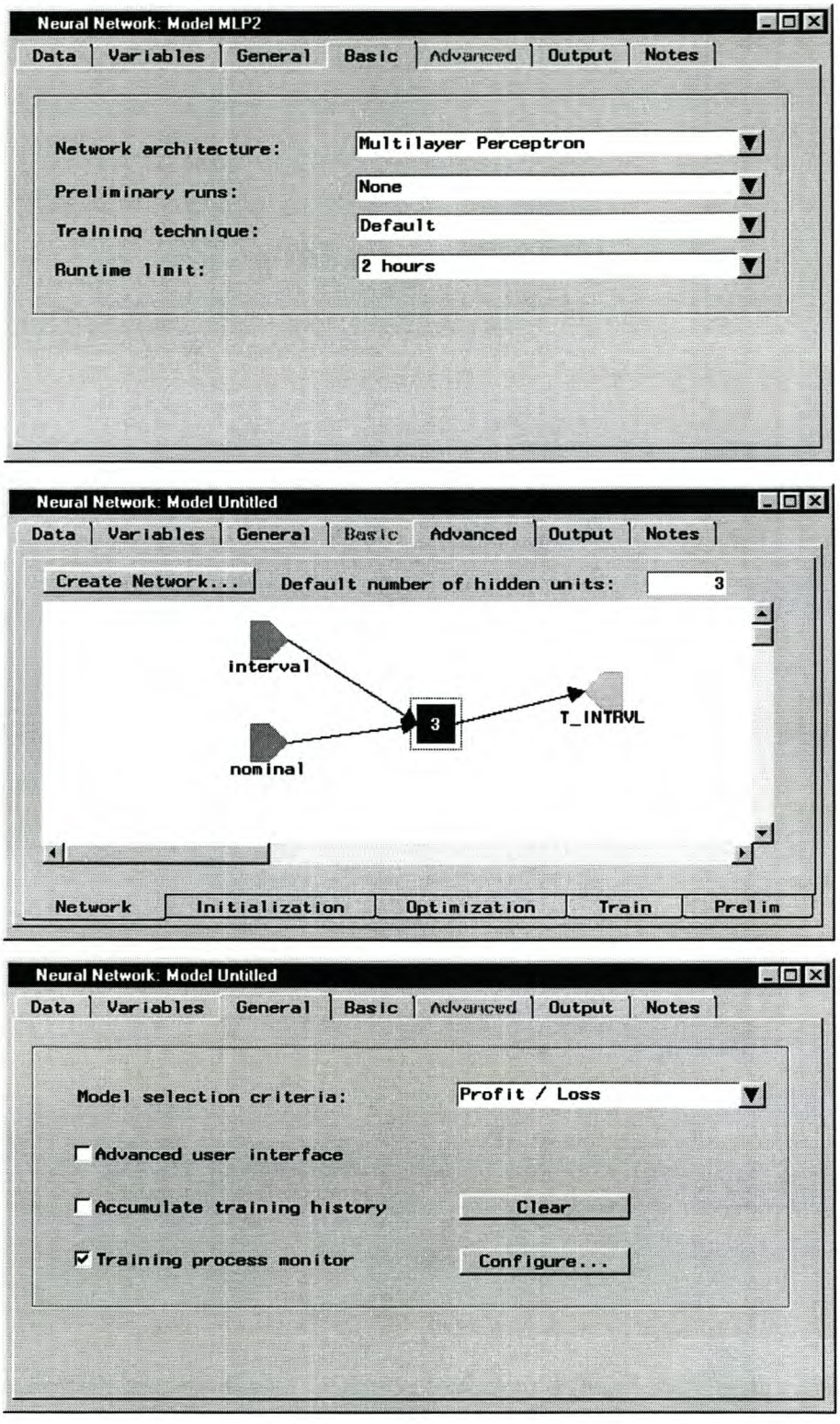

FIGURE 7.5: *The Basic Tab, the Advanced Tab and the General Tab found within the neural network node displayed in Figure 7.4. The settings of the neural network architecture are done in these tabs.*

We experimented with the settings within these options, in order to obtain a network that classifies each test sample the best. Therefore, the settings for each data set differ. The rest of this section is a discussion of the neural network settings used to analyze the data in Table 7.4a.

The form of neural network used, is the popular *multilayer perceptron* (MLP), which is the default architecture in SAS Enterprise Miner (SAS online help). This is a built-in architecture of SAS Enterprise Miner. It uses a linear combination function, the *tanh* (hyperbolic tangent) activation function and one can set the number of hidden layers. The *standard backpropagation* training method was used, with different settings of the learning rate. Learning rates of 0.1 to 0.5 were used. The model selection criterion was the misclassification rate.

In some of the analyses, the neural network architecture was created in the Advanced Tab. We experimented with the *softmax* and *logistic* activation function, the linear combination function, and the *Normal* (*Gaussian*), *Bernoulli* and *Multiple Bernoulli* error functions. SAS online help gives the likelihood of these distributions. SAS online help also gives a list of all possible activation, combination and error functions. Hastie *et al.* (2001) discuss some issues in training the neural network, and suggest that the starting value for the weights be randomly chosen near zero. They also suggest that the range for the number of hidden units be 5 to 100. However, they give no suggestions for the number of hidden layers, leaving that up to experimentation. In our analyses, the number of hidden layers was kept at one. Hidden units equal to 20, 30, 40 or 50 were selected at random. The normal distribution with a mean of zero and a standard deviation of one was used for the initial selection of the bias term. For each data set, the entire learning sample was used as the training set, thus no validation set was used. The test samples were classified by using a separate Input Data Source node, which is connected to the neural network trained on the learning sample.

7.6.2 Classification results of Trees and Neural Networks

Table 7.4a is a summary of the data sets that were subjected to a classification tree and neural network analysis. Table 7.4b contains the error rates, based on the test sample of each data set. The data were prepared as mentioned in Section 7.5.2, using random

numbers generated in Excel. The learning samples and test samples consist of exactly the same observations as in Tables 7.3a and 7.3b. However, the difference here is that these data sets contain both numerical and categorical explanatory variables. The ship data were also divided into a learning sample and a test sample. It was subjected to the classification methods and the results are displayed in Table 7.4b. The error rates of the Satellite image and Glass data obtained from the neural network analysis are also given. The error rate is calculated as the number of misclassifications divided by the number of cases in the test sample. In Section 7.7 we will discuss and compare the error rates in Table 7.4b.

TABLE 7.4a: *The data sets containing a mixture of variables.*

Data set	*# of cases in* $\mathcal{L}_1$	*# of cases in* $\mathcal{L}_2$	*Numerical variables*	*Categorical variables*	*# of classes*
Ship	754	251	14	5	2
School	216	72	10	4	2
Satellite image	4435	2000	36	-	6
Glass	161	53	9	-	6
German credit	750	250	7	13	2
SA heart	347	115	8	1	2

TABLE 7.4b: *The error rates for the test samples using trees and a neural network.*

	Classification Trees			*Neural network*
Data set	*Single Tree*	*Bagging*	*ARCing*	*Multilayer Perceptron*
Ship	0.1713	0.2111	0.1992	0.1952
School	0.3611	0.3194	0.375	0.3333
Satellite image	0.157	0.102	0.0925	0.274
Glass	0.4339	0.3584	0.283	0.3585
German credit	0.372	0.236	0.252	0.276
SA heart	0.2956	0.3217	0.3478	0.2783

7.7 DISCUSSION OF RESULTS

It should be noted that the error rates in Tables 7.3b and 7.4b can still be reduced if one should experiment more with the settings of the different classification methods. Since this is a time consuming process, only a few settings were experimented with, and the lowest error rate reported in the tables.

The error rates in Table 7.3b show that the classification accuracy of the classification methods differs very much, but are still comparable. Comparing the single tree error rates to that of the discriminant analyses, we see that some discriminant methods do very well for certain data sets. When Bagging and ARCing were applied, the classification accuracy increased for almost every data set, compared to the single tree results. This shows that these resampling methods are very powerful at forcing down the error rates. For some data sets a large number of trees in the committees were used. For each data set, more or less the same number of trees were used for ARCing and Bagging. ARCing seems to reduce the error rate faster than Bagging. Overall, the results for Bagging and ARCing look much more promising than the results of the single trees and the discriminant analyses. However, a stepwise discriminant procedure may still reduce the error rates of discriminant analyses. One advantage of trees is that splits are only on the most important variables and the data need not be subjected to a variable selection procedure such as a stepwise discriminant analysis. Although Bagging and ARCing reduce the error rate quite a lot, their disadvantage is the loss of the hierarchical structure of the classifier. Thus, the gain in classification accuracy comes at the cost of sacrificing insight into the data structure. Bagging and ARCing could also become very time consuming with large data sets and a large number of trees in the committee of experts.

Modeling data with a neural network can also be a time consuming process. Despite the fact that it requires more settings than trees, its iterative nature also takes some time to train the network. Most of the MLP neural network error rates are fairly low when compared to the single tree error rates in Table 7.4b. However, the single tree results are still comparable to neural network results, and it is much more informative. The hierarchical single tree provides more information about the data than the "black-box" that the neural network exhibits. As stated before, Bagging and ARCing sacrifices the tree interpretability and therefore provide more or less the same amount of information as the neural network.

In summary of the above experiments, no single classifier stands out as best every time. Interestingly enough, classification trees performed better when both categorical and numerical explanatory variables were included in the analyses. This provides an indication that when dealing with a mixture of data types, a robust method should be the

preferred choice. The advantage of each method lies in its simplicity, the interpretability of the output, as well as the insight it provides into the data structure. Parametric discriminant analysis and classification trees are here the superior methods. For future classification, a linear or quadratic discriminant function and a hierarchical classification tree can be used. With non-parametric discriminant analyses and neural networks, very little information is provided in this regard. Both methods need to refer back to the learning sample data when doing future classifications. This process is impossible without the aid of computer software. The availability of these methods in software and their computation time are more factors that will eventually influence the choice of a method.

Using Bagging and ARCing to increase classification accuracy is definitely worthwhile. These results are also shown in the papers of Breiman (1996 and 1998) and Opitz and Maclin (1999), and was seen in Chapter 3. With the use of resampling methods, only one learning algorithm (trees in our case) was used. Using them with other learning algorithms (discriminant analyses and neural network) may also give lower error rates than those reported in Tables 7.3b and 7.4b.

7.8 DISADVANTAGES OF CLASSIFICATION TREES

Some disadvantages of trees isolated in this study are:

1. Classification trees (and neural networks) need large data sets to perform at their best. Discriminant analyses perform well even with small data sets.
2. Some single trees are unstable classifiers (due to the messy nature of certain data sets), but the methods in Chapter 3 may eliminate this problem.
3. In some cases the best tree may be too big and/ or too complicated to interpret.
4. When using the methods in Chapter 3 and 4:
   - Further complications are introduced when interpreting the results.
   - The resampling methods may not always work for small data sets.
   - Resampling methods increase the computation time of growing trees. Sometimes the error rates are not significantly reduced by these methods, and sometimes it may be higher.

CHAPTER 8

General remarks and conclusions

8.1 CONCLUSION

Using graphical images, the human eye is by far the best in recognizing patterns. But since human ability is restricted to a maximum of 3 dimensional problems, classification methods are good approximations in recognizing patterns undetectable by the human eye. Especially when dealing with p-dimensional ($p>3$) problems. Classification methods play a prominent role in research and cannot be isolated from the practical demands of many research problems. It helps in making clever and effective decisions by unlocking valuable information, in high dimensional databases.

The theory of classification trees is mathematically much simpler than the theory of other classification methods. Classification trees have been researched and developed quite extensively, making it a more flexible technique than discriminant analyses. The output from trees is very simple and involves no advanced mathematical or statistical expertise to understand. Trees are very computer intensive methods and no analyst will ever attempt doing such an analysis by hand. The availability of effective tree algorithms in special purpose software such as CART, STATISTICA, S-Plus, SAS Enterprise Miner, etc., make trees an attractive modeling technique.

In many experiments and surveys data are collected that are categorical of nature. Classical statistical classification methods have been developed for numerical data and uses distance metrics. Categorical data possess no distance and in such a case, the classical statistical methods will be less appropriate as an analysis tool. Classification trees are very robust and make no assumptions about the distribution of data. Traditional methods cannot handle missing values and these values are usually estimated or deleted. By deleting missing values in data, valuable information is lost and by estimating these values, one creates an artificial data set. Due to Breiman *et al.* (1984), classification trees

can handle missing values by means of surrogate splits on single variables. When using linear combinations, handling the missing values becomes a problem. The flexibility of classification trees provides the analyst with a steering wheel with which he can drive the classification into a more desirable direction. Given the many options that can be exercised with trees, it seems almost certain that you would end up with a good classification analysis. However, a good result is not always guaranteed since the data might be completely stochastic and no real patterns might exist.

The user-friendly CART software is an easy-to-use and easy-to-learn analysis package. Judging by the user comments mentioned in Chapter 5, the software has already gained popularity, especially with analysts investigating big databases with high dimensionality and a mixture of data types. Using CART to analyze the data in Chapter 7 showed the power of classification trees and makes one realize its advantages. The popular SAS software (SAS Base and SAS Enterprise Miner) are also very user-friendly, but more sophisticated than CART.

In this study, no Bagging or Boosting was applied to neural network and discriminant analyses. Breiman (1998) demonstrated that discriminant analysis is stable and that Bagging and Boosting do not have a significant effect on reducing the error rate. Opitz and Maclin (1999) showed that neural network classifiers are sometimes unstable and that Boosting and Bagging applied to this method will increase stability and may even produce a more accurate classifier than classification trees. Many studies on Bagging and Boosting classifiers have shown that the misclassifications decrease tremendously when the number of trees or neural networks in resampling are increased. In some cases Boosting may outperform Bagging in reducing the error rate. However, as the results of Opitz and Maclin (1999) also show, the error rate reaches a plateau as the number of trees increases. Bagging reaches a plateau faster than Boosting and you can therefore use fewer trees as a classifier. Boosting uses quite a number of trees as a classifier before it reaches a plateau in the misclassification rate. As seen in this study and as stated by Opitz and Maclin (1999), the effect of Bagging and Boosting is dependent on the nature of the data set under investigation.

Through the analyses of the data in this text, I realized that classification trees have many distinct advantages over other methods. It is an explorative technique and lets

the data "speak for itself". No transformation is needed to meet certain assumptions. Most of the advantages are mentioned in Chapter 1. Of the methods discussed in this text, classification trees and parametric discriminant analysis provide the most usable/ reportable information. Classification trees provide a nice hierarchical tree with a quite logical and informative interpretation. LDA or QDA creates a linear or quadratic classification function. Classification trees and parametric discriminant analysis can classify new cases based on a tree and a linear or quadratic classification function derived from $\mathcal{L}$, without the aid of computer software. Thus they are model-based classifiers and they give you more insight into the separation power of the explanatory variables. Non-parametric discriminant analyses and neural networks are less informative. They exhibit a "black-box" behaviour, but still provide excellent classifications. These two methods can only classify new cases with the aid of computer software by using the calibration information obtained in $\mathcal{L}$. Non-parametric discriminant analyses and neural networks have no visually appealing classification function, and use historical data ($\mathcal{L}$) as reference for future classifications. They are known as memory-based classifiers.

Single trees give the analyst a picture of which variables are important in discriminating among classes, as well as how the separation of the classes came about. The application of classification trees should depend on the nature of the data observed and the purpose of the research being done. If the purpose is accurate classification, the analyst should at least try some of the other methods to see if classification is better. If the purpose is understanding the data, the separation power of variables, uncovering hidden structures or learning stories from the data, much value will be added to research by the use of classification trees. Having just an accurate classifier is not what researchers want all the time, but rather a means of discovering hidden patterns in a data set, which can lead to new theories and new research breakthroughs. There is no doubt that classification trees will add value to research in other fields.

8.2 FUTURE WORK

As stated in the beginning of this text, the focus was to explore classification trees in comparison with other classification methods. Breiman *et al.* (1984) and Breiman (1996, 1998, 2001) provided the foundation, and many developments of trees, but there are still some desirable aspects one might consider in future work on trees.

- The methods in Chapter 3 bring a new dimension to classification methods. One disadvantage of Bagging and Boosting trees, is the loss of their interpretability. Here, the hierarchical structure of the classifier is sacrificed to obtain better classifications. Since each tree in the committee of experts still possesses a hierarchical structure, it might be worthwhile in future research to explore the possibility of obtaining a pooled hierarchical tree from all the trees in the committee. The possible advantage of such a pooled tree is that it might be a better classifier than a single tree, and might provide better insight into the data, based on the aggregation of several classification trees.
- Aggregating the important variables in the individual trees might also give more insight and might lead to better conclusions from the analysis of the data.
- How should one handle data sets where some classes are relatively small? As stated in Chapter 3 and 7, resampling methods may not work properly in this case. For a classifier to work right, cases of each class should be present in the bootstrap samples.

8.3 REMARKS

Although classification methods recognize the patterns in natural phenomena very well, nature will always keep dragging us around by the nose. The analyst not only has to acquire advanced knowledge of methods, but has to understand the data and the research problem very well. The analyst should also keep an open mind and explore other possible solutions. Hastie *et al.* (2001) give references for, and explain many of the recent ideas in supervised learning, including the methods in this study. Other methods include Support Vector Machines, Logistic Regression, Multivariate Adaptive Regression Splines (MARS), Naïve Bayes, Additive Models, etc. With the ongoing developments in computing software and the many data sets generated from different fields, statisticians

and researchers in other fields, such as Data Mining, Machine Learning and Artificial Intelligence, are continuously developing new ideas for classification procedures. With so many possibilities, choosing the appropriate classification method must be based on the research environment, the data that it generates, and what we want to learn from an analysis.

"*An appropriate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.*"

- John Wilder Tukey

Appendix A

An example of SAS code to perform parametric and non-parametric discriminant analyses, and the classification of new observations.

```
title 'The Satellite data';
options ls=78 ps=62;

                    /* Importing the learning sample */

data learn;
 infile 'O:\Morne_Lamont\thesis\data\satellite.csv' delimiter=',';
 input x1-x36 Group @;
output;
cards;

                    /* Importing the test sample */

data test;
 infile 'O:\Morne_Lamont\thesis\data\satellitetest.csv' delimiter=',';
 input x1-x36 Group @;
output;
cards;

                    /* Performing the analyses */

proc discrim data=learn outstat=parametric method=normal pool=yes
crossvalidate;
title2 'Linear discriminant analysis';
  class Group;
  var x1-x36;
run;

proc discrim method=normal data=parametric testdata=test testlist
testout=out1;
   class Group;
   var x1-x36;
run;

proc discrim data=learn method=npar kernel=normal metric=full r=4
crossvalidate testdata=test testlist testout=out2;
title2 'Kernel method with normal kernel, r=4 and Mahalanobis
distance';
  class Group;
  var x1-x36;
run;

proc discrim data=learn method=npar k=6 metric=full crossvalidate
testdata=test testlist testout=out3;
title2 'k-nearest-neighbour method with k=6 and Mahalanobis distance';
   class Group;
   var x1-x36;
run;
```

Appendix B
Brief descriptions of data sets used.

Example 1: Hemophilia data

This data set is an example taken from Johnson & Wichern (1992). It comes from a study of 75 cases, concerned with the detection of hemophilia A carriers.

Class Distribution:

Class 1 (Non carriers) : 30 (40%)

Class 2 (Obligatory carriers): 45 (60%)

Explanatory variables taken on the log scale:

1. Log_{10} (AHF activity)
2. Log_{10} (AHF antigen)

An independent data set of 10 cases is also provided with unknown groups/classes, which need to be classified into a class.

Example 2: Australian credit approval data

This is credit card application data. The data set of size 690 contains a binary response with classes 0 and 1, 6 numerical and 8 categorical explanatory variables.

Class Distribution:

Class 0: 383 (55.5%)

Class 1: 307 (44.5%)

Note: *Examples 2, 3 & 4 were obtained from the data library in the CART software.*

Example 3: Diabetes data

This data set has a binary response (1 or 2), which is the response of whether a patient shows signs of diabetes according to World Health Organization criteria. (i.e. if the 2 hour post-load plasma glucose was at least 200 *mg/dl* at any survey examination or if found during routine medical care).

Class Distribution:

Class 1: 500 (65.1%)

Class 2: 268 (34.9%)

Explanatory variables:

Data set consists of 8 numerical explanatory variables.

Example 4: Heart disease data

This data set contains 270 cases with binary response absence (1) or presence (2) of heart disease.

Class Distribution:

Class 1: 150 (55.6%)

Class 2: 120 (44.4%)

Explanatory variables:

This data set contains 13 variables (which have been extracted from a larger set of 75). Variable types [6 numerical; 1 ordered; 3 binary and 3 nominal]

Example 5: School data

This is data from a study done in the Economics Department at Stellenbosch University. It contains measurements observed on school children. The response variable was the Winged Scapulae of children and had two classes.

Class Distribution:

Class	*Total*	*Learning sample*	*Test sample*
1	120	89	31
2	168	127	41
Total	**288**	**216**	**72**

Explanatory variables:

There are 14 variables [4 Categorical and 10 Numerical]

Example 6: Ship data

This subset of data comes from a study at the Institute for Maritime Technology. The details of this study are given in Chapter 7, Section 7.4.

Class Distribution:

Class	*Total*	*Learning sample*	*Test sample*
0	343	257	86
1	662	497	165
Total	**1005**	**754**	**251**

Explanatory variables:

There are 19 variables [5 Categorical and 14 Numerical]

Note: *Examples 5 &6 are data sets used with permission from Dr. M. Kidd at the Centre for Statistical Consultation, Stellenbosch University.*

Example 7: German credit data

This data set is an example in the data library of SAS Enterprise Miner. It has a binary response variable of good or bad credit ratings. It contains 1000 records and 20 explanatory variables (numerical, nominal and ordinal data types).

Class Distribution:

Class	*Total*	*Learning sample*	*Test sample*
0	700	519	181
1	300	231	69
Total	**1000**	**750**	**250**

Explanatory variables:

There are 7 numerical and 13 categorical variables.

Example 8: SA Heart data

This data is a subset of the Coronary Risk-Factor Study baseline survey, carried out in three rural areas in the Western Cape, South Africa. (See Hastie *et al.*, 2001; *http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/*). The data represent white males between 15 and 64 years of age and the response variable is the presence (1) or absence (0) of myocardial infarction.

Class Distribution:

Class	*Total*	*Learning sample*	*Test sample*
0	302	224	78
1	160	123	37
Total	**462**	**347**	**115**

Explanatory variables:

There are 8 numerical and 1 categorical variable.

Example 9: Glass data

This database was created in the Central Research Establishment, Home Office Forensic Science Service Alderman, Reading, Berkshire. Each case consists of 9 chemical measurements on one of 6 types of glass. There are 214 cases.

Class Distribution:

Class	*Total*	*Learning sample*	*Test sample*
1	70	47	23
2	76	62	14
3	17	13	4
4	13	9	4
5	9	7	2
6	29	23	6
Total	**214**	**161**	**53**

Explanatory variables:

There are 9 numerical variables.

Example 10: Satellite Image data

The database consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values.

Class Distribution:

Class	*Total*	*Learning sample*	*Test sample*
1	1533	1072	461
2	703	479	224
3	1358	961	397
4	626	415	211
5	707	470	237
6	-	-	-
7	1508	1038	470
Total	**6435**	**4435**	**2000**

Explanatory variables:

There are 36 numerical variables, in the range 0 to 255. No missing values.

There are 7 classes: 1 (Red soil), 2 (Cotton crop), 3 (Grey soil), 4 (Damp grey soil), 5 (soil with vegetation stubble), 6 (mixture class (all types present)), 7 (very damp grey soil)

NB. There are no cases for class 6 in this data set - they have all been removed because of doubts about the validity of this class.

Note: *Examples 9 & 10 are data sets taken from http://oz.berkeley.edu/users/breiman/*

References

1. Bauer, E. & Kohavi, R., "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants", *Machine Learning*, **36** (1999), 105-142.
2. Bennett, S. & Bowers, D., *An Introduction to Multivariate Techniques for Social and Behavioural Sciences*, 1976.
3. Berthold, M. & Hand, D.J., *Intelligent Data Analysis: An Introduction*, Springer, 1999.
4. Breiman, L., "Arcing Classifiers", *Annals of Statistics*, **26** (1998), 801-824.
   (The reference for this study is a technical report by Leo Breiman with the same title found at *http://www.salford-systems.com/*)
5. Breiman, L., "Bagging Predictors", *Machine learning*, **26,** no. 2 (1996), 123-140.
   (The reference for this study is a technical report by Leo Breiman with the same title found at *http://www.salford-systems.com/*)
6. Breiman, L., "Random Forests", Statistics Department, University of California at Berkeley, CA 94720, January 2001. (*http://oz.berkeley.edu/users/breiman/*)
7. Breiman, L., Friedman, J., Olshen, R. & Stone, C., *Classification and Regression Trees*, Chapman and Hall, 1984.
8. CART® for Windows User's Guide: A Salford Systems Implementation of the original CART Program, 2000.
9. Chambers, J.M. & Hastie, T.J., *Statistical Models in S*, Chapman and Hall/CRC, 1997.
10. Cheng, B. & Titterington, D.M., "Neural Networks: A Review from a Statistical Perspective", *Statistical Science*, **9**, no. 1 (1994), 2-54.
11. Ciampi, A. & Lechevallier, Y., "Statistical Models as Building Blocks of Neural Networks", *Commun. Statist.-Theory Meth.*, **26**, no 4 (1997), 991-1009.
12. Course notes on Pattern Recognition found on the Oxford University website: (*http://www.stats.ox.ac.uk/~northrop/teaching/PAN/*)
13. Debeljak, M., Džeroski, S., Jerina, K., Kobler, A. & Adamič, M., "Habitat suitability modeling for red deer (Cervus elaphus L.) in South-central Slovenia with

classification trees." *Elsevier Science B.V., Ecological modeling,* **138** (2001), 321-330.

14. Efron, B. & Tibshirani, R., *An Introduction to the Bootstrap*, Chapman and Hall, 1993.
15. Enterprise Miner ^{TM}: Applying Data Mining Techniques Course Notes, SAS Institute Inc., NC 27513, 1999.
16. Freund, Y. & Schapire, R.E., "A Short Introduction to Boosting", *Journal of Japanese Society for Artificial intelligence*, (September 1999), 14(5): 771-780.
17. Garson, G.D., *Neural Networks: An introductory guide for social scientists*, Sage publications Ltd., 1998.
18. Gorman, R.P., & Sejnowski, T. J. "Analysis of Hidden Units in a Layered Network to Classify Sonar Targets," *Neural Networks*, **1** (1988), 75-89.
19. Hastie, T., Tibshirani, R. & Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction,* Springer, 2001.
20. Johnson, R.A. & Wichern, D.W., *Applied Multivariate Statistical Analysis*, 3rd edition, Prentice-Hall International, Inc, 1992.
21. Klecka, W.R., *Discriminant analysis*, a SAGE University paper, 1980.
22. Kowalski, B.R. & Bender, C.F., "Pattern Recognition. A Powerful Approach to Interpreting Chemical data", *Journal of the American Chemical Society* |94:16|, August 9, 1972.
23. Kowalski, B.R. & Kwan, W.O, "Classification of wines by applying pattern recognition to chemical composition data", *Journal of Food Science*, **43**, 1978.
24. Krishnaiah, P.R. & Kanal L.N., Handbook of Statistics 2, *Classification, Pattern recognition and reduction of dimensionality*, North-Holland publishing company, 1982.
25. Latorre M.J., García-Jares C., Mèdina B., Herrero C., "Pattern Recognition Analysis Applied to Classification of Wines from Galicia (Northwestern Spain) with Certified Brand of Origin", *American Chemical Society*, 1994.
26. Manly, B.F.J., *Multivariate Statistical Methods – A Primer*, Chapman and Hall, 1986.
27. Meléndez M.E., Sánchez M.S., Íñiguez M., Sarabia L.A. & Ortiz M.C., "Psychophysical parameters of colour and the chemometric charaterisation of wines

of the certified denomination of origin Rioja", *Elsevier, Analytica Chimica Acta*, **446** (2001), 159-169.

28. Mendenhall, W., Wackerly, D. & Scheaffer, R. *Mathematical Statistics with Applications*, 4th edition, Duxbury Press: Wadsworth, Inc., 1990.
29. Ogden, R.T., *Essential Wavelets for Statistical Applications and Data Analysis*, Birkhauser Boston, 1997.
30. Opitz, D. & Maclin, R., "Popular Ensemble Methods: An Empirical Study", *Journal of Artificial Intelligence Research*, **11** (1999), 169-198.
31. Pan, W., "Shrinking classification trees for bootstrap aggregation", *Elsevier Science B.V., Pattern Recognition letters*, **20** (1999), 961-965.
32. Salford Systems: (*http://www.salford-systems.com/*)
33. SAS Institute Inc., SAS/STAT® User's Guide, Version 8, Cary, NC: SAS Institute Inc., 1999.
34. Spector, P., *An Introduction to S and S-Plus*, Duxbury press: Wadsworth, Inc., 1994.
35. S-Plus® Guide to Statistics, Vol. 1, Modern Statistics and Advanced Graphics. Data Analysis Products Division, Mathsoft, Seattle, WA., 2000.
36. Srivastava, M.S. & Carter, E.M., *An Introduction to Applied Multivariate Statistics*, North-Holland, 1983.
37. StatSoft. Inc.: (*http://www.statsoftinc.com/textbook/stathome.html*)
38. Vichi, M. & Opitz, O., "Classification and Data analysis. Theory and Application", *Preceedings of the Biannual Meeting of the Classification Group of Società Italiana di Statistica (SIS) Pescara*, Springer, July 3-4 1997.
39. Warner, B. & Misra, M., "Understanding Neural Networks as Statistical Tools", *The American Statistician*, **50**, no. 4 (November 1996).