

*The University of Melbourne,
Department of Mathematics and Statistics*

Identifying risk factors associated with new onset cardiovascular disease in patients with type I diabetes using Classification Tree

Fanny Sampurno

SUPERVISOR: DR. KEN SHARPE
CO-SUPERVISOR: A/PROF MERLIN C THOMAS

Honours Thesis, November 2006.

Contents

1	Introduction	4
1.1	Classification and Regression Tree (CART)	5
1.2	The Statistical Problem	6
2	Tree Building	7
2.1	Basic Idea of tree diagram	7
2.2	Splitting a Node	9
2.2.1	Node Impurity	11
2.3	Class Classification	15
2.3.1	Prior probability	15
2.3.2	Unequal cost of Misclassification	16
2.4	Incorporating Losses	18
3	Determining when to stop splitting	21
3.1	Resubstitution Cost	22
4	Tree Pruning and Optimal Tree Selection	24
4.1	Tree Cost Complexity	26

4.2	k-fold Cross validation	30
4.3	Standard Error of R^{cv*}	33
5	Missing Variables	36
6	Disadvantage of CART	38
7	Advantage of CART	40
8	Concluding Remarks	42

Abstract

This study examines type I diabetes patients without a history of cardiovascular complications, to define which risk factors are most important in the pathogenesis of cardiovascular disease (CVD) in diabetes. Recursive Partitioning and Regression Trees (RPART) statistical software was used for data analysis. CART models were used to investigate which risk factors predict CVD events during nine year follow-up in 136 type I diabetic patients aged 20-62 years.

It is found that diabetes patients aged over 46.68 years and with IDL% TG level less than 29.94 are most at risk of cardiovascular complication over the next nine years. As well, patients younger than 46.68 and with HbA1 level greater than 12.75 are at risk of developing CVD complication.

Chapter 1

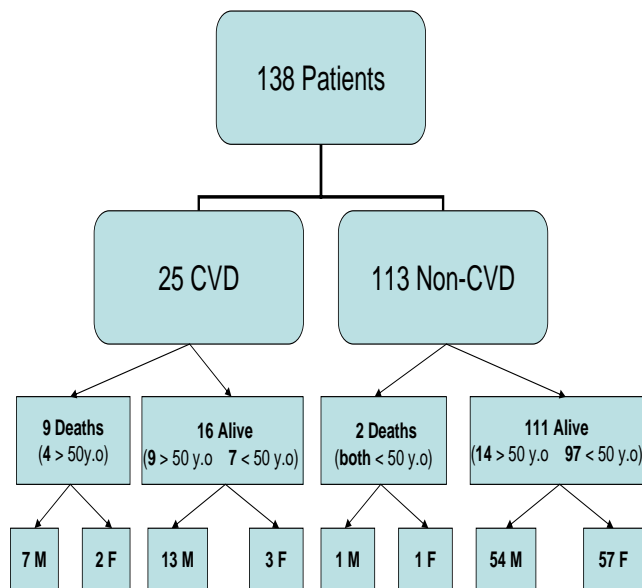
Introduction

A total of 138 patients with type I diabetes and no previous history of cardiovascular disease (CVD) (75 males and 63 females) were recruited for a cohort study. Participants were then followed for a mean of 8.7 years, during which time they received standard care.

Standard clinical data were obtained from patient's records including age, gender, age of onset and duration of diabetes, medication history, anthropometric indices (height, weight, BMI), indices of glycaemic control such as insulin dose, FPG, Hb1Ac measurement and the presence of existing microvascular complications (nephropathy, retinopathy, neuropathy), and some lipid variables.

Altogether, 138 variables relating to a patient's background were recorded with the aim to learn which variables best predict whether or not the patient is likely to have cardiovascular complication. Cardiovascular events were defined as coronary heart disease (myocardial infarction, coronary revascularization or angioplasty), cerebrovascular disease (stroke, transient ischemic attack), or peripheral vascular disease (claudication, amputation or revascularization) based on clinical records.

Twenty-five patients developed cardiovascular complications during the study follow-up period. There were 11 deaths; nine of these deaths were due to cardiovascular causes, one due to malignancy and one suicide. The diagram below illustrates the patients involved in the study.



M= male, F= female

1.1 Classification and Regression Tree (CART)

CART builds classification and regression trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification). Regression-type problems are generally those where one attempts to predict the value of a continuous variable from one or more continuous and/or categorical predictor variables. Whereas classification-type problems are generally those where one attempts to predict values of a categorical dependent variable (class, group membership, etc.) from one or more continuous and/or categorical predictor variables. This project will focus on a simple binary classification problem, where the categorical dependent variable can only assume one of two distinct and mutually exclusive values.

CART analysis is a form of binary recursive partitioning. The term “*binary*” implies that each group of patients, represented by a “node” in a decision tree can only be split into two groups. Thus, each node can

be split into two child nodes, in which case the original node is called a parent node. The term “*recursive*” refers to the fact that the binary partitioning process can be applied over and over again. Thus, each parent node can give rise to two child nodes and, in turn, each of these child nodes may themselves be split, forming additional children. The term “*partitioning*” refers to the fact that the dataset is split into sections or partitioned.

CART analysis consists of four basic steps. The first step consists of tree building, during which a tree is built using recursive splitting nodes. A split forms two new nodes, a left child node and a right child node. Each is assigned a prediction, whether a classification or a mean (for a regression problem). The assignment of a predicted class to each node occurs whether or not that node is subsequently split into child nodes. The second step consists of stopping the tree building process. At this point a “maximal” tree has been produced, which probably greatly overfits the information contained within the learning data set. The third step consist of tree “pruning”, which results in the creation of a sequence of simpler and simpler trees, through the cutting off of increasingly important nodes. The fourth step consist of optimal tree selection, during which the tree which fits the information in the learning dataset, but does not overfit the information, is selected from among the sequence of pruned trees. Each of these steps will be discussed in more details in the following chapters.

1.2 The Statistical Problem

Given an outcome variable, Y, and a set of p predictors, x_1, x_2, \dots, x_p . The x 's will be regarded as fixed variables, and Y is a random variable. The statistical problem is to establish a relationship between Y and the x 's so that it is possible to predict Y based on the values of the x 's. Mathematically, we want to estimate the conditional probability of the random variable Y,

$$P(Y = y|x_1, x_2, \dots, x_p).$$

Research question: Which patients are most at risk of cardiovascular complication?

A list of Candidate Predictor Variables

Variable Name	Label	Type	Range/levels
Gender	x_1	Nominal	Male,Female
Age of Onset	x_2	Continuous	2-37 years
Age at Baseline	x_3	Continuous	18-62 years
Duration of Diabetes	x_4	Continuous	7-47 years
⋮	⋮	⋮	⋮

Chapter 2

Tree Building

2.1 Basic Idea of tree diagram

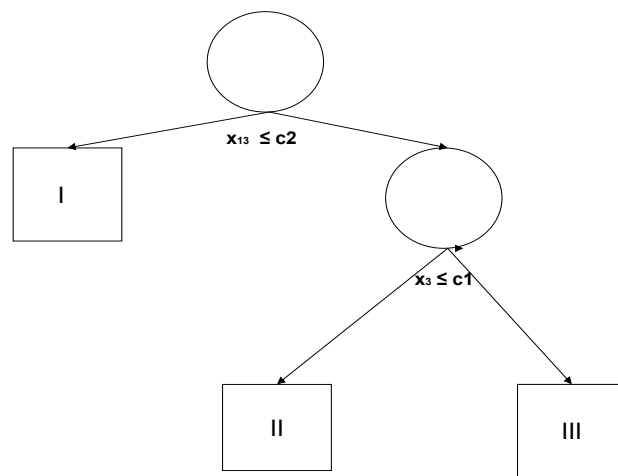


Figure 2.1: An Illustrative Tree Diagram.

The tree in Figure 2.1 has three layers of nodes. The first layer is the root node, namely, the circle on the top. One internal (the circle) node is in the second layer, and three terminal (the boxes) nodes are respectively in the second and third layers. Here, the root node can also be regarded as an internal node. Both the root and

internal nodes are partitioned into two nodes in the next layer which are called left and right child nodes. By definition, however, the terminal nodes do not have offspring nodes.

The root node contains a sample of subjects from which the tree is grown. Those subjects constitute the so-called learning data set. Tree building begins at the root node, which includes all subjects in the learning dataset. For our data, the root node contains all 136 type I diabetes patients who were the study subjects. Note that the two patients which died due to non-CVD related death were excluded from the analysis. All nodes in the same layer constitute a partition of the root node. The partition becomes finer and finer as the layer gets deeper and deeper. Therefore, every node in a tree is merely a subset of the learning data set.

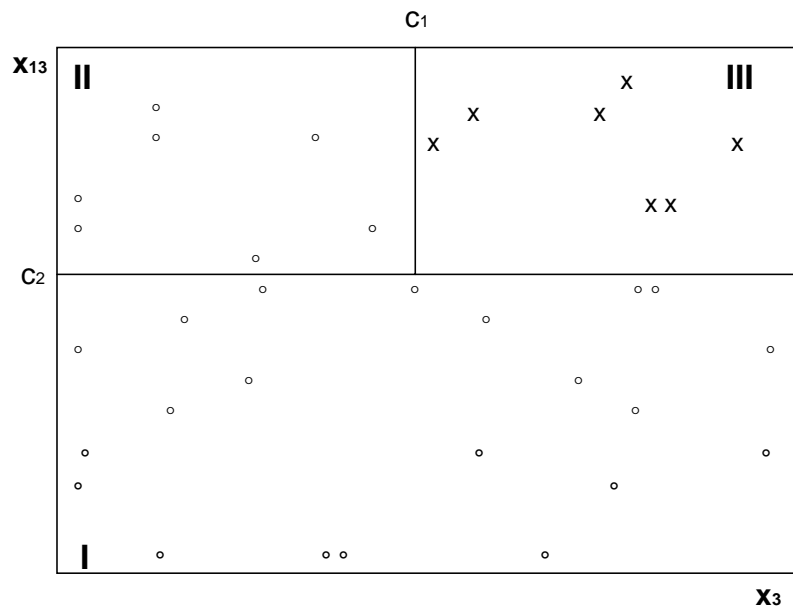


Figure 2.2: An Illustrative Tree Structure.

Figure 2.2 illustrates a hypothetical situation. Let a cross denote a patient who ends up having a cardiovascular (CVD) complication and a circle denote a patient who does not, after the 9 years of follow up. The two coordinates represent two covariates, say x_3 (age at baseline) and x_{13} (BMI). We can draw two line segment to separate the crosses from the circles and obtain three disjoint regions: (I) $x_{13} \leq c_2$; (II) $x_{13} > c_2$ and $x_3 \leq c_1$; and (III) $x_{13} > c_2$ and $x_3 > c_1$.

Figure 2.1 is a tree representation of this separation. First, we put both the crosses and the circles on to the root node. The root node which contains all patients, is split in two, analogous to a horizontal line being drawn at $x_{13} = c_2$. The two arrows below the root node direct a cross or circle to terminal node I or the internal node in the second layer, depending on whether or not $x_{13} \leq c_2$. Those with $x_{13} > c_2$ are further directed to terminal nodes II and III based on whether or not $x_3 \leq c_1$. Hence, the nodes in Figure 2.1 correspond to the regions in Figure 2.2. When we draw a line to separate a region, it amounts to partitioning a node in the tree.

The essence of recursive partitioning is to try to achieve terminal nodes that are homogeneous in the sense that they contain either crosses or circles. On the other hand, terminal nodes are heterogeneous if they contain both crosses and circles. Because crosses and circles represent CVD and non-CVD groups of patients, respectively, it would suggest that all type I diabetes patients who are older than a certain age (c_1) and who have a BMI greater than a certain value(c_2) will have a CVD complication within 9 years.

Complete homogeneity of terminal nodes is an ideal that is rarely realized. Thus, the objective of partitioning is to make the contents of the terminal nodes as homogenous as possible. A quantitative measure of the extent of node homogeneity is the notion of node impurity. The simplest such measure is

$$\frac{\text{Number of patients with CVD complication in a node}}{\text{Total number of patients in the node}}$$

The closer this ratio is to 0 or 1, the more homogeneous the node.

2.2 Splitting a Node

The fundamental idea in the construction of the tree classifier is to select each split of a subset so that the data in each of the descendant subsets are “purer” than the data in the parent subset.

Beginning with the root node, the RPART software finds the best possible variable to split the node into two child nodes. RPART checks all possible splitting variables (called splitters) that could be used to split the node. In choosing the best splitter, RPART seeks to maximize the average “purity” of the two child nodes.

This section will describe the process of why and how a parent node is split into two child nodes. Consider variable x_3 (age at baseline). There are 136 different age values in the range of 18.86-61.43 years in the 136 study subjects. Thus, we can split the root node in 135 different ways based on the age of patients when

they enter the study. In general, for a continuous predictor, x_j , the number of allowable splits is one fewer than the number of its distinctly observed values.

If one or more predictors is categorical it is slightly more complex because every combination of predictor and response categories has to be evaluated. The number of potential splits is $2^{(k-1)} - 1$ where k is the number of categories of the predictor variable. The fact that a power function is involved means that the number of potential splits increases very rapidly as the number of categories increases, as shown below. The number of splits is low when there are few categories but beyond eight categories the number of possible splits starts to become very large.

Categories (k)	Splits
2	1
3	3
4	7
5	15
8	127
16	32767
32	2147483647

Adding together the numbers of allowable splits from the 138 predictors, we have a lot of possible ways to divide the root node into two subnodes. The basic question to be addressed now is, how do we select one or several preferred splits from the pool of allowable splits?

Before selecting the best split, we must define the goodness of a split. What we want is a split that results in two pure (or homogeneous) child nodes. However, in reality the child nodes are seldom completely homogenous. Therefore, the goodness of a split must weigh the homogeneities (or the impurities) in the two child nodes. If we take age at baseline as a splitting covariate and consider a cutoff at c , as a result of the question “is $x_3 > c$?” we have the following table:

		Non-CVD (Y=0)	CVD (Y=1)	
Left Node (t_L)	$x_3 \leq c$	n_{11}	n_{12}	$n_{1.}$
Right Node (t_R)	$x_3 > c$	n_{21}	n_{22}	$n_{2.}$
		$n_{.1}$	$n_{.2}$	n

Table 1 Illustration of split

Note, $Y = 1$ if a patient ends up having a CVD complication and $Y = 0$ otherwise. We estimate $P(Y=1|t_L)$ and $P(Y=1|t_R)$ by $n_{12}/n_{1.}$ and $n_{22}/n_{2.}$, respectively.

2.2.1 Node Impurity

A node is pure (impurity = 0) when all classes have the same value for the response or target variable. A node is impure if classes have more than one value for the response. The impurity is one measure for assessing the node homogeneity in the sense that the node contains observations of a single class.

In a binary outcome, the impurity of a node is a function of the relative frequencies of the classes in that node:

$$i(t) = \psi(p(Y = 0|t), p(Y = 1|t)), \text{ where } t \text{ denotes some node of a tree}$$

One measure of impurity called the *Gini criterion* is defined as

$$i(t) = \sum_{i \neq j} p(i|t)p(j|t)$$

For the two-class case,

$$i(t) = 2p(0|t)p(1|t)$$

and the impurity function is illustrated in the graph below

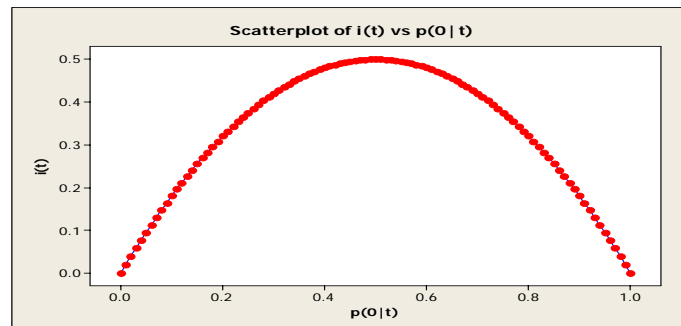


Figure 2.3: Plot of impurity function based on Gini criterion.

Sensible requirements of any quantification of impurity:

- Should be a maximum when the observations are distributed evenly over all classes.

$$i(t) = f\left(\frac{1}{2}, \frac{1}{2}\right) = \text{maximum}$$

- Should be a minimum when all observations belong to a single class.

$$i(t) = f(0, 1) = f(1, 0) = 0$$

- Should be a symmetric function of $p(Y=0)$, $p(Y=1)$.

$$f(p(0), 1 - p(0)) = f(1 - p(0), p(0))$$

Hence, the notion of Gini impurity in the left child node is defined as

$$i(t_L) = 2 \frac{n_{11}}{n_1} \frac{n_{12}}{n_1}$$

Likewise, the impurity in the right child node is defined as

$$i(t_R) = 2 \frac{n_{21}}{n_2} \frac{n_{22}}{n_2}$$

The impurity in the parent node :

$$i(t_P) = 2 \frac{n_{.1}}{n} \frac{n_{.2}}{n}$$

The goodness of a split (or the change in impurity) is then determined through the expression

$$\Delta i(t) = i(t_P) - (P(t_L)i(t_L) + P(t_R)i(t_R)),$$

Where $P(t_L)$ and $P(t_R)$ are the probabilities that a subject falls into nodes t_L and t_R , respectively.

$$P(t_L) = \frac{n_{1.}}{n_{1.} + n_{2.}}$$

and

$$P(t_R) = \frac{n_{2.}}{n_{1.} + n_{2.}}$$

The split producing the greatest change (degree of reduction) in the impurity is then selected.

To appreciate those concepts in more detail, let us go through a concrete example. If we take $c= 46.54$ as the age of baseline threshold, we have the 2×2 table

		Non-CVD (Y=0)	CVD (Y=1)	
Left Node (t_L)	$x_3 \leq 46.54$	94	10	104
Right Node (t_R)	$x_3 > 46.54$	17	15	32
		111	25	136

Calculating the change in impurity for every subject's age at baseline we have

	$i(t_L)$	$i(t_R)$	$i(t_P)$	$P(t_L)$	$P(t_R)$	$\Delta i(t)$
18.86	0	0.3018	0.3001	0.0074	0.9926	0.0005
20.58	0.5000	0.2941	0.3001	0.0147	0.9853	0.0030
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
46.22	0.1753	0.4959	0.3001	0.7574	0.2426	0.0470
46.54	0.1738	0.4980	0.3001	0.7647	0.2353	0.0500
46.82	0.1876	0.4953	0.3001	0.7721	0.2279	0.0423
47.87	0.2008	0.4911	0.3001	0.7794	0.2206	0.0352
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Table 2 Change in Impurity of Age at Baseline Variable

From the above table, we see that the greatest reduction in impurity comes from age 46.54 with $\Delta i(t)= 0.05$. RPART used the value of 46.68, which is the average of 46.54 and 46.82.

The best age at baseline split is used to compete with the best splits from the other 137 predictors. Due to the huge amount of computation that needs to be done, we use RPART software to construct the tree diagram. At the root node, a search was made through all of the candidate splits to find the one which gave the largest decrease in impurity. From Figure 4.4 (Saturated Tree from RPART), we can see that age at baseline is used to split the root node. Thus, age at baseline < 46.68 produced the greatest $\Delta i(t)$.

Sample Tree

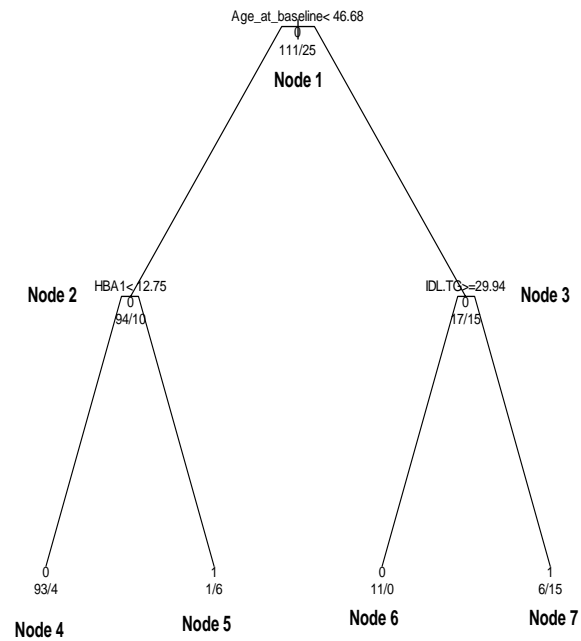


Figure 2.4: Illustration of Tree.

After splitting the root node, we continue to divide its two child nodes. The partitioning principle is the same. For example, to further divide node 2 in Figure 2.4 into nodes 4 and 5, we repeat the previous partitioning process with minor adjustment. That is the partition uses only the 104 patients whose age at baseline ≤ 46.68 , the remaining 32 patients with age at baseline > 46.68 are put aside. One important message is that an offspring node may use the same splitting variable as its ancestors. In the same way, we can divide node 3 as we did for node 2. After we finish node 3, we go on to nodes 4 and 5, and so on. This is the so-called *recursive partitioning* process. Because we partition one node into two nodes only, the resulting tree is called a *binary tree*.

2.3 Class Classification

Once a terminal node is found we must decide how to classify all cases falling within it. How does the classification procedure work? In other words, how are the node classes assigned? Each node, even the root node, is assigned a predicted class because each node has the potential for being a terminal node. The predicted class assigned to each node depends on two factors: (1) the assumed prior probability of each class within future datasets and (2) the misclassification cost. We will explain these two factors shortly.

Our goal in classification tree analysis is to obtain the most accurate prediction possible. The most accurate prediction is defined as the prediction with the minimum cost, where *cost* simply correspond to the proportion of subjects misclassified by a tree. This method of node class assignment ensures that the tree has a minimal expected misclassification cost (the lowest misclassification error rate) for future datasets similar to the learning dataset in which the probability of each outcome is equal to the assumed prior probabilities. The expected misclassification cost will be discussed in section 2.3.2.

2.3.1 Prior probability

Let $f(x|Y=1)$ represent the density function for x from population 1 and $f(x|Y=0)$ be the density for x from population 0. Suppose that the patient has baseline $(x_1, x_2, x_3, \dots, x_p)$

\mathbf{X} is a $p \times 1$ vector of observations

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

Population 1 = Group of patients with cardiovascular complication (CVD)

Population 0 = Group of patients with no cardiovascular complication (non-CVD)

Let $P(Y = 1)$ and $P(Y = 0)$ be *priors*, or, a priori probabilities. Priors specify how likely it is, without using any prior knowledge of the values for the predictor variables (before observing x 's) in the model, that a patient will fall into one of the classes (population 1 or 0), where $P(Y = 0) = 1 - P(Y = 1)$. In other word, the best estimate (guess) of the 9-year incidence rate for CVD complication among the underlying population (patient with type I diabetes) from which the subjects are selected.

According to Bayes' theorem, the rule that minimizes the total probability of misclassification is: Assign x to population 1 if

$$P(Y = 1)f(x|Y = 1) > P(Y = 0)f(x|Y = 0)$$

and to population 0 otherwise.

If the priors $P(Y = j)$ are estimated by N_j/N , Breiman called this the *plurality rule*, where the group with the greatest representation determines the class assignment. So in tree classification, we can classify a node t as class j for which $N_j(t)$ is largest. That is classify t to class 1 if

$$P(Y = 1|t) > P(Y = 0|t)$$

and to class 0 otherwise.

Note: In many applications such as prospective studies, the prior probabilities (the 9-years incidence rate, $P(Y = CVD)$) can be estimated by the class proportions of the learning dataset. At other times, if you have specific knowledge about the incidence rate (for example, based on previous research), then one would specify priors in accordance with that knowledge. The general point is that the relative size of the priors assigned to each class can be used to “adjust” the importance of misclassifications for each class.

2.3.2 Unequal cost of Misclassification

In CART, the rules of class assignment can be modified from simple plurality to account for the costs of making a mistake in classification. The desire for more accurate classification for some classes than others for reasons unrelated to relative class sizes lead us to ‘misclassification cost’. In this example, clinicians’ must predict whether a patient with type I diabetes will have a CVD complication within 9 years, based on the information available. To this end, we first classify a node t to either class 0 (non-CVD) or 1 (CVD), and we predict the outcome of an individual based on the membership of the node in which the individual belongs. Unfortunately, we often make mistakes in such a classification, because some normal subjects with no risk of CVD will be predicted as CVD and vice versa. In any case, to weight these mistakes, we need to assign misclassification costs. In other word, a misclassification cost is simply a number that is assigned as a penalty for making a mistake.

Consider the root node in Figure 2.4. In this root node, there are 25 CVD and 111 non-CVD patients. If we assign class 1 for the root node, 111 normal subjects are misclassified. In this case, we would wrongly predict subjects without a risk of CVD as having a risk, and many false positive errors occur. On the other hand, we misclassify the 25 patients with CVD if the root node is assigned class 0. These are false negative errors.

Sometimes a mistake could be fatal, such as a false negative diagnosis of potential heart failure, because the diabetic patient may not get needed care. In most applications, false negative errors are more serious than false positive errors. So, regardless of their relative frequency, a patient who will have a CVD complication within 9 years might need to be more accurately predicted than a patient who will not have a CVD complication within 9 years. And therefore, higher misclassification costs could be specified for misclassifying a CVD as non-CVD than for classifying a non-CVD as CVD. However, no matter which error is made, the two kinds of mistake must be weighted.

		Observed	
		Class <i>CVD</i> ($Y=1$)	Class <i>Non-CVD</i> ($Y=0$)
Predicted	<i>CVD</i> ($Y=1$)		False Positives
	<i>Non-CVD</i> ($Y=0$)	False Negatives	

Table 3 Two-class classification performance

Let $C(i|j)$ be a unit misclassification cost that a class j subject is classified as a class i subject and satisfies

- $C(i|j) \geq 0, i \neq j$
- $C(i|j) = 0, i = j$.

Since i and j take only the values 0 or 1, without loss of generality let $C(1|0) = 1$. In other words, one false positive error counts as one. The clinicians and the statisticians need to work together to gauge the relative cost of $C(0|1)$. This is a subjective and difficult, but important decision.

As mentioned earlier, in most situations false negative errors are more serious than false positive errors, we assume that $C(0|1) \geq C(1|0)$. To illustrate, let us consider what happens if $C(0|1) = 4$.

In this case, it means that the effect of having false negatives cost four times that of false positives so that we will tolerate many more false positive errors than false negative ones, for a fixed classifier design. We know that the cost is 111 if the root node is assigned class 1. It becomes $25 \times 4 = 100$ if the root node is assigned class 0. Therefore, the root node should be assigned class 0 since $100 < 111$ (the cost of false negative errors < the cost of false positive errors). In other words, the class membership of 0 or 1 for a node depends on whether or not the cost of the false negative errors is lower than that of the false positive errors and vice versa. Classes are assigned to minimize the misclassification cost (error rate) for the cases in the node.

Putting the two criteria together, the function used to assign predicted classes to each node becomes:

Node t is assigned class 0 if

$$C(0|1)P(Y = 1|t) < C(1|0)P(Y = 0|t)$$

and to class 1 otherwise.

Note:

- $P(Y = 0|t) = N_0(t)/N(t)$ and $P(Y = 1|t) = N_1(t)/N(t)$;
- $N_0(t), N_1(t)$ refers to the number of patients in node t that are in class 0 and class 1, respectively.
- $N(t)$ refers to the number of patients in node t .

Assumed Class	Node Number						Cost of
	1	2	3	4	5	...	
1	111	94	17	93	1	...	<i>false positive errors</i>
0	100	40	60	16	24	...	<i>false negative errors</i>
Predicted class	0	0	1	0	1	...	

Table 4 Misclassification Costs when $C(0|1) = 4$

2.4 Incorporating Losses

From the previous section, the goodness of the split (or the change in impurity) is determined through the expression

$$\Delta i(t) = i(t_P) - (P(t_L)i(t_L) + P(t_R)i(t_R)),$$

Where $P(t_L)$ and $P(t_R)$ are the probabilities that a subject falls into nodes t_L and t_R , respectively.

$$P(t_L) = \frac{n_{1.}}{n_{1.} + n_{2.}}$$

and

$$P(t_R) = \frac{n_{2.}}{n_{1.} + n_{2.}}$$

When there are misclassification costs, an altered Gini criterion is used. The generalized Gini Index is defined using the following formula:

$$i(t) = \sum_{i \neq j} C(i|j)p(i|t)p(j|t),$$

where $C(i|j)$ is the cost of misclassifying j as a class i case.

This expression is used as the Gini measure of node impurity for variable misclassification costs. In the 2-class problem the formula reduces to

$$i(t) = (C(1|0) + C(0|1))p(0|t)p(1|t)$$

Let $P(t)$ be the posterior probability that a subject falls into node t .

$$\begin{aligned}
 P(t) &= \frac{N_0(t)}{N} + \frac{N_1(t)}{N} \\
 &= \frac{N_0}{N} \frac{N_0(t)}{N_0} + \frac{N_1}{N} \frac{N_1(t)}{N_1} \\
 &= P(Y = 0) \frac{N_0(t)}{N_0} + P(Y = 1) \frac{N_1(t)}{N_1}
 \end{aligned}$$

where $P(Y = 0)$ and $P(Y = 1)$ are prior probabilities of classes 0 and 1, respectively, assumed to be equal to the observed proportions.

When $C(0|1) : C(1|0) = 4 : 1$, every misclassified case in class 1 counts four times as much as a misclassified case in class 0. This situation is similar to one where the prior probability ($P^*(Y = 0)$ and $P^*(Y = 1)$) are

$$\begin{aligned}
 P^*(Y = 0) &= \frac{P(Y = 0)C(1|0)}{P(Y = 1)C(0|1) + P(Y = 0)C(1|0)} \\
 &= \frac{\frac{N_0}{N}C(1|0)}{\frac{N_1}{N}C(0|1) + \frac{N_0}{N}C(1|0)} \\
 &= \frac{N_0C(1|0)}{N_1C(0|1) + N_0C(1|0)}
 \end{aligned}$$

Similarly for

$$P^*(Y = 1) = \frac{N_1C(0|1)}{N_0C(1|0) + N_1C(0|1)}$$

Therefore, under the loss function $C(j|i)$, the modified posterior probability that a subject falls into node t is:

$$\begin{aligned}
 P^*(t) &= P^*(Y = 0) \frac{N_0(t)}{N_0} + P^*(Y = 1) \frac{N_1(t)}{N_1} \\
 &= \frac{N_0C(1|0)}{N_1C(0|1) + N_0C(1|0)} \frac{N_0(t)}{N_0} + \frac{N_1C(0|1)}{N_0C(1|0) + N_1C(0|1)} \frac{N_1(t)}{N_1} \\
 &= \frac{C(1|0)N_0(t) + C(0|1)N_1(t)}{N_1C(0|1) + N_0C(1|0)}
 \end{aligned}$$

Given $C(0|1) : C(1|0) = 4 : 1$, the altered form of $P(t_L)$ and $P(t_R)$ then becomes

$$P^*(t_L) = \frac{n_{11} + 4n_{12}}{n_{.1} + 4n_{.2}}$$

and

$$P^*(t_R) = \frac{n_{21} + 4n_{22}}{n_{.1} + 4n_{.2}}$$

Note: refer to Table 1 on section 2.2 for notation.

Chapter 3

Determining when to stop splitting

The recursive partitioning process may proceed until the tree is saturated in the sense that the offspring nodes subject to further division cannot be split. This happens, for instance, when there is only one subject in a node. Note that the total number of allowable splits for a node drops as we move from one layer to the next. As a result, the number of allowable splits eventually reduces to zero, and the tree cannot be split any further. Any node that we cannot or will not split is a terminal node.

The saturated tree is usually too large to be useful, because the terminal nodes are so small that we cannot make sensible statistical inference; and this level of detail is rarely scientifically interpretable. It is typically unnecessary to wait until the tree is saturated. Instead, a minimum for the number of observations in a node is set. We stop splitting when the number of observations in a node is smaller than the minimum. The choice of minimum often depends on the sample size (e.g., one percent).

Breiman et al. (1984) argued that depending on the stopping threshold, the partitioning tends to be too soon or too late. Accordingly, they made a fundamental shift by introducing a second step, called pruning. Instead of attempting to stop the partitioning, they propose to let the partitioning continue until it is saturated. Beginning with this maximum grown tree, we prune from the bottom up to an optimum size. The point is to find a subtree of the saturated tree that is most predictive of the outcome.

3.1 Resubstitution Cost

The quality of a tree is merely the quality of its terminal nodes. Thus,

$$r(t) = C(i|j)P(Y = j|t), \quad \text{for node } t \text{ classified as class } i$$

$r(t)$ is defined as the resubstitution estimate of the expected misclassification cost within node t . This cost is usually referred to as the *within-node misclassification cost*. Thus $r(t)$ measures a certain quality of node t .

Multiplying $r(t)$ by $P(t)$, we have $R(t)$, where $R(t)$ denotes the resubstitution estimate of the misclassification cost for node t , weighted by the proportion of subjects in node t .

$$R(t) = P(t)r(t)$$

Node Number	Node Class	Weight	Within node cost	Cost
t	i	$P(t) = N(t)/N$	$r(t)$	$R(t) = P(t) \times r(t)$
1	0	136/136	100/136	100/136
2	0	104/136	40/104	40/136
3	1	32/136	17/32	17/136
4	0	97/136	16/97	16/136
5	1	7/136	1/7	1/136
\vdots	\vdots	\vdots	\vdots	\vdots

Table 5 Resubstitution estimates of Misclassification Costs (Unit cost: $C(0|1) = 4$)

When the class memberships and misclassification costs are determined for all nodes, the misclassification cost for a tree can be computed easily by summing all costs in the terminal nodes.

For a tree T we define

$$\begin{aligned} R(T) &= \sum_{t \in \tilde{T}} P(t)r(t) \\ &= \sum_{t \in \tilde{T}} R(t) \end{aligned}$$

where \tilde{T} is the set of terminal nodes of tree T and $r(t)$ is the within-node misclassification cost of node t .

This $R(T)$ measures the performance of the maximal tree on the original learning dataset. Precisely, the quality of a tree, denoted by T , is reflected by the quality of its terminal nodes. The largest tree will always yield the lowest resubstitution cost. This corresponds to the fact that the maximal tree will always give the

best fit to the learning dataset. In practice, it is a bad idea to let the tree keep growing until all terminal nodes are homogenous, because it tends to make the tree very large and thus hard to interpret, and can reduce its accuracy as a classifier.

Chapter 4

Tree Pruning and Optimal Tree Selection

The purpose of pruning is to find the right-sized tree, which should be a sub-tree of the saturated tree. Regardless of whether the correct tree size is found by stopping early or by post-pruning, a key question is what criterion is to be used to determine the ‘best’ final tree size.

The maximal tree will always fit the learning dataset with higher accuracy than any other tree. The performance of the maximal tree on the original learning dataset, termed the “*resubstitution cost*”, generally greatly overestimates the performance of the tree on a new (future) dataset. This is because the deeper nodes are fitting noise in the training dataset (which is unlikely to occur with the same pattern in a different set of data) rather than the real signals that discriminate the classes.

The goal in selecting the optimal tree, defined with respect to expected performance on an independent set of data, is to find the correct so-called “*complexity parameter*” (details given in section 4.1), so that the information in the learning data set is fit but not over-fit.

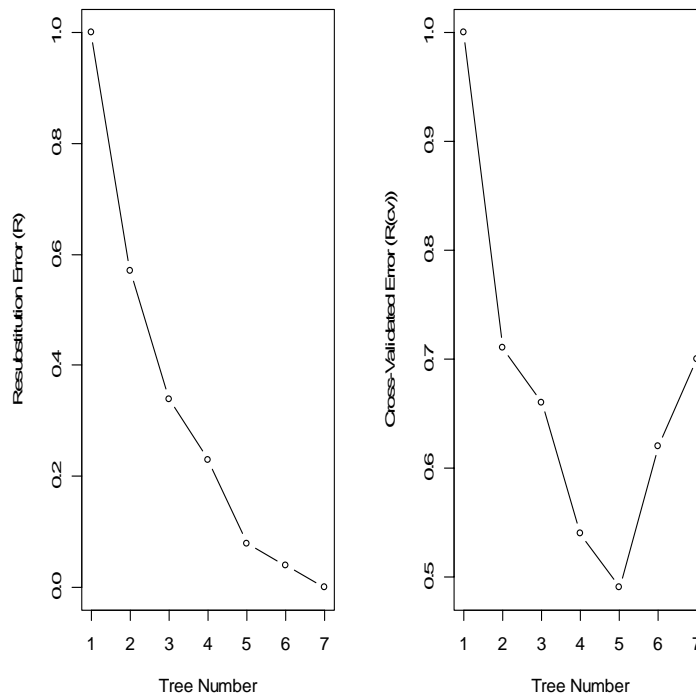


Figure 4.1: Plot of the resubstitution error rate (*a*-left) and the cross-validated error rate (*b*-right) for trees produced on diabetes patient dataset.

The figure above shows the relationship between tree complexity, reflected by the number of terminal nodes, and the resubstitution cost (or resubstitution error) for an independent dataset (b) and for the original learning dataset (a). As the number of nodes increases the resubstitution cost decreases monotonically for the learning data (a). This corresponds to the fact that the maximal tree will always give the best fit to the learning dataset. In contrast, the resubstitution cost for an independent dataset reaches a minimum, and then increases as the tree complexity increases. This reflects the fact that an overfitted and overly complex tree will not perform well on a new set of data. Thus, the purpose of pruning is to select the best and simpler tree T^* from a sequence of trees, such that $R(T)$ is minimized for an independent data set.

4.1 Tree Cost Complexity

By now, a large tree and/or complex tree has been constructed. Our next task is to decide how much of that tree to retain. Recall, the ultimate objective of tree pruning is to select a subtree of the saturated tree so that the misclassification cost (error rate) of the selected subtree is the lowest on an independent, identically distributed sample, called a *test sample*. The resubstitution error rate needed to perform the cost-complexity pruning is computed as the tree is being grown. A penalizing cost, the so-called *complexity parameter*, is assigned to a unit increase in complexity, i.e., one extra terminal node. The sum of all costs becomes the penalty for the tree complexity, and the *cost-complexity of a tree* is:

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

where $\alpha(> 0)$ is the complexity parameter. Note that tree size (the complexity of T) is taken to be the number of the terminal nodes, denoted by $|\tilde{T}|$, because for binary trees the tree size starts at one (the root node) and increases by one with each added split.

The difference between $R_\alpha(T)$ and $R(T)$ as a measure of tree quality resides in that $R_\alpha(T)$ penalizes a big tree. In RPART, the tree is grown to its maximum size, then pruned back to minimize the complexity. As the complexity parameter is increased, larger trees are penalized for their complexity more and more, until a discrete threshold is reached at which a smaller sized tree's higher cost are outweighed by the larger tree's higher complexity.

For any tree with over, say 20 nodes, many subtrees are possible. And the combinatorics involved are usually complicated. The use of tree cost-complexity allows us to construct a sequence of trees which are nested, because successively pruned trees contain all the nodes of the next smaller tree in the sequence. Initially, many nodes are often pruned going from one tree to the next smaller tree in the sequence, but fewer nodes tend to be pruned as the root node is approached. Based on the sequence of nested essential subtrees from any given tree T , we can examine the properties of these subtrees and make a selection from them. The resubstitution approach plays a useful role in evaluating cost complexity.

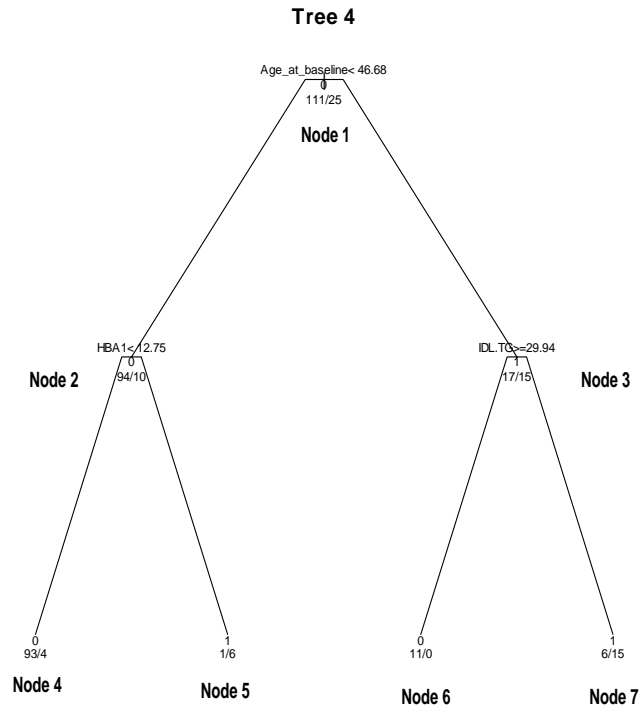


Figure 4.2: 7 nodes tree with 3 splits and 4 terminal nodes.

Consider Tree 4, in Figure 4.2. Its cost-complexity is

$$\begin{aligned}
 R_\alpha(T) &= (R(t=4) + R(t=5) + R(t=6) + R(t=7)) + \alpha|\tilde{T}| \\
 &= \left(\frac{16}{136} + \frac{1}{136} + 0 + \frac{6}{136}\right) + 4\alpha \\
 &= (0.1176 + 0.0074 + 0 + 0.0441) + 4\alpha \\
 &= 0.1691 + 4\alpha
 \end{aligned}$$

That is, the cost for Tree 4 is 0.1691 and its complexity is 4. Thus, its cost-complexity is $0.1691 + 4\alpha$

for a given complexity parameter α . The question is, is there a subtree of Tree 4 that has a smaller cost-complexity?

According to Breiman, “For any value of the complexity parameter, α , there is a unique smallest subtree of an initial constructed large tree that minimizes the cost-complexity.”

This theorem ensures that we cannot have two subtrees of a given size with the same cost-complexity. We call this smallest subtree the optimal subtree with respect to the complexity parameter. And an optimal subtree is optimal for an interval range of the complexity parameter. Thus, we need to find the limits of these intervals or the thresholds of α to make use of the corresponding optimal subtree.

Each internal node in Tree 4 has two offspring terminal nodes. The question is, what happens if an internal node becomes a terminal node? In other words, what is the consequence of pruning off all off spring nodes of an internal node? For instance, if we cut the offspring nodes off the root node, we have the root-node tree whose cost-complexity is $0.7353 + \alpha$. For it to have the same cost-complexity as the initial seven-node tree, we need $0.1691 + 4\alpha = 0.7353 + \alpha$, giving $\alpha = 0.1887$. We can also find out the consequence of changing node 2 to a terminal node. Then an initial seven-node tree is compared with the five-node subtree, consisting of nodes 1 to 3, 6 and 7. For the new subtree to have the same cost-complexity as the initial tree, we find $\alpha = 0.16912$.

Node(t)	R(t)	R(T)	T		α
7	6/136	6/136			
6	0	0			
5	1/136	1/136			
4	16/136	16/136			
3	17/136	6/136	2	$0.0441 + 2\alpha = 0.125 + \alpha$	0.081 (α_3)
2	40/136	17/136	2	$0.125 + 2\alpha = 0.2941 + \alpha$	0.1691
1	100/136	23/136	4	$0.1691 + 4\alpha = 0.7353 + \alpha$	0.1887

The first threshold parameter, say $\alpha_3 = 0.081$ in the above table, is the smallest α over two internal nodes. Note “ α_3 ” refers to the complexity parameter computed from Tree 4 in Figure 4.2.

Using α_3 we change an internal node t to a terminal node when

$$R(t) + \alpha_3 \leq R(T) + \alpha_3|\tilde{T}|$$

until this is no longer possible.

This pruning process results in the optimal subtree corresponding to α_3 . The optimal tree with respect to α_3 is shown in Figure 4.3, where internal node 3 becomes a terminal node.

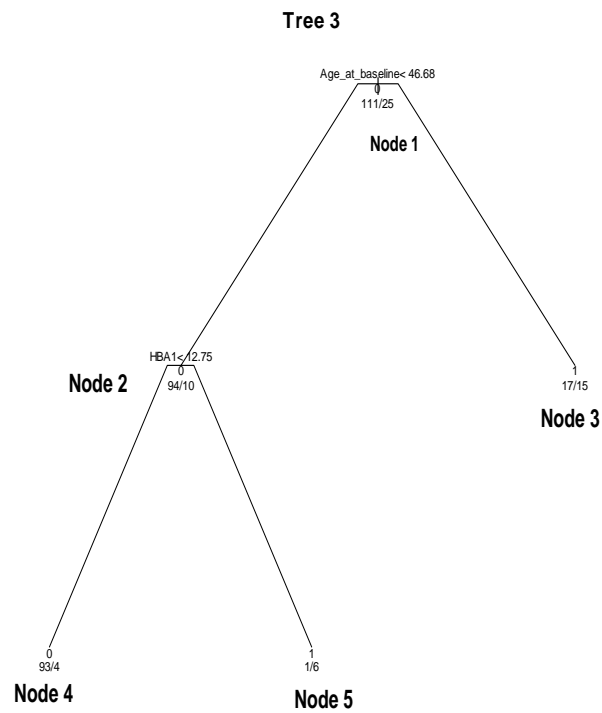


Figure 4.3: 5 nodes tree with 2 splits and 3 terminal nodes.

After pruning the tree using the first threshold, we seek the second threshold, say α_2 , (where “ α_2 ” refers to the complexity parameter computed from Tree 3) in the same way as the first one except that the once pruned subtree in Figure 4.3 plays the role of an initial tree.

Based on our learning dataset we end up with 6 thresholds,

$$0 < 0.0074(\alpha_{\text{SaturatedTree}}) < 0.0294(\alpha_5) < 0.05515(\alpha_4) < 0.081(\alpha_3) < 0.1691(\alpha_2) < 0.3162(\alpha_1)$$

(refer to Appendix B for calculation details).

Also, let the corresponding optimal subtrees be

$$\text{Saturated Tree} \supset T_{\alpha_{\text{saturatedtree}}}(\text{Tree 6}) \supset T_{\alpha_5}(\text{Tree 5}) \supset T_{\alpha_4}(\text{Tree 4}) \supset \dots \supset T_{\alpha_1}(\text{Tree 1}),$$

Where $T_{\alpha_2} \supset T_{\alpha_1}$ means that T_{α_1} is a subtree of T_{α_2} . In particular, T_{α_1} is the root-node subtree. These are so-called *nested optimal subtrees*. The final subtree will be selected from among them.

	CP	nsplit	rel error	xerror	xstd
1	4.3e-01	0	1.00	1.00	0.1806850
2	2.3e-01	1	0.57	0.71	0.1333918
3	1.1e-01	2	0.34	0.66	0.1288296
4	7.5e-02	3	0.23	0.54	0.1185575
5	4.0e-02	5	0.08	0.49	0.1128475
6	1.0e-02	6	0.04	0.62	0.1287382
7	1.0e-06	10	0.00	0.70	0.1421164

Table 6 CP table produced by RPART based on diabetes patient dataset

Note: for an easier reading, the ‘rel error’ column in RPART summary (referring to the resubstitution error rate) have been scaled so that the first node has an error of 1. Since in this example, the tree with no splits must make 100/136 misclassifications, multiply this column by 136/100 to get a result in terms of relative error scale. The complexity parameter (CP) column has been similarly scaled.

4.2 k-fold Cross validation

As mentioned above, the goal in selecting the optimal tree is to find the correct complexity parameter α so that the information in the learning data set is fit but not overfit. Beginning with a learning sample, a large tree is constructed by recursively splitting the nodes. From this large tree, a sequence of complexity parameters $\alpha_m, \alpha_{m-1}, \dots, \alpha_2, \alpha_1$ and their corresponding optimal subtrees $T_{\alpha_m}, T_{\alpha_{m-1}}, \dots, T_{\alpha_2}, T_{\alpha_1}$ are obtained. In general, finding this value of α would require a new or independent set of data, a so-called *test sample*. When a test sample is available, estimating $R(T)$ for any subtree T is straightforward, because we only need to apply the subtree to the test sample. However, in practice, we rarely have a test sample. Breiman proposed to use a technique of cross validation based on cost-complexity.

k-fold cross validation makes use of the existing data to simulate independent test data. First, all of the data are used and the tree is allowed to grow as large as possible. This is the reference, unpruned tree (saturated

tree). This will give the best performance with the learning data - a tree with 10 splits and 21 nodes in the figure below.

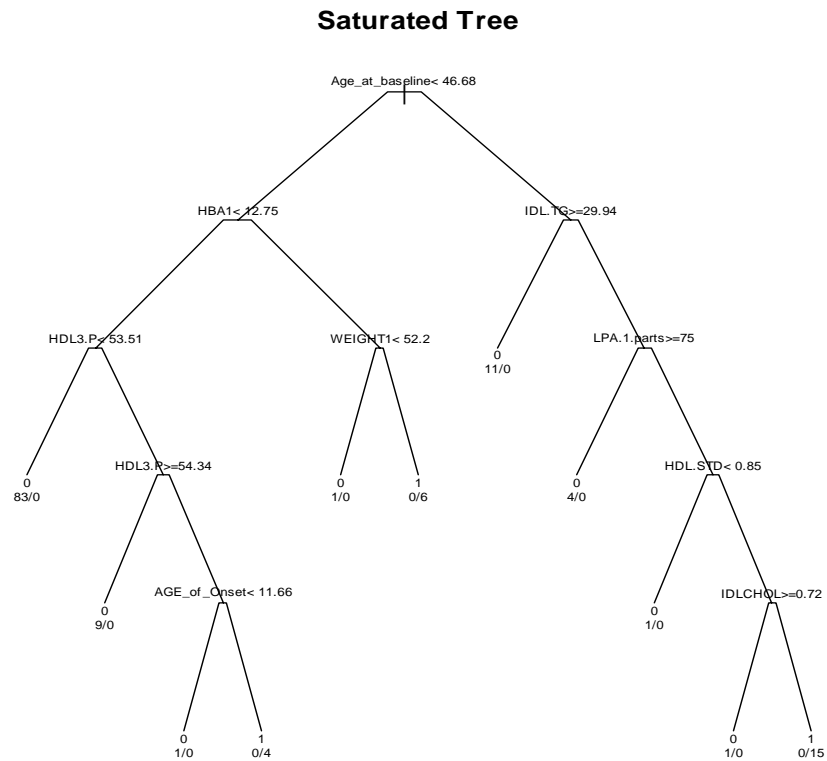


Figure 4.4: 21 nodes tree with 10 splits and 11 terminal nodes .

The learning dataset is partitioned into k subsets, often 5, 10, or 25 corresponding to 5-, 10-, or 25- fold cross-validation, respectively. Usually, an attempt is made to keep the same class frequencies in each fold and each fold is stratified by the outcome variable of interest. This ensures that a similar distribution of outcomes is present in each of the k subsets of data.

One of these subsets of data is reserved for use as an independent test dataset, while the other $k-1$ subsets are combined for use as the learning dataset in the model-building procedure (see the figure below).

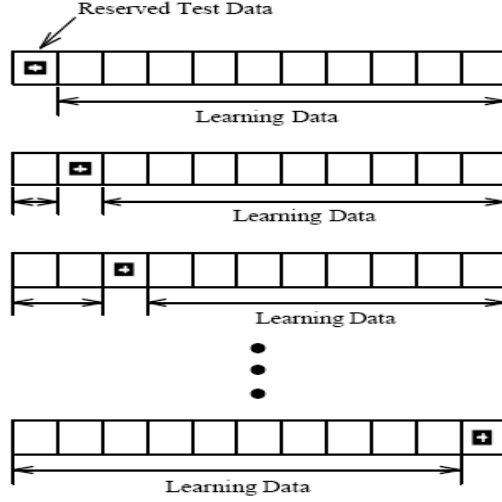


Figure 4.5: Illustration of 11- fold cross validation.

Let us consider a 10-fold cross validation. Each of the 10 folds will be used as an ‘*independent*’ test set while the other nine form the learning set. This means that there will be 10 different models i.e. 10 separate learning and test data sets produced (although the learning sets will always share a large number of the same cases), each one of which can be tested against an independent subset of the data.

Let $\mathcal{L}_{(-i)}$ be the sample set including all but those subjects in group i , $i = 1, 2, \dots, 10$.

Using the patients in $\mathcal{L}_{(-1)}$, where

$$\mathcal{L}_{(-1)} = \text{total number of patients in the dataset} - \text{number of patients in group 1}$$

another large tree, say $T_{(-1)}$ can be produced, in the same way as we did using all subjects.

Taking each of α_k ($\alpha_m, \alpha_{m-1}, \dots, \alpha_2, \alpha_1$) from the sequence of complexity parameters as has been derived previously, obtain the optimal subtree, $T_{(-1),k}$ of $T_{(-1)}$ corresponding to α_k . Then, we would have a sequence of the optimal subtrees of $T_{(-1)}$, that is, $T_{(-1),m}, T_{(-1),m-1}, \dots, T_{(-1),2}, T_{(-1),1}$.

Using group 1 as a test sample relative to $\mathcal{L}_{(-1)}$, we have $R^{ts}(T_{(-1),k})$, of $R(T_{(-1),k})$. Likewise, using $\mathcal{L}_{(-i)}$ as the learning sample and the data in group i as the test sample, we obtain $R^{ts}(T_{(-i),k})$.

Thus, the final cross-validation estimate, $R^{cv}(T_{\alpha_k})$ of $R(T_{\alpha_k})$ follows from averaging $R^{ts}(T_{(-i),k})$ over $i = 1, 2, \dots, 10$

$$R^{cv}(T_{\alpha_k}) = \frac{R^{ts}(T_{(-1),k}) + R^{ts}(T_{(-2),k}) + \dots + R^{ts}(T_{(-10),k})}{10}$$

where $k = m, m - 1, \dots, 2, 1$

This means that 10 separate trees will generate an error rate for each of the 10 test data sets. These 10 error rates are averaged to produce an error rate as a function of tree size. This information can then be used to prune the reference tree to the number of nodes that produces the minimum error from the cross-validated trees. Pruning starts at the terminal nodes and proceeds in a stepwise fashion, sequentially removing the least important nodes until the desired size is reached. Using this patient dataset, a tree with 5 splits and 6 terminal nodes gives the lowest error with the test data (refer to Table 6), this tree is illustrated in Figure 4.7, Tree 5.

The cross-validated trees are used as a guide to the optimum tree size. The end product of this process is a pruned reference tree which should produce the optimum performance with new data. Therefore, by using the cross-validation method, a minimum cost occurs when the tree is complex enough to fit the information in the learning dataset, but not so complex as to fit the “noise” in the data.

4.3 Standard Error of R^{cv*}

The subtree corresponding to the smallest $R^{cv}(T_{\alpha_k})$ is obviously desirable. There are some issues that arise when dealing with the cross-validation estimate (CV cost). The cross-validation estimates generally have substantial variabilities due to the random number generator used to separate the dataset into k test sets.

And oftentimes, when choosing the “right-sized” tree with the minimum CV cost, there will be several trees with CV costs close to the minimum. Breiman et.al make a reasonable suggestion that one should choose the “right-sized” tree with the smallest sized (least complex) tree whose CV cost does not differ appreciably from the minimum CV cost. Breiman proposed a revised strategy to select the final tree, which takes into account the standard errors of the cross validation estimates, the so called “*1 SE rule*”.

For making a selection of trees with CV costs close to the minimum, Breiman suggests to choose as the “right-sized” tree the smallest-sized tree whose CV cost does not exceed the minimum CV cost by more than one standard error of the CV cost for the minimum CV cost tree.

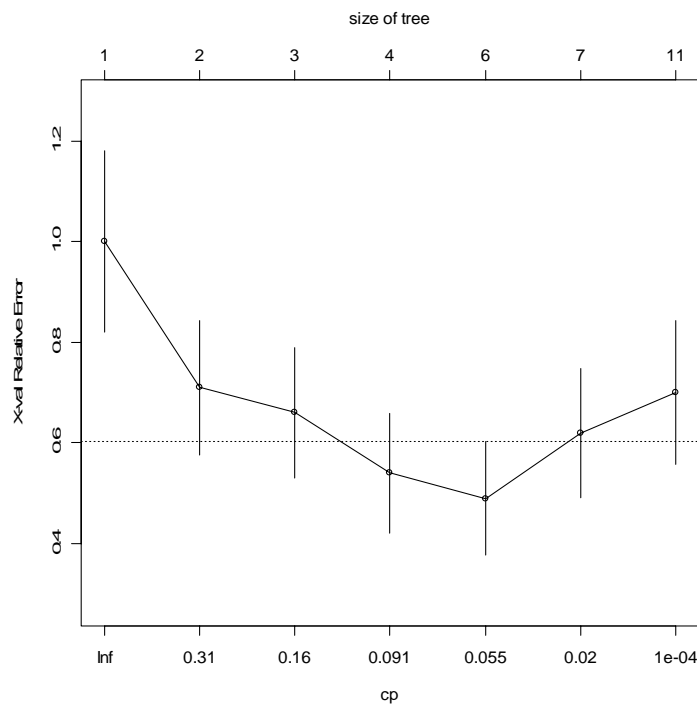


Figure 4.6: Graphical representation of complexity parameter.

The graph above illustrates the relation between complexity parameter and the tree complexity (the size of tree refers to the number of the terminal nodes) to the cross validation estimate (x-val Relative Error). Vertical lines (error bar) represent the standard error around the R^{cv} for the different tree complexities. The horizontal line refers to the minimum of CV cost plus one standard error. As shown in the graph above, the CV costs, approach the minimum as tree size initially increases, and start to rise as tree size becomes very large. Note that the selected “right-sized” tree is close to the inflection point in the curve, that is, close to the point where the initial drop on CV costs with increased tree size starts to level out.

From this graph (Figure 4.6), we can see that a tree with 5 splits and 6 terminal nodes (Tree 5) gives the minimum CV estimate of the misclassification cost with the test data. However, a simpler tree (that is, a

tree with 3 splits and 4 terminal nodes (Tree 4) is within one standard error of the tree with the minimum test error (refer to the error bar). In other words, this simpler tree is not significantly worse than Tree 5. Both trees will perform equally well. Therefore, taking account of the standard errors of the cross validation estimate, Tree 4 is chosen because it is less complex and its accuracy is comparable to the tree with minimum cross-validation estimates (i.e. Tree 5).

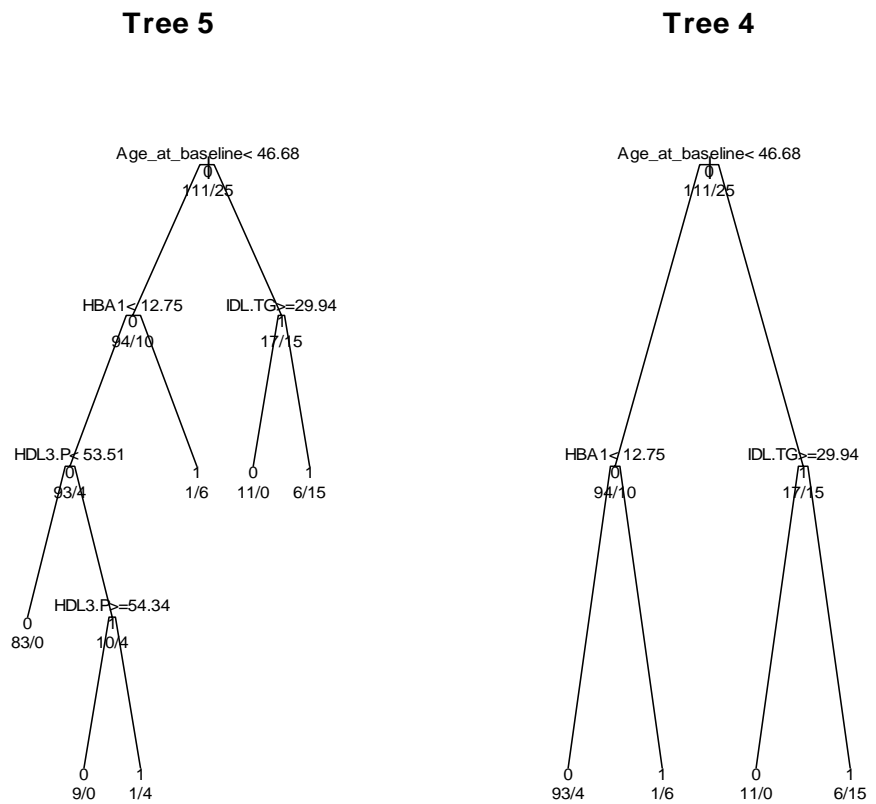


Figure 4.7: 11 nodes tree with 5 splits and 6 terminal nodes vs 7 nodes tree with 3 splits and 4 terminal nodes.

Chapter 5

Missing Variables

For each node, the “*primary splitter*” is the variable that best splits the node, maximizing the purity of the resulting child nodes. When the primary splitting variable is missing for an individual observation, that observation is not discarded but, instead, a surrogate splitting variable is sought. The RPART algorithm calculates to what extent the alternative splits resemble the best split in terms of the number of cases that they send to each class in the same way. Any observation with a missing value for the best split is then classified using the first (most resembling) surrogate split, or if that value is missing also, the second surrogate split, and so on.

For node 2 (refer to RPART summary output in Appendix A), the primary split variable is missing on six subjects. Since, HbA1 is the splitting variable for that node, we have a problem in deciding to which child node we send these six subjects. The surrogate splits attempt to utilize the information in other predictors to assist us in making such a decision. For example, an obvious choice for replacing HbA1 is the second-best splitting variable, a so-called *competitive split*. The problem is that a competitive split does not necessarily offer a good replacement for HbA1 when HbA1 is missing. Therefore, it is a good idea to look for a predictor that most resembles HbA1 in classifying the subjects.

One measure of resemblance between two splits suggested by Breiman et al. (1984), is the coincidence probability that the two splits send a subject to the same node.

	HbA1 < 12.75 (Non-CVD)	HbA1 > 12.75 (CVD)	NA	
HbA1c < 10.245 (Non-CVD)	91	0	1	92
HbA1c > 10.245 (CVD)	1	6	0	7
	92	6	1	99

At node 2, HbA1 is missing for 6 observations. 98.97 % (97/98) of the observations that have both HbA1c and HbA1 agree at their respective splits, hence HbA1c is chosen as the best surrogate for HbA1.

Of the six subjects with missing HbA1, five of them were also missing the value of HbA1c. Therefore, we have to look for the second best surrogate split.

	HbA1 < 12.75 (Non-CVD)	HbA1 > 12.75 (CVD)	NA	
BMI < 19.85 (CVD)	1	2	1	4
BMI > 19.85 (Non-CVD)	91(+1)	4	4	100
	92(+1)	6	5	104

94.90 % (94/99) of the observations that have both BMI and HbA1 agree at their respective splits. Hence BMI is chosen as the second best surrogate.

One subject is split based on the first surrogate, HbA1c, and the remaining five subjects are split based on the second best surrogate, BMI.

	HbA1	HbA1c	BMI	Total
Non-CVD	92	1	4	97
CVD	6		1	7

From the RPART summary output, we can see that 97 subjects are sent to node 4 and 7 subjects to node 5.

Chapter 6

Disadvantage of CART

Insignificant modification of the learning sample, such as eliminating several observations, could lead to radical changes in the decision tree. As well, an increase or decrease in the ratio of misclassification costs may change the splitting variables.

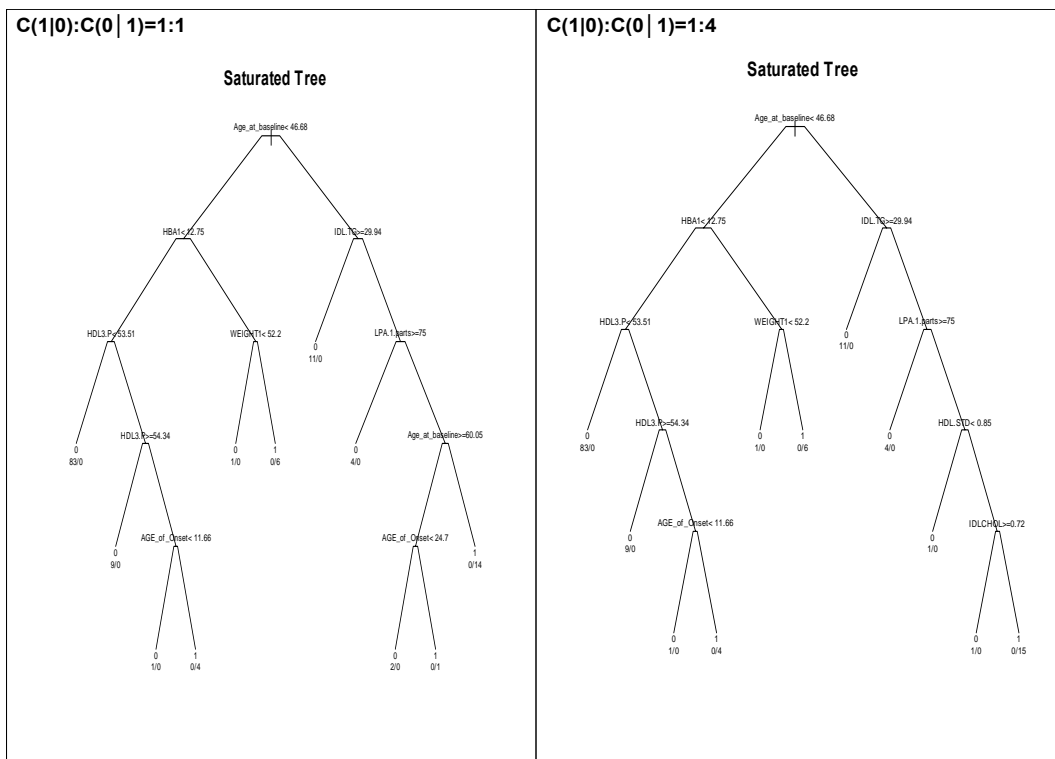


Figure 6.1: Classification Tree using different unit of misclassification cost.

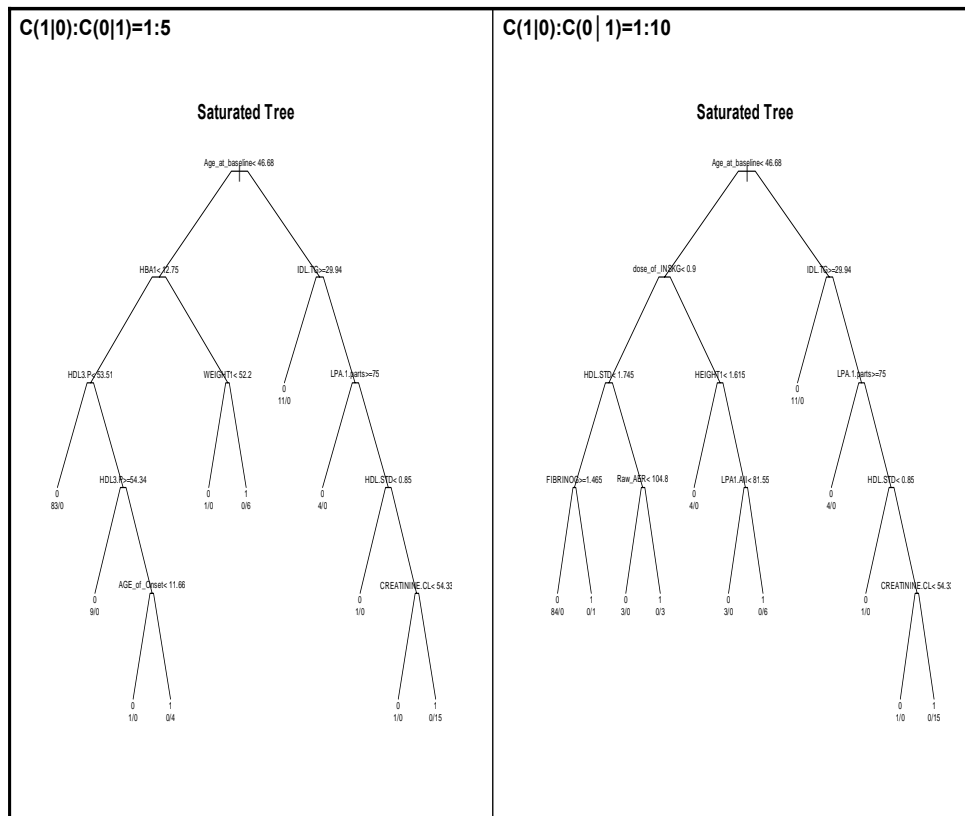


Figure 6.2: Classification Tree using different unit of misclassification cost.

Chapter 7

Advantage of CART

There are other statistical methods that are commonly used to discriminate among subjects who are at risk for a certain condition based on the measured variables, such as discriminant analysis and logistic regression.

These parametric regression methods, however, may not lead to faithful data descriptions when the underlying assumptions are not satisfied. Sometimes, model interpretation can be problematic in the presence of higher-order interactions among potential predictors.

So, why should we use CART?

Many clinical variables are not normally distributed and different groups of patients may have markedly different degrees of variation or variance. Decision trees are non-parametric and therefore do not require Normality assumptions of the data.

Complex interactions or patterns may exist in the data. For example, the value of one variable (e.g. Age) may substantially affect the importance of another variable (e.g., sex). These types of interactions are generally difficult to model and virtually impossible to model when the number of interactions and variables becomes substantial.

The Figure below displays a diagram of a decision tree that highlights an interaction between age and sex. If a decision tree were being used as an exploratory tool, this model would suggest incorporating an interaction term between sex and age in a model.

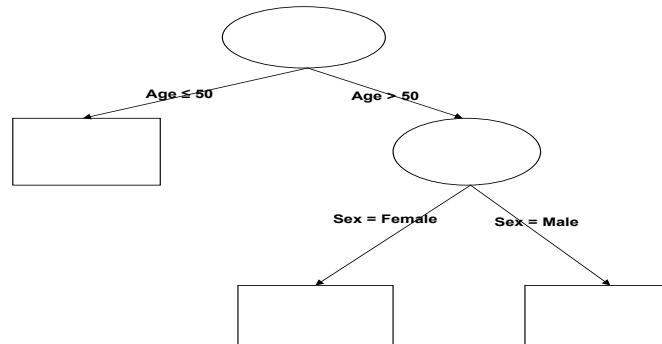


Figure 7.1: Illustration of Interaction .

Decision trees can handle data of different types: continuous, categorical, ordinal, binary. Transformations of the data are not required.

Decision trees cope with missing data by identifying surrogate splits in the modeling process.

When we are dealing with many possible predictor variables, it makes the task of variable selection difficult. Decision trees do not require variables to be selected in advance. The CART algorithm identifies the most significant variables and eliminates non-significant ones.

Decision trees have been recognized as a useful modeling tool among non-statisticians, particularly in the medical area, as they produce a model that is very easy to interpret.

Chapter 8

Concluding Remarks

Based on the classification tree analysis, we can conclude that type I diabetes patients aged over 46.68 years and with IDL%TG level less than 29.94 are most at risk of cardiovascular complication over the next nine years. As well, patients younger than 46.68 and with HbA1 greater than 12.75 are at risk of developing a cardiovascular complication.

Since HbA1c is a specific subtype of HbA1 (that is 90% of HbA1 is made up of HbA1c), it is evident from the result that those patients who are younger than 46.68 with HbA1c level greater than 10.245 are also at risk of developing CVD complication.

From the saturated tree, we can see that some of the lipid variables (IDL.TG, HDL3.P, LpAI (commonly found in HDL2), HDL.STD, IDLChol) are important markers for developing cardiovascular disease. In addition to these lipid variables, there might be an association between overweight/obesity with cardiovascular disease (based on the two variables body mass Index (BMI) and weight).

The present study included only a limited number of diabetic patients. Thus, more studies are needed to confirm the results. However, incorporation of these risk factors into the current medical standards for commercial pilots merits serious consideration.

Notation

N	number of subjects in learning sample
N_i	number of subjects in class i
$N_i(t)$	number of class i subjects in the learning sample which are in node t
\mathcal{L}	learning sample
$\mathcal{L}(-i)$	the sample set including all but those subjects in group $i, i = 1, 2, \dots, n$
t	index of node
t_P	parent node
t_L	left child node
t_R	right child node
$P(t)$	probability that a subject falls into node t
$P(t_L)$	probability that a subject falls into nodes t_L
$P(t_R)$	probability that a subject falls into nodes t_R
$i(t)$	impurity function
$i(t_P)$	impurity value for parent node
$i(t_L)$	impurity value for left child node
$i(t_R)$	impurity value for right child node
$\Delta i(t)$	change of impurity
$C(i j)$	cost of misclassifying a class j object as a class i object
T	(binary) tree
$r(t)$	within-node misclassification cost of node t
$R(t)$	resubstitution estimate of the misclassification cost for node t
$R(T)$	resubstitution estimate of the misclassification cost for a tree
\tilde{T}	set of terminal nodes of T
$ \tilde{T} $	number of terminal nodes in tree T
α	complexity parameter
$R_\alpha(T)$	cost-complexity measure : $R(T) + \alpha T $
α_k	a sequence of complexity parameters
T_{α_k}	k^{th} optimally prune subtree
$R^{ts}(T_{(-i),k})$	resubstitution cost based on group i as a test sample relative to $\mathcal{L}(-i)$ sample set with tree $T_{(-i),k}$ as the optimal subtree
$R^{cv}(T_{\alpha_k})$	final cross- validation estimate taken by averaging $\sum_{i=1}^n \frac{R^{ts}(T_{(-i),k})}{n}$
$SE(R^{cv})$	estimate of the standard error of R^{cv}

Appendix A

```
# Load the data

> diabetes<-read.csv(file="D://fannys/My Documents/Project/diabetes.csv")
> study1<-read.csv(file="D://fannys/My Documents/Project/study1.csv")
> fix(study)
> y<-rep(1:0,c(25,111))

# Note: diabetes file contains all the predictor variables,
whereas study1 file contains all predictor variables with the exclusion
of the categorical variables.

# Attach the library so that the function can be found
> require(rpart)
Loading required package: rpart
[1] TRUE

# We set the seed value to ensure that we all get the same results as sometimes
we may get different trees through the cross-validation process.
> set.seed(123456)

# Convert the categorical variable to a factor

> Macroalbuminuria.f<-factor(diabetes$Macroalbuminuria)
> Claas_of_AER.f<-factor(diabetes$Claas_of_AER)
> Gender.f<-factor(diabetes$Gender)
> overwieight.f<-factor(diabetes$overwieight)
> Antihypertenive_med.f<-factor(diabetes$Antihypertenive_med)
> ACE_Inhibitor.f<-factor(diabetes$ACE_Inhibitor)
> OTHER.AHT1.f<-factor(diabetes$OTHER.AHT1)
> LIPID.LOWering1.f<-factor(diabetes$ LIPID.LOWering1)
> Apirin1.f<-factor(diabetes$ Apirin1)
> NITRO1.f<-factor(diabetes$NITRO1)
> Any.MicroVD.f<-factor(diabetes$Any.MicroVD)
> RETINOp athy1.f<-factor(diabetes$RETINOp athy1)
> NEUROp athy1.f<-factor(diabetes$NEUROp athy1)
> SMOKING1.f<-factor(diabetes$SMOKING1)
> HISTSMOK.f<-factor(diabetes$HISTSMOK)
```

```

# The loss matrix, where C(1|0):C(0|1)=4:1
> loss <- matrix(c(0, 4, 1, 0), byrow=TRUE, ncol=2)

# Fit the model
# We set the complexity parameter(cp) and minsplit really small to produce
a really large tree.
# Minsplit indicates the minimum number of observations in a node for which
the routine will even try to compute the split.
# Cp: the threshold complexity parameter.
# The spltiing index is Gini, by default
# A value of usesurrogate = 2, the default. This is referring to Breimans
surrogate rule.
# 10-fold cross-validation is used by default.
> study.rp<-rpart(y~.+Macroalbuminuria.f+Claas_of_AER.f+Gender.f+overwieight.f+
Antihypertenive_med.f+ACE_Inhibitor.f+OTHER.AHT1.f+LIPID.LOWering1.f+Apirin1.f+
NITRO1.f+Any.MicroVD.f+RETINOp athy1.f+NEUROp athy1.f+SMOKING1.f+HISTSMOK.f,
method="class", data=studyl,control=rpart.control(minsplit=1,cp=0.000001),
parms=list(loss=loss))

# Print a summary which examines each node in depth
> summary(study.rp)
Call:
rpart(formula = y ~ . + Macroalbuminuria.f + Claas_of_AER.f +
  Gender.f + overwieight.f + Antihypertenive_med.f + ACE_Inhibitor.f +
  OTHER.AHT1.f + LIPID.LOWering1.f + Apirin1.f + NITRO1.f +
  Any.MicroVD.f + RETINOp athy1.f + NEUROp athy1.f + SMOKING1.f +
  HISTSMOK.f, data = studyl, method = "class", parms = list(loss = loss),
  control = rpart.control(minsplit = 1, cp = 1e-06))
n= 136

      CP nsplit rel error xerror      xstd
1 4.3e-01     0     1.00   1.00 0.1806850
2 2.3e-01     1     0.57   0.71 0.1333918
3 1.1e-01     2     0.34   0.66 0.1288296
4 7.5e-02     3     0.23   0.54 0.1185575
5 4.0e-02     5     0.08   0.49 0.1128475
6 1.0e-02     6     0.04   0.62 0.1287382
7 1.0e-06    10     0.00   0.70 0.1421164

Node number 1: 136 observations,      complexity param=0.43
predicted class=0 expected loss=0.7352941
  class counts:   111    25
probabilities: 0.816 0.184
left son=2 (104 obs) right son=3 (32 obs)
Primary splits:
  Age_at_baseline < 46.68 to the left, improve=14.567060, (0 missing)
  AGE_of_Onset    < 16.71 to the left, improve=10.594190, (0 missing)
  IDL.TG          < 28.83 to the right, improve=10.501360, (2 missing)
  Mean_arterial_BP < 97.055 to the left, improve=10.006300, (0 missing)

```

```

    FPG                < 17.7    to the left,  improve= 9.881804, (4 missing)
Surrogate splits:
  AGE_of_Onset        < 21.53    to the left,  agree=0.853, adj=0.375, (0 split)
  Duration_of_diabetes < 35.25    to the left,  agree=0.824, adj=0.250, (0 split)
  Mean_pulse.pressure1 < 63.835  to the left,  agree=0.824, adj=0.250, (0 split)
  Mean_systolic_BP1   < 158      to the left,  agree=0.801, adj=0.156, (0 split)
  TOTHDLCE            < 1.715    to the left,  agree=0.794, adj=0.125, (0 split)

Node number 2: 104 observations,    complexity param=0.23
  predicted class=0  expected loss=0.3846154
  class counts:      94    10
  probabilities: 0.904 0.096
  left son=4 (97 obs) right son=5 (7 obs)
Primary splits:
  HBA1                < 12.75    to the left,  improve=14.362060, (6 missing)
  HBA1C               < 10.245   to the left,  improve=13.352030, (5 missing)
  dose_of_INSKG       < 0.9      to the left,  improve=12.813880, (3 missing)
  HDL.P               < 46.88    to the left,  improve= 9.221756, (2 missing)
  LPA.1.parts         < 35.5     to the right, improve= 8.839540, (8 missing)
Surrogate splits:
  HBA1C               < 10.245   to the left,  agree=0.990, adj=0.833, (1 split)
  BMI1                < 19.85    to the right, agree=0.949, adj=0.167, (5 split)
  Mean_diastolic_BP1 < 99.5     to the left,  agree=0.949, adj=0.167, (0 split)
  HDL3TG              < 0.165    to the left,  agree=0.949, adj=0.167, (0 split)

Node number 3: 32 observations,    complexity param=0.11
  predicted class=1  expected loss=0.53125
  class counts:      17    15
  probabilities: 0.531 0.469
  left son=6 (11 obs) right son=7 (21 obs)
Primary splits:
  IDL.TG              < 29.94    to the right, improve=10.044930, (0 missing)
  LPA.1.parts         < 70.5     to the right, improve= 7.787574, (1 missing)
  HDL2P               < 87.4     to the right, improve= 6.987778, (0 missing)
  Gender.f            splits as LR,          improve= 5.372178, (0 missing)
  HDL2CHOL            < 1.28     to the right, improve= 5.093205, (0 missing)
Surrogate splits:
  Duration_of_diabetes < 20.3     to the left,  agree=0.781, adj=0.364, (0 split)
  NONHDL              < 2.58     to the left,  agree=0.781, adj=0.364, (0 split)
  TOTALCE              < 3.15     to the left,  agree=0.781, adj=0.364, (0 split)
  IDL.CE               < 17.515   to the left,  agree=0.781, adj=0.364, (0 split)
  AGE_of_Onset        < 34.385   to the right, agree=0.750, adj=0.273, (0 split)

Node number 4: 97 observations,    complexity param=0.075
  predicted class=0  expected loss=0.1649485
  class counts:      93     4
  probabilities: 0.959 0.041
  left son=8 (83 obs) right son=9 (14 obs)
Primary splits:
  HDL3.P              < 53.505   to the left,  improve=9.608472, (2 missing)
  HDL.P               < 46.88    to the left,  improve=6.016576, (2 missing)

```

```

LPA.1.parts < 35.5    to the right, improve=5.889436, (8 missing)
TOTHDLCE    < 0.86    to the right, improve=5.275414, (2 missing)
HDL2.PL     < 21.735  to the right, improve=5.275414, (2 missing)
Surrogate splits:
HDL.P       < 45.85   to the left,  agree=0.937, adj=0.571, (0 split)
TOTHDLTG   < 0.08    to the right, agree=0.874, adj=0.143, (2 split)
VLDL.CE    < 15.285  to the left,  agree=0.874, adj=0.143, (0 split)
HDL.TG     < 2.145   to the right, agree=0.874, adj=0.143, (0 split)
Raw_AER    < 966.775  to the left,  agree=0.863, adj=0.071, (0 split)

Node number 5: 7 observations,    complexity param=0.01
predicted class=1  expected loss=0.1428571
class counts:      1      6
probabilities: 0.143 0.857
left son=10 (1 obs) right son=11 (6 obs)
Primary splits:
WEIGHT1 < 52.2    to the left,  improve=1.237536, (0 missing)
HDL3CHOL < 0.675  to the left,  improve=1.237536, (0 missing)
LDLTG    < 0.155  to the left,  improve=1.237536, (0 missing)
LDLFC    < 0.725  to the left,  improve=1.237536, (0 missing)
HDL3CE   < 0.565  to the left,  improve=1.237536, (0 missing)

Node number 6: 11 observations
predicted class=0  expected loss=0
class counts:      11      0
probabilities: 1.000 0.000

Node number 7: 21 observations,    complexity param=0.04
predicted class=1  expected loss=0.2857143
class counts:      6      15
probabilities: 0.286 0.714
left son=14 (4 obs) right son=15 (17 obs)
Primary splits:
LPA.1.parts < 75    to the right, improve=4.496792, (1 missing)
HDL.STD     < 1.925  to the right, improve=3.362361, (4 missing)
Age_at_baseline < 61.045 to the right, improve=3.348310, (0 missing)
THDLCHOL    < 2.275  to the right, improve=3.348310, (0 missing)
HDL2CHOL    < 1.34    to the right, improve=3.348310, (0 missing)
Surrogate splits:
Raw_AER    < 7.23    to the left,  agree=0.95, adj=0.75, (1 split)
THDLCHOL   < 2.275  to the right, agree=0.95, adj=0.75, (0 split)
HDL2CHOL   < 1.34    to the right, agree=0.95, adj=0.75, (0 split)
TOTHDLFC   < 0.49    to the right, agree=0.95, adj=0.75, (0 split)
HDL2FC     < 0.355  to the right, agree=0.95, adj=0.75, (0 split)

Node number 8: 83 observations
predicted class=0  expected loss=0
class counts:      83      0
probabilities: 1.000 0.000

Node number 9: 14 observations,    complexity param=0.075

```

predicted class=1 expected loss=0.7142857
 class counts: 10 4
 probabilities: 0.714 0.286
 left son=18 (9 obs) right son=19 (5 obs)
 Primary splits:
 HDL3.P < 54.335 to the right, improve=6.719650, (0 missing)
 Mean_diastolic_BP1 < 83.165 to the left, improve=5.641188, (0 missing)
 APO_AII < 37.5 to the left, improve=5.641188, (0 missing)
 Age_at_baseline < 35.52 to the left, improve=4.676248, (0 missing)
 Mean_arterial_BP < 97.945 to the left, improve=4.676248, (0 missing)
 Surrogate splits:
 Age_at_baseline < 35.52 to the left, agree=0.857, adj=0.6, (0 split)
 BMI1 < 21.83 to the right, agree=0.857, adj=0.6, (0 split)
 IDL.CE < 18.21 to the left, agree=0.857, adj=0.6, (0 split)
 HDL3.PL < 21.455 to the left, agree=0.857, adj=0.6, (0 split)
 LPL_15minute < 41.405 to the left, agree=0.857, adj=0.6, (0 split)

Node number 10: 1 observations
 predicted class=0 expected loss=0
 class counts: 1 0
 probabilities: 1.000 0.000

Node number 11: 6 observations
 predicted class=1 expected loss=0
 class counts: 0 6
 probabilities: 0.000 1.000

Node number 14: 4 observations
 predicted class=0 expected loss=0
 class counts: 4 0
 probabilities: 1.000 0.000

Node number 15: 17 observations, complexity param=0.01
 predicted class=1 expected loss=0.1176471
 class counts: 2 15
 probabilities: 0.118 0.882
 left son=30 (1 obs) right son=31 (16 obs)
 Primary splits:
 HDL.STD < 0.85 to the left, improve=1.262791, (4 missing)
 IDLCHOL < 0.72 to the right, improve=1.227065, (0 missing)
 LDLTG < 0.395 to the right, improve=1.227065, (0 missing)
 IDLPL < 23.48 to the right, improve=1.227065, (0 missing)
 LDLPL < 136.025 to the right, improve=1.227065, (0 missing)

Node number 18: 9 observations
 predicted class=0 expected loss=0
 class counts: 9 0
 probabilities: 1.000 0.000

Node number 19: 5 observations, complexity param=0.01
 predicted class=1 expected loss=0.2

```
class counts:      1      4
probabilities: 0.200 0.800
left son=38 (1 obs) right son=39 (4 obs)
Primary splits:
  AGE_of_Onset      < 11.665  to the left,  improve=1.21327, (0 missing)
  Age_at_baseline   < 44.81   to the right, improve=1.21327, (0 missing)
  Duration_of_diabetes < 32.6   to the right, improve=1.21327, (0 missing)
  Raw_AER           < 3.175   to the left,  improve=1.21327, (0 missing)
  HEIGHT1          < 1.66    to the left,  improve=1.21327, (0 missing)
Surrogate splits:
  Gender.f splits as LR, agree=1, adj=1, (0 split)
```

```
Node number 30: 1 observations
predicted class=0 expected loss=0
class counts:      1      0
probabilities: 1.000 0.000
```

```
Node number 31: 16 observations, complexity param=0.01
predicted class=1 expected loss=0.0625
class counts:      1     15
probabilities: 0.063 0.938
left son=62 (1 obs) right son=63 (15 obs)
Primary splits:
  IDLCHOL < 0.72   to the right, improve=1.267967, (0 missing)
  IDLPL   < 23.48  to the right, improve=1.267967, (0 missing)
  IDLFC   < 0.255  to the right, improve=1.267967, (0 missing)
  IDLCE   < 0.48   to the right, improve=1.267967, (0 missing)
  IDLP    < 80.425 to the right, improve=1.267967, (0 missing)
```

```
Node number 38: 1 observations
predicted class=0 expected loss=0
class counts:      1      0
probabilities: 1.000 0.000
```

```
Node number 39: 4 observations
predicted class=1 expected loss=0
class counts:      0      4
probabilities: 0.000 1.000
```

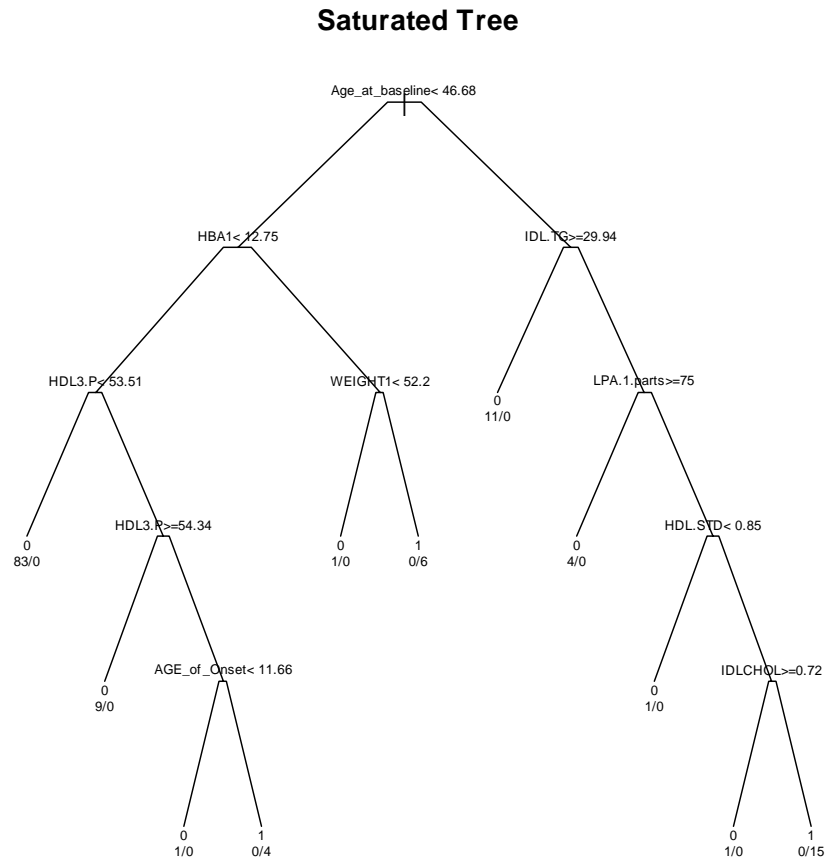
```
Node number 62: 1 observations
predicted class=0 expected loss=0
class counts:      1      0
probabilities: 1.000 0.000
```

```
Node number 63: 15 observations
predicted class=1 expected loss=0
class counts:      0     15
probabilities: 0.000 1.000
```

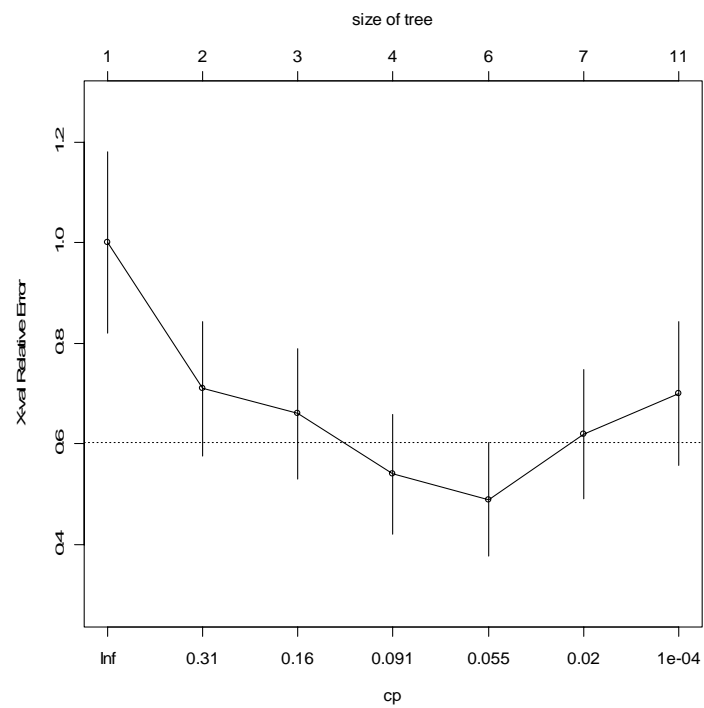
```

# Plotting the tree
> plot(study.rp, uniform=T, branch=0.1, main="Saturated Tree")
> text(study.rp, pretty=1, use.n=T, cex=0.6)

```



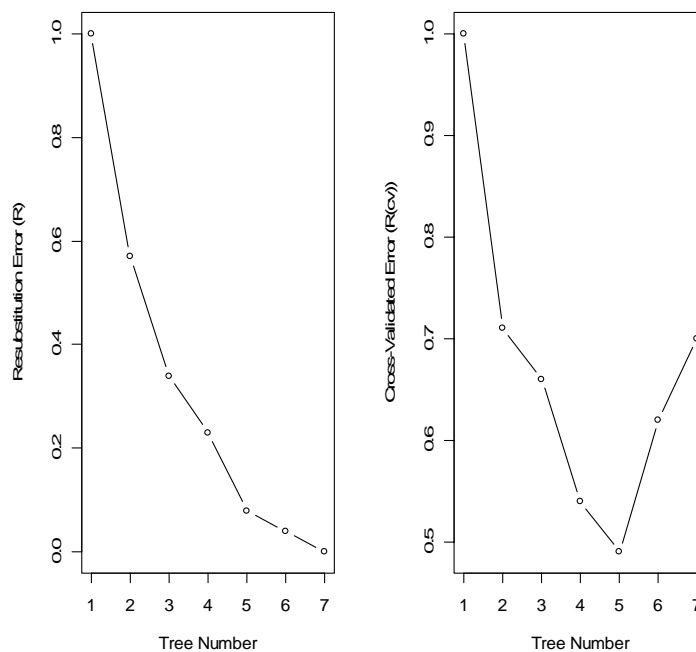
```
# Plot the complexity table that we saw in the summary of the tree.  
> plotcp(study.rp)
```



```

# Plot the resubstitution error
> opar <- par(mfrow = c(1, 2))
> with(study.rp,plot(cptable[,3],xlab="Tree Number",
ylab="Resubstitution Error (R)", type="b"))
> with(study.rp,plot(cptable[,4],xlab="Tree Number",
ylab="Cross-Validated Error (R(cv))", type="b"))
> par(opar)

```



```

# The cross-validation plot (right hand side above) shows that the minimum tree
error rates occur somewhere near the tree with 5 splits, 6 terminal nodes
(Tree Number 5).

```

```

# I chose cp= 0.04, refer to the cp table in the summary output.

```

```
> study.prune1<-prune(study.rp,cp=0.04)
> printcp(study.prune1)
```

Classification tree:

```
rpart(formula = y ~ . + Macroalbuminuria.f + Claas_of_AER.f +
  Gender.f + overwieight.f + Antihypertenive_med.f + ACE_Inhibitor.f +
  OTHER.AHT1.f + LIPID.LOWering1.f + Apirin1.f + NITRO1.f +
  Any.MicroVD.f + RETINOp athy1.f + NEUROp athy1.f + SMOKING1.f +
  HISTSMOK.f, data = studyl, method = "class", parms = list(loss = loss),
  control = rpart.control(minsplit = 1, cp = 1e-06))
```

Variables actually used in tree construction:

```
[1] Age_at_baseline HBA1 HDL3.P IDL.TG
```

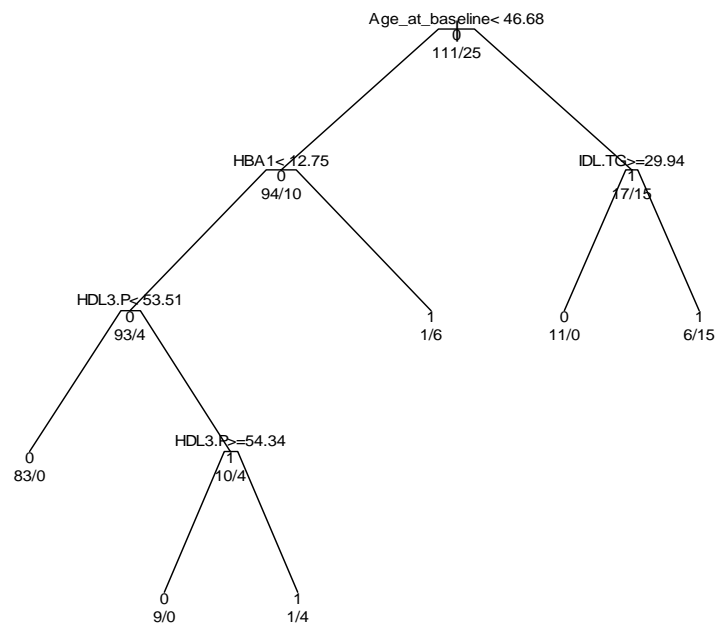
Root node error: 100/136 = 0.7353

n= 136

	CP	nsplit	rel error	xerror	xstd
1	0.430	0	1.00	1.00	0.18068
2	0.230	1	0.57	0.81	0.13739
3	0.110	2	0.34	0.57	0.11879
4	0.075	3	0.23	0.53	0.11847
5	0.040	5	0.08	0.48	0.11792

```
# Plot the new pruned tree
> plot(study.prunel,uniform=T,branch=0.1,margin=0.1,main="Pruned Tree1")
> text(study.prunel,pretty=1,all=T,use.n=T,cex=0.7)
```

Pruned Tree1



```
> study.prune2<-prune(study.rp,cp=0.075)
> printcp(study.prune2)
```

Classification tree:

```
rpart(formula = y ~ . + Macroalbuminuria.f + Claas_of_AER.f +
  Gender.f + overweight.f + Antihypertensive_med.f + ACE_Inhibitor.f +
  OTHER.AHT1.f + LIPID.LOWering1.f + Apirin1.f + NITRO1.f +
  Any.MicroVD.f + RETINOPathy1.f + NEUROPathy1.f + SMOKING1.f +
  HISTSMOK.f, data = study1, method = "class", parms = list(loss = loss),
  control = rpart.control(minsplit = 1, cp = 1e-06))
```

Variables actually used in tree construction:

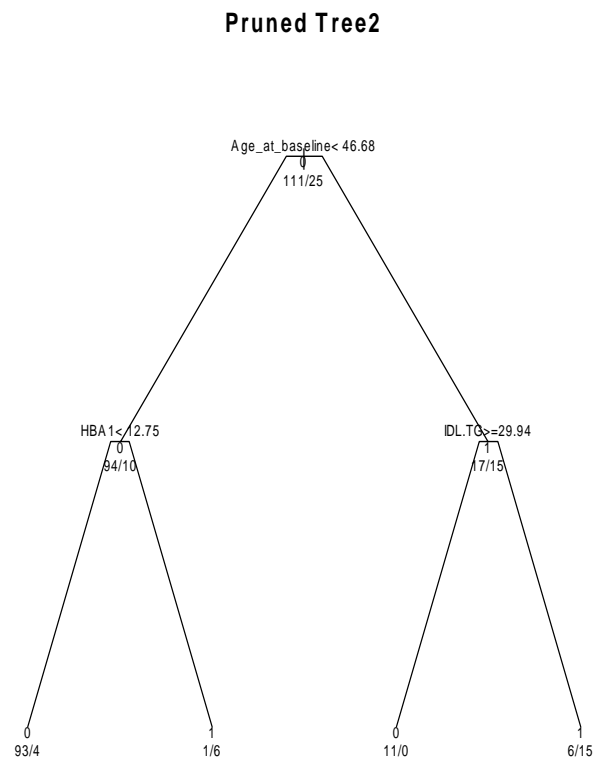
```
[1] Age_at_baseline HBA1 IDL.TG
```

Root node error: 100/136 = 0.7353

n= 136

	CP	nsplit	rel error	xerror	xstd
1	0.430	0	1.00	1.00	0.18068
2	0.230	1	0.57	0.81	0.13739
3	0.110	2	0.34	0.57	0.11879
4	0.075	3	0.23	0.53	0.11847

```
> plot(study.prune2, uniform=T, branch=0.1, margin=0.1, main="Pruned Tree2")  
> text(study.prune2, pretty=1, all=T, use.n=T, cex=0.7)
```



Appendix B

Node t	Node Class	Weight $P(t)$	Within Node Cost $r(t)$	Cost $R(t) = P(t) \times r(t)$
1	0	136/136	100/136	100/136
2	0	104/136	40/104	40/136
3	1	32/136	17/32	17/136
4	0	97/136	16/97	16/136
5	1	7/136	1/7	1/136
6	0	11/136	0	0
7	1	21/136	6/21	6/136
8	0	83/136	0	0
9	1	14/136	10/14	10/136
10	0	1/136	0	0
11	1	6/136	0	0
12	0	4/136	0	0
13	1	17/136	2/17	2/136
14	0	9/136	0	0
15	1	5/136	1/5	1/136
16	0	1/136	0	0
17	1	16/136	1/16	1/136
18	0	1/136	0	0
19	1	4/136	0	0
20	0	1/136	0	0
21	1	15/136	0	0

Table 7 Resubstitution estimates of Misclassification Costs. (Unit Cost: $C(0|1) = 4$)

Node(t)	R(t)	R(T)	T	α
17	1/136	0	2	1/136
15	1/136	0	2	1/136
13	2/136	0	3	1/136
9	10/136	0	3	10/(2 x 136)
7	6/136	0	4	6/(3 x 136)
5	1/136	0	2	1/(2 x 136)
4	16/136	0	4	16/(3 x 136)
3	17/136	0	5	17/(4 x 136)
2	40/136	0	6	40/(5 x 136)
1	100/136	0	11	100/(12 x 136)

Table 8 Cost Complexity obtained from saturated tree.

Minimum $\alpha_{SaturatedTree} = 1/136 = 0.0074$.

In RPART output $1/136 \times 136/100 = 0.01$

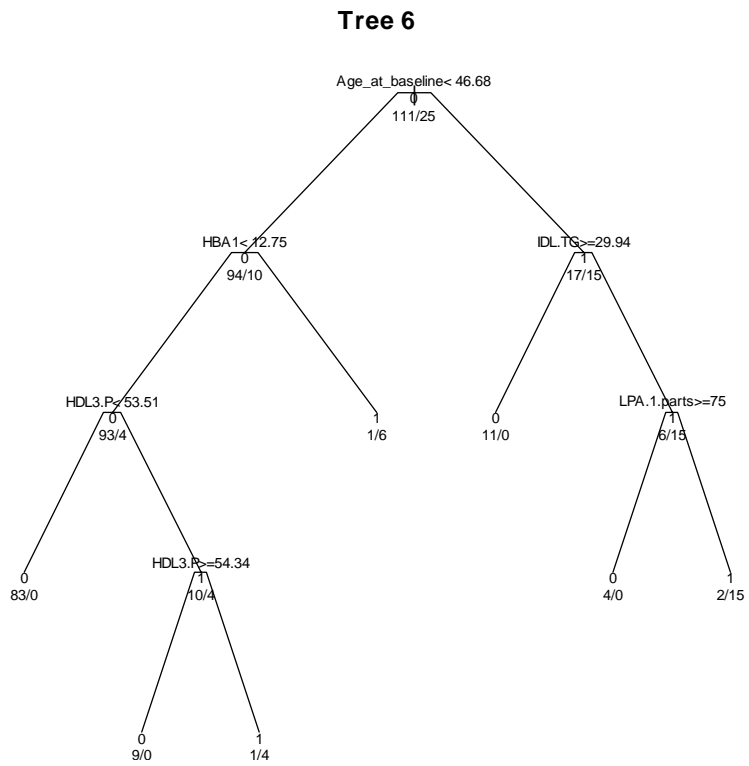


Figure 8.1 13 nodes tree with 6 splits and 7 terminal nodes.

Node(t)	R(t)	R(T)	T	α
13	2/136	0	1	
12	0	0	1	
9	10/136	1/136	2	9/136
8	0	0	1	
7	6/136	2/136	2	4/136
6	0	0	1	
5	1/136	0	1	
4	16/136	1/136	3	15/(2 x 136)
3	17/136	2/136	3	15/(2 x 136)
2	40/136	2/136	4	38/(3 x 136)
1	100/136	4/136	7	96/(6 x 136)

Table 9 Cost Complexity obtained from Tree 6.

Minimum $\alpha_5 = 4/136 = 0.0294$.

In RPART output $4/136 \times 136/100 = 0.04$

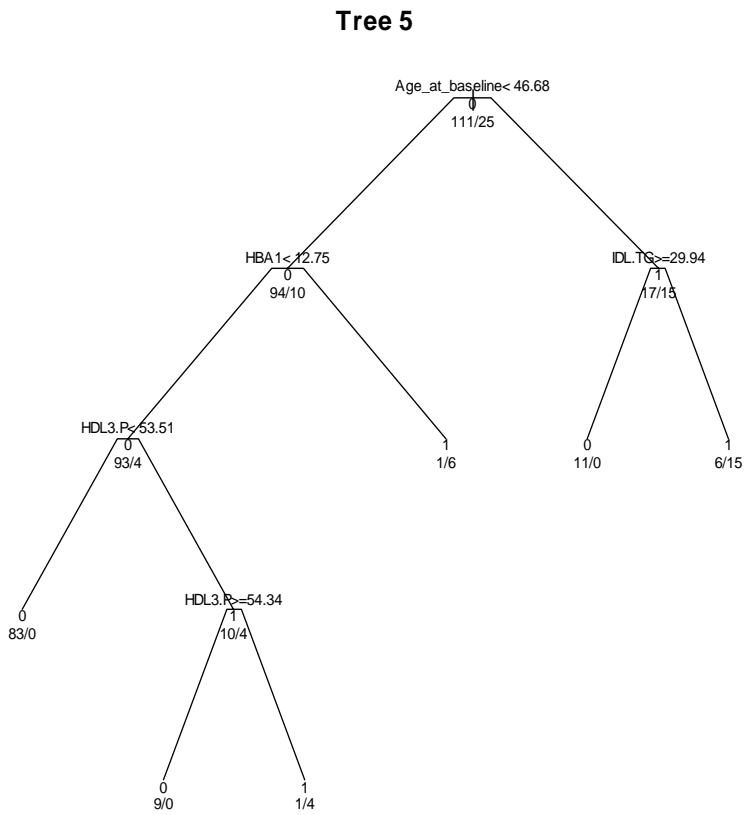


Figure 8.2 11 nodes tree with 5 splits and 6 terminal nodes.

Node(t)	R(t)	R(T)	T	α
9	10/136	1/136	2	9/136
8	0	0	1	
7	6/136	0	1	
6	0	0	1	
5	1/136	0	1	
4	16/136	1/136	3	15/(2 x 136)
3	17/136	6/136	2	11/136
2	40/136	2/136	4	38/(3 x 136)
1	100/136	8/136	6	92/(5 x 136)

Table 10 Cost Complexity obtained from Tree 5.

Minimum $\alpha_4 = 15/(2 \times 136) = 0.05515$.

In RPART output $15/(2 \times 136) \times 136/100 = 0.075$

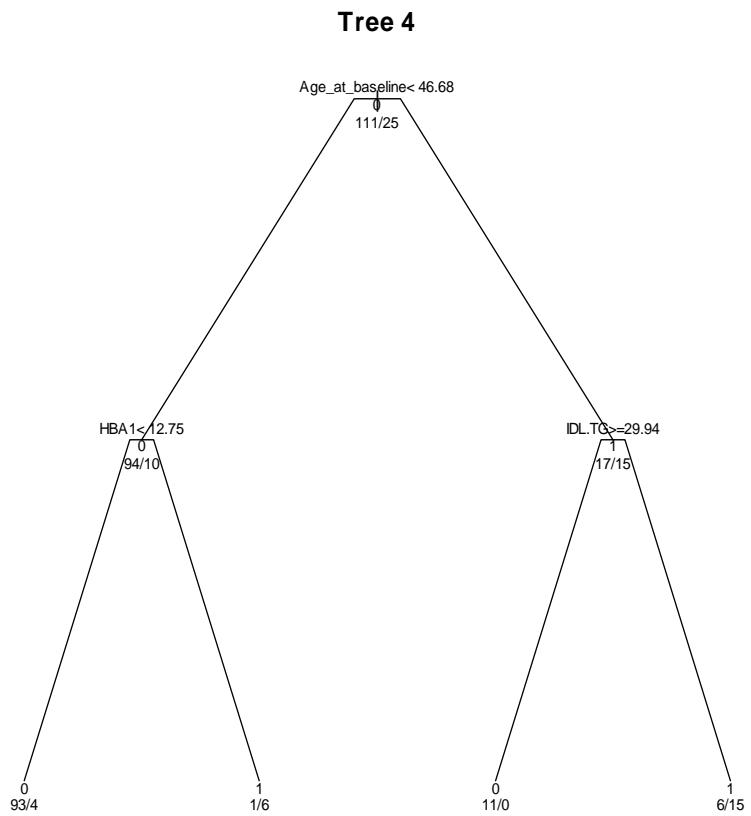


Figure 8.3 7 nodes tree with 3 splits and 4 terminal nodes.

Node(t)	R(t)	R(T)	T	α
7	6/136	0	1	
6	0	0	1	
5	1/136	0	1	
4	16/136	0	1	
3	17/136	6/136	2	11/136
2	40/136	17/136	2	23/136
1	100/136	23/136	4	77/(3 x 136)

Table 11 Cost Complexity obtained from Tree 4.

Minimum $\alpha_3 = 11/136 = 0.081$.

In RPART output $11/136 \times 136/100 = 0.11$

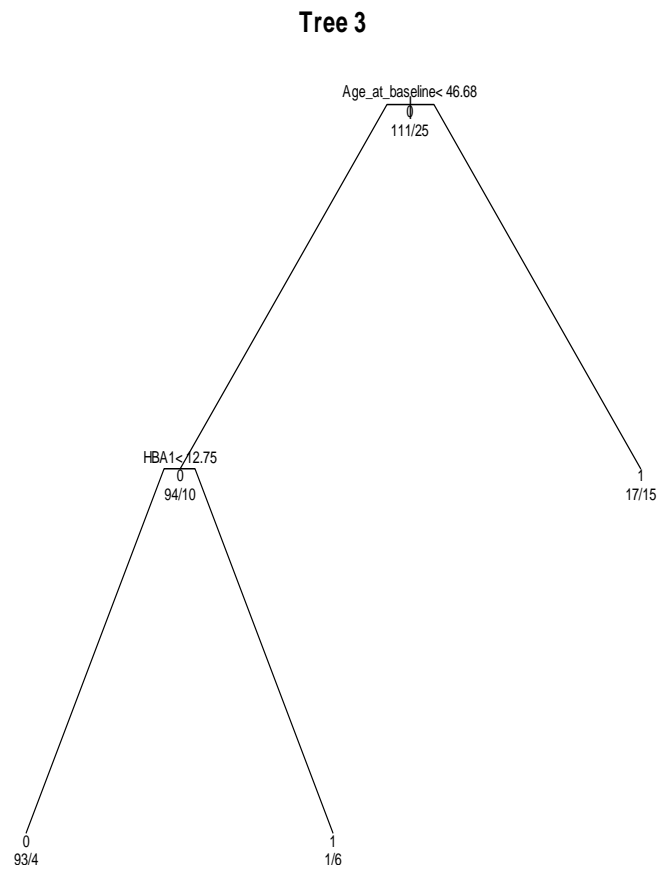


Figure 8.4 5 nodes tree with 2 splits and 3 terminal nodes.

Node(t)	R(t)	R(T)	T	α
5	1/136	0	1	
4	16/136	0	1	
3	17/136	0	1	
2	40/136	17/136	2	23/136
1	100/136	34/136	3	66/(2 x 136)

Table 12 Cost Complexity obtained from Tree 3.

Minimum $\alpha_2 = 23/136 = 0.1691$.

In RPART output $23/136 \times 136/100 = 0.23$

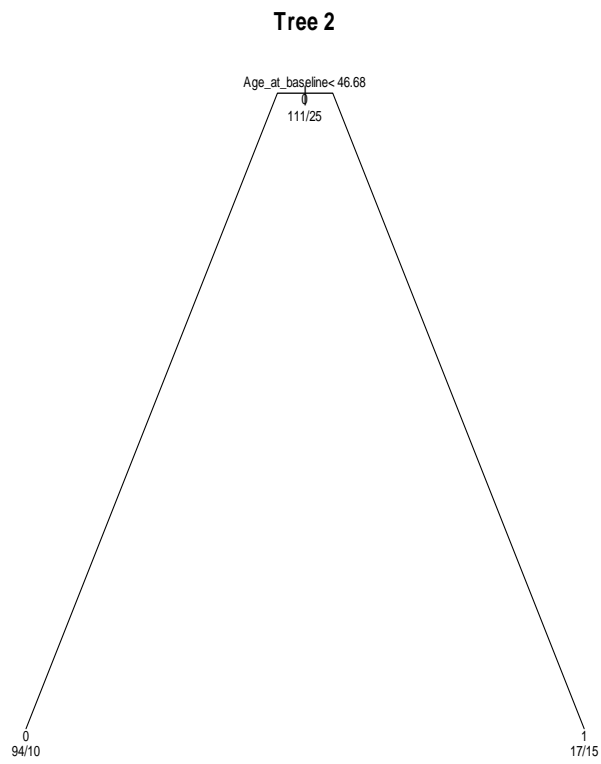


Figure 8.5 3 nodes tree with 1 split and 2 terminal nodes.

Node(t)	R(t)	R(T)	T	α
3	17/136	0	1	
2	40/136	0	1	
1	100/136	57/136	2	43/136

Table 13 Cost Complexity obtained from Tree 2.

Minimum $\alpha_1 = 43/136 = 0.3162$.

In RPART output $43/136 \times 136/100 = 0.43$

Acknowledgements

I would like to thank Merlin Thomas (Baker Heart Research Institute) for providing the data for this honours project and Ken Sharpe, my supervisor for his help and suggestions.

Bibliography

- [1] Leo Breiman, Jerome H. Friedman, Richard A. Olshen and Charles J. Stone, *Classification And Regression Trees*. Wadsworth, Inc, 1984.
- [2] Alvin C. Rencher , *Multivariate Statistical Inference and Applications*. John Wiley and Sons, Inc, 1998.
- [3] Elizabeth J. Atkinson, Terry M. Therneau , *An Introduction to Recursive Partitioning Using the RPART Routines*. Mayo Foundation, 2000.
- [4] Yisehac Yohannes and John Hoddinott, *Classification and Regression Trees:An Introduction*. International Food Policy Research Institute, 1999.
- [5] Petra Kuhnert and Bill Venables, *An Introduction to R: Software for Statistical Modelling & Computing*. CSIRO Mathematical and Information Sciences, 2005.
- [6] Roger J. Lewis, M.D., Ph.D., *An Introduction to Classification and Regression Tree (CART) Analysis*. <http://www.saem.org/download/lewis1.pdf>.
- [7] Heping Zhang, Burton Singer. *Statistics for Biology and Health*. Springer-Verlag, New York, Inc., 1999.
- [8] Electronic Textbook, StatSoft, *Classification and Regression Trees (C&RT)*. StatSoft, Inc., 1984-2003. <http://www.statsoft.com/textbook/stcart.html>.
- [9] Michigan Diabetes Research and Training Center, *HemoglobinA1C Fact Sheet*. MDRTC, <http://www.med.umich.edu/mdrtc/cores/ChemCore/hemoa1c.htm>.