University of Alberta

CLASSIFICATION IN THE MISSING DATA

by

Xin Zhang

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Statistics

Department of Mathematical and Statistical Sciences

Fall 2010
Edmonton, Alberta

Examining Committee

Dr. Ivan Mizera, Statistics, University of Alberta

Dr. Peng Zhang, Statistics, University of Alberta

Dr. Sandra Garvie-Lok, Anthropology, University of Alberta

Abstract

Missing data is always a problem when it comes to data analysis. This is especially the case in anthropology when sex determination is one of the primary goals for fossil skull data since many measurements were not available. We expect to find a classifier that can handle the large amount of missingness and improve the ability of prediction/classification as well. These are the objectives of this thesis.

Besides of the crude methods (ignore cases with missingness), three possible techniques in handling of missing values are discussed: bootstrap imputation, weighted-averaging classifier and classification trees. All these methods do make use of all the cases in data and can handle any cases with missingness.

The diabetes data and fossil skull data are used to compare the performance of different methods regarding to misclassification error rate. Each method has its own advantages and certain situations under which better performance will be achieved.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr. Ivan Mizera for his generous support, guidance and kindness throughout my whole research work.

I also really appreciate the other members of my thesis committee, Dr. Peng Zhang and Dr. Sandra Garvie-Lok, for their helpful suggestions.

I am also grateful to my parents and my fiance. Without their consecutive support and understanding, there is no way to finish this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis studies several methods to deal with missing data in the context of binary classification problem. A typical situation where such data occurs is anthropology.

Classical methods for binary classification problems include linear regression for classification, linear discriminant analysis, and logistic regression. All these methods are modeled on associated independent variables. This needs their values to be available. If some cases in training data contain missing values on the predictors, they have to be ignored and models are fitted based on complete cases only. Also if a new instance has missing values, then no class prediction/classification can be obtained. Thus, these crude methods suffer from loss of information and lack of accuracy since they discard too many observations.

Another technical tool used in classification problems is classification tree, which is a type of supervised learning methods. The attractions of decision trees are that they have some unique and simple ways to handle missing values, and if the proportion of missingness is large, the superiority of decision

trees presents. Some of the most widely used decision tree learning algorithms are Classification and Regression Trees (CART [1]), ID3(Quinlan, 1986) and C4.5 (Quinlan, 1993). Like most of machine learning methodologies, classification tree algorithms build classification trees on training data and test the models on validation data. The purpose of classification trees is to predict the right class of an instance based on the values of its explanatory variables. There are many different methods used by classification tree algorithms to handle the situation where missing occurs in predictors. In this work, we will focus on the CART algorithm and consider the attractive strategy, surrogate split, adopted in CART to deal with missing values.

To handle missing data problems, many different statistical methods have been developed. Little and Rubin [5] have summarized the most popular methods into four categories, one of which is data imputation. Imputation refers to replace each missing value by some value obtained from a predictive distribution, which is modeled based on observed data. There is a variety of imputation methods, the one that is applied to real datasets here is nonparametric bootstrap, which is computer intensive and has been used frequently in applied statistics. The advantage of bootstrap is its great simplicity. It is straightforward to apply the bootstrap to derive estimates of standard errors and confidence intervals. Efron (1994) showed that nonparametric bootstrap imputation requires no knowledge of the missing-data mechanism and the confidence interval turns out to give convenient and accurate answers. However, the correctness of imputation methods heavily depends on the model used to impute values. If the model is not correct, then all the analysis would be invalid.

A new method is proposed in this thesis, that is, the weighted-averaging

classifier. The main idea is that we take all of the submodels into account, fit classical regression for each submodel, predict the category of an instance on each submodel and then take a weighted average of all the predictions to get the final class prediction. The weight is chosen to be inversely proportional to the standard error of each prediction. In conventional methods, we may consider a single model with all associated predictors, or a smaller model with less predictors selected according to some criteria, thus, if an instance has missing value on one predictor, then it will be excluded from the analysis. The weighted-averaging classifier avoids such problems. As long as the instance has at least one non-missing value, it could be used to fit some submodels and its class prediction is obtainable. Thus, from this aspect, this new method in some sense overcomes the drawback of loss of information and enlarge the applicable sample size.

The thesis is structured as follows: Chapter 2 introduces the standard theory of missing data, imputation methods and nonparametric bootstrap methods to impute missing values. In Chapter 3, the basic concepts about Classification and Regression Trees (CART), surrogate variables, and implementation in R are elaborated. Typical methods for binary classification problem such as linear regression and logistic regression are discussed in Chapter 4, as well as formal introduction of the new method weighted-averaging classifier. Chapter 5 presents methods comparison and empirical studies. The linear and logistic regression models with all covariates, models selected by AIC criteria, nonparametric bootstrap to impute missing values, the surrogate split in CART and the weighted- averaging classifier with different weights are applied to real datasets and the performance is compared. Chapter 6 concludes the results of empirical studies.

Chapter 2

Standard Theory of Missing Data

Missing data can occur for a variety of reasons. For example, in surveys that ask people to report their income, typically, a large proportion of the respondents refuse to answer. Explicit refusals are only one cause of missing data. In longitudinal studies, subjects often drop out before the study is completed because they have moved out of the area, died, no longer see personal benefit to participating, or do not like the effects of the treatment. In self-administered surveys, people often skip or forget to answer some of the questions. Even well-trained interviewers sometimes may neglect to ask some questions. And sometimes respondents do not know the answer or do not have the information available to them. Sometimes the question is inapplicable to some respondents, such as asking unmarried people to rate the quality of their marriage. Because of all these reasons and many others, anyone who does statistical analysis will sooner or later meet the problems of missing data.

There are several missingness mechanisms and the concept was formal-

ized in the theory of Rubin (1976a), through the simple strategy of treating the missing data indicators as random variables and assigning them a distribution.

Before we briefly introduce different types of missingness mechanisms, we first define the notations we will use in this chapter. Usually, a data set can be presented in a 2 dimensional matrix with each row representing a subject and each column representing a variable. An entry in the matrix is the value of a variable for some subject. We refer to any data point that has an underlying value but is not observed as missing data.

We follow the notations in [5]. The full data set is denoted by $Y = (y_{ij})$,

$$Y = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{pmatrix}$$

and the missingness $(missing - data\ indicator\ matrix)$ is denoted by $M = (M_{ij})$,

$$M = \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1p} \\ M_{21} & M_{22} & \cdots & M_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{np} \end{pmatrix}$$

where

$$M_{ij} = \begin{cases} 1 & \text{if } y_{ij} \text{ is missing} \\ 0 & \text{otherwise} \end{cases}$$

We use Y_{obs}, Y_{miss} to denote the observed and missing components of Y, respectively.

2.1 Typical Types of Missingness

Statistically, missing data can be categorized into different types. Little and Rubin [5] classified the types of missingness into three categories according to their different statistical properties: missing completely at random ($MCAR$), missing at random (MAR) and not missing at random ($NMAR$).

As we defined above, the complete data is $Y = (y_{ij})$ and the missing-data indicator matrix is $M = (M_{ij})$, with M_{ij} indicating whether the corresponding Y_{ij} is missing or not. The missing-data mechanism is characterized by the conditional distribution of M given Y, say $f(M|Y, \phi)$, where ϕ denotes unknown parameters.

The three typical types of missingness are:

- Data are missing completely at random ($MCAR$), if the probability distribution of missingness does not depend on either the observed values or the missing values. That is, the distribution of M does not depend on the value of data Y, i.e.

$$f(M|Y, \phi) = f(M|\phi) \ for \ all \ Y, \ \phi.$$

  $MCAR$ means that the probability that an observation is missing is unrelated to the value of any variable, whether missing or observed. For example, data that are missing because a researcher dropped the test tubes or survey participants accidentally skipped questions are likely to be $MCAR$. Unfortunately, most missing data are not $MCAR$.

- Data are missing at random (MAR), if the probability distribution of missingness depends only on the observed components of Y, that is Y_{obs},

and not on the missing values Y_{miss}. That is,

$$f(M|Y,\phi) = f(M|Y_{obs},\phi) \; for \; all \; Y_{miss}, \; \phi.$$

This assumption is less restrictive than $MCAR$. The data can be considered as missing at random if the data meet the requirement that missingness does not depend on the value of Y_{ij} after controlling for another variable. For example, in an income survey, people who are depressed might be less inclined to report their income, and thus reported income will be related to depression (one variable in the survey). If within depressed patients the probability of reported income was unrelated to income level, then the data would be considered as MAR, though not $MCAR$.

- Data are not missing at random ($NMAR$), if the probability distribution of M depends on the missing values Y_{miss} in the data matrix Y. That is,

$$f(M|Y,\phi) = f(M|Y_{obs},Y_{miss},\phi).$$

Some literature also calls it *nonignorable missing data*. It commonly occurs when people do not want to reveal something very personal or unpopular about themselves. For example, if people with lower incomes are less likely to reveal them on a survey than people with higher incomes. Whether income is missing or observed is related to its value.

For instance, let us consider the simplest case given in [5]. The data structure is a univariate random sample for which some units are missing. Let $Y = (y_1, \ldots, y_n)^T$ where y_i denotes the value of a random variable for unit i,

and let $M = (M_1, \ldots, M_n)$ where $M_i = 0$ if unit i is observed and $M_i = 1$ if unit i is missing. Suppose the joint distribution of (y_i, M_i) is independent across units, in particular, the probability that a unit is observed does not depend on the values of Y or M for other units. Then

$$f(Y, M|\theta, \psi) = f(Y|\theta)f(M|Y, \psi) = \prod_{i=1}^{n} f(y_i|\theta) \prod_{i=1}^{n} f(M_i|y_i, \psi) \tag{2.1}$$

where $f(y_i|\theta)$ denotes the density of y_i indexed by unknown parameter θ, which is the parameter of interest. ψ is the parameter for the distribution of M. Thus, in this case, $\phi = (\theta, \psi)$. And $f(M_i|y_i, \psi)$ is usually the density of a Bernoulli distribution for the binary indicator M_i with probability $\Pr(M_i = 0|y_i, \psi)$ that y_i is missing. If missingness is independent of Y, that is if $\Pr(M_i = 1|y_i, \psi)$=ψ, a constant that does not depend on y_i, then the missing data mechanism is $MCAR$ or in this case equivalently MAR. If the missingness depends on y_i, then the mechanism is $NMAR$ since the probability depends on y_i that are missing, assuming that there are some missing values in Y.

Suppose r is the number of cases in the sample with respondence. Sometime, we might want to carry out analysis on the completed cases and the sample size is reduced from n to r [5]. For example, if we assume the values are normally distributed and want to make inferences about the mean, we might estimate the mean by the sample mean of the r responding units with standard error $s/\sqrt{r}$, where s is the sample standard deviation of the responding units. If the missingness mechanism is $MCAR$ or MAR, then the observed cases are a random subsample of all the cases. This strategy is valid. However, if the data are $NMAR$, the analysis based on the responding subsample is generally biased for the parameter.

The following definitions and proofs are reproduced from [5], Definition 6.4 and 6.5, pp.119-120.

Theorem 2.1.1. *The missing-data mechanism is ignorable for likelihood inference if (1) MAR: the missing data are missing at random; and (2) Distinctness: the parameters θ and ψ are distinct, in the sense that the joint parameter space of (θ, ψ) is the product of the parameter space of θ and the parameter space of ψ.*

PROOF. The joint distribution of M and Y can be determined by

$$f(Y, M|\theta, \psi) = f(Y|\theta)f(M|Y, \psi), \quad (\theta, \psi) \in \Omega_{\theta,\psi}$$

where $\Omega_{\theta,\psi}$ is the parameter space of (θ, ψ).
Let $f(Y, M|\theta, \psi) \equiv f(Y_{obs}, Y_{miss}, M|\theta, \psi)$ denote the probability or density of the joint distribution of Y_{obs}, Y_{miss} and M.
The distribution of the observed data is obtained by the following:

$$f(Y_{obs}, M|\theta, \psi) = \int f(Y_{obs}, Y_{miss}|\theta)f(M|Y_{obs}, Y_{miss}, \psi)dY_{miss}$$

The full likelihood of the parameters given the observed information (Y_{obs}, M) is

$$L_{full}(\theta, \psi|Y_{obs}, M) \propto f(Y_{obs}, M|\theta, \psi),$$

If the distribution of missingness does not depend on the missing data, i.e. $MCAR$ or MAR holds and θ, ψ are distinct (in the sense that $\Omega_{\theta,\psi} = \Omega_\theta \times \Omega_\psi$), then

$$f(M|Y_{obs}, Y_{miss}, \psi) = f(M|Y_{obs}, \psi),$$

and

$$\begin{aligned} f(Y_{obs}, M|\theta, \psi) &= f(M|Y_{obs}, \psi) \int f(Y_{obs}, Y_{miss}|\theta) dY_{miss} \\ &= f(M|Y_{obs}, \psi) f(Y_{obs}|\theta) \end{aligned} \tag{2.2}$$

Therefore, if θ and ψ are distinct,

$$L_{full}(\theta, \psi|Y_{obs}, M) \propto f(Y_{obs}, M|\theta, \psi) \propto f(Y_{obs}|\theta) \propto L_{ignored}(\theta|Y_{obs}),$$

that is, the likelihood-based inferences for θ from $L_{full}(\theta, \psi|Y_{obs}, M)$ will be the same as likelihood-based inferences for θ from $L_{ignored}(\theta|Y_{obs})$. We can ignore the missing-data mechanism since the simpler likelihood based only on observed data is proportional to the more complex one that includes also the missing data-mechanism. □

Theorem 2.1.2. *The missing-data mechanism is ignorable for Bayesian inference if (1) MAR: the missing data are missing at random; and (2) The parameters θ and ψ are a priori independent, that is, the prior distribution has the form $p(\theta, \psi) = p(\theta)p(\psi)$.*

PROOF. The posterior distribution of θ and ψ is given by

$$p(\theta, \psi|Y_{obs}, M) \propto L_{full}(\theta, \psi|Y_{obs}, M) \times p(\theta, \psi),$$

Suppose if the conditions in Theorem 2.1.2 hold, we have

$$p(\theta, \psi) = p(\theta)p(\psi),$$

and

$$\begin{aligned} p(\theta, \psi|Y_{obs}, M) &\propto L_{full}(\theta, \psi|Y_{obs}, M)p(\theta, \psi) \\ &\propto p(\psi)f(M|Y_{obs}, \psi)p(\theta)f(Y_{obs}|\theta) \\ &\propto p(\psi)L(\psi|M, Y_{obs})p(\theta)L_{ignored}(\theta|Y_{obs}) \\ &\propto p(\psi|M, Y_{obs})p(\theta|Y_{obs}) \end{aligned}$$

Therefore, inference about θ can be only based on the posterior distribution $p(\theta|Y_{obs})$ ignoring the missing-data mechanism,since the simpler posterior distribution of θ conditional only on observed data is proportional to the more complex one that conditional also on missing-data mechanism. □

2.2 Treatment of Missing Data in Statistics

Many different statistical methods have been proposed to deal with missing data problems. Little and Rubin [5] had summarized the most popular methods into four categories. In this section, we will give a brief description of basic approaches to handle missing data.

2.2.1 Procedures Based on Completely Recorded Units

By far, the most simple and common methods are just to ignore the data with missing values in any one of the variables and analyze the remaining completely recorded units only. These types of methods are easy to be carried out, and have been widely adopted (usually by default) in many popular statistical packages. When only a small amount of missing values presents, these procedures may be satisfactory.

However, these procedures obviously suffer from loss of information.

This is especially true if a large proportion of the data contain missing values on only a few variables, which will produce a larger estimated variance, or in other words, a more unstable estimation.

It can also lead to serious biases. For example, when low income individuals are less likely to report their income level and more higher income subjects are left in the sample, the resulting mean income given by complete case analysis is biased in favor of higher incomes.

2.2.2 Weighting Methods

Weighting methods have been widely used in survey analysis. An ideal survey sample should obtain subsamples from different subpopulations with sample sizes proportional to their corresponding proportions in the whole population, since a sample should represent the whole population well and reflect the characteristics of the population from which it is drawn. However, in most situations, researchers can not survey the entire population for two reasons. First of all, the cost is too high to reach all the subpopulations, so the samples are less than they should be. In some other cases, the populations are dynamic in that individuals making up the populations may change over time. Therefore, some subpopulations would be under-sampled. This means some data from under-sampled sub-populations are missing.

The essence of weighting methods is to produce weights by which the non-missing cases are multiplied, in order to account for the cases with missing values which have been deleted before the analysis. Weighting methods give larger weights to observations from under-sampled subpopulations to make the resulting samples more close to the entire population. However, a major

problem of these methods is that they do not provide ways to obtain appropriate variances for estimates. Thus, this drawback precludes these methods from being generally recommended.

2.2.3 Imputation Methods

Imputation methods refer to replacing each missing value by some value obtained from a predictive distribution of the missing values. The predictive distribution for the imputation is modeled based on the observed data. After imputation, the data can be considered as complete and the standard analysis can be applied. Many different imputation methods have been developed. We only review some of them.

- Mean/Mode/Median Imputation. Missing values of some variable are filled by the mean/mode/median of the variable calculated from the responding subjects.

  The drawback of this method is that it tends to under estimate the variability of the data since we use mean/mode/median as replacement values, which are measures of the "center" of their estimated distribution.

- Regression Imputation. Each missing value is replaced by the predicted value from a regression of the variable with missing values on complete variables. The appropriate regression model depends on the type of the to-be-imputed variable. A probit or logit model is used for binary variables, Poisson or other count models for integer-valued variables, and for continuous variables, we normally use OLS or related models.

- Draw Imputation. This method computes the replacement values by

a random draw, instead of mean/mode/median, from the distribution modeling the missing values.

- Multiple Imputation. Instead of filling in a single value for each missing value, Rubin's (1987) multiple imputation procedure replaces each missing value with a set of plausible values that reflect the uncertainty about the right value to impute. These multiple imputed data sets are then analyzed by using standard procedures for complete data and combining the results from these analysis. The advantage of this approach is that it incorporates and measures imputation uncertainty.

- Resampling Methods. These methods are computer intensive methods and have been used frequently in applied statistics. The main idea is that we resample the original data set, and use some imputation method to create a complete data set, and then analyze it using standard techniques. Repeat this process multiple times, and combine the results. Two popular and widely used resampling methods are bootstrap and jackknife.

The correctness of imputation methods heavily depends on the model used to generate the imputation values. Thus, the model must be "correct" in some sense. If incorrect data are imputed, all the following analysis will be wrong.

2.2.4 Model-Based Procedures

A broad class of procedures is generated by defining a model for the observed data and basing inferences on the likelihood or posterior distribution

under that model, with parameters estimated by procedures such as maximum likelihood.

When the probability of missing observations does not depend on the missing values as in $MCAR$ or MAR data, the missing mechanism could be ignored for likelihood inference. The objective is to maximize the likelihood

$$L(\theta|Y_{obs}) = \int f(Y_{obs}, Y_{miss}|\theta)dY_{miss} \tag{2.3}$$

with respect to θ. The distribution of all the data Y can be factored as

$$f(Y|\theta) = f(Y_{obs}, Y_{mis}|\theta) = f(Y_{obs}|\theta)f(Y_{mis}|Y_{obs}, \theta), \tag{2.4}$$

where $f(Y_{obs}|\theta)$ is the density of the observed data Y_{obs} and $f(Y_{mis}|Y_{obs}, \theta)$ is the density of the missing data given the observed data. The corresponding decomposition of the likelihood is

$$l(\theta|Y) = l(\theta|Y_{obs}, Y_{mis}) = l(\theta|Y_{obs}) + \ln f(Y_{mis}|Y_{obs}, \theta). \tag{2.5}$$

The objective is to estimate θ by maximizing the incomplete-data log-likelihood $l(\theta|Y_{obs})$ with respect to θ for fixed Y_{obs}. We could write

$$l(\theta|Y_{obs}) = l(\theta|Y) - \ln f(Y_{mis}|Y_{obs}, \theta). \tag{2.6}$$

However, in some cases, the latter $\ln f(Y_{obs}, Y_{mis}|\theta)$ is difficult to be minimized. Unlike the mean of a normal distribution, the maximum likelihood estimates with missing data do not have a closed form solution. Instead, the solution must be found either using numerical methods, such as Newton-Raphson to

find the maximum of the likelihood.

The advantage of this method are flexibility and availability of estimates of variance that take into account incompleteness in the data.

2.2.5 Applications: Bootstrap Imputation

In this section, we introduce a type of resampling methods to deal with missing data: bootstrap imputation. The primary advantage of the nonparametric bootstrap imputation over other imputation methods is that it does not depend on the missing-data mechanism. Moreover, bootstrap imputation also successfully incorporates the estimates of uncertainty (variance) associated with the imputed data.

The basic ideas of nonparametric bootstrap for complete data are as follows: Let $\hat{\theta}$ be an estimate of a parameter θ based on a random sample $Y = (y_1, y_2, \ldots, y_n)^T$. Suppose $Y^{(b)}$ is a sample of size n obtained from the original sample Y using simple random sampling with replacement, and $\hat{\theta}^{(b)}$ is the estimate of θ based on sample $Y^{(b)}$ using standard estimation procedures, where $b = 1, 2, \ldots, B$ indexes the drawn sample from Y. Then repeat this process B times and $(\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)})$ represents the sequence of estimates of θ. The bootstrap estimate of θ is defined as the average of the B bootstrap estimates:

$$\hat{\theta}_{boot} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^{(b)}. \tag{2.7}$$

The bootstrap estimate of the variance of $\hat{\theta}$ or $\hat{\theta}_{boot}$ is

$$\hat{V}_{boot} = \frac{1}{B-1} \sum_{b=1}^{B} (\hat{\theta}^{(b)} - \hat{\theta}_{boot})^2. \tag{2.8}$$

Efron (1987) has been shown that $\hat{V}_{boot}$ is a consistent estimate of the variance of $\hat{\theta}$ or $\hat{\theta}_{boot}$ with large samples, i.e. as n and B go to infinity. Thus, a $100(1-\alpha)\%$ bootstrap confidence interval for θ is calculated as follows if the bootstrap distribution is approximately normal:

$$CI_{norm}(\theta) = \hat{\theta} \pm z_{1-\alpha/2}\sqrt{\hat{V}_{boot}} \tag{2.9}$$

where $z_{1-\alpha/2}$ is the $100(1-\alpha)\%$ percentile of the normal distribution.

Suppose there is a simple random sample $Y = (y_1, y_2, \ldots, y_n)^T$, however, some observations y_i are missing. Then the bootstrap estimates $(\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)})$ are obtained as follows:

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

1. Draw a bootstrap sample $Y^{(b)}$ by simple random sampling with replacement from the original incomplete sample Y.

2. Replace the missing values in $Y^{(b)}$ by imputed values calculated from some imputation procedure $Imputation$ based on the bootstrap sample $Y^{(b)}$, so that $\hat{Y}^{(b)} = Imputation(Y^{(b)})$.

3. Compute the estimate $\hat{\theta}^{(b)}$ for the imputed complete data $\hat{Y}^{(b)}$.

The nonparametric bootstrap method to impute missing values can be implemented using the following algorithm:

Algorithm:

1. Draw B (e.g. 2000) bootstrap samples.

2. For each bootstrap sample, $b = 1, 2, \ldots, B$, impute missing values using the following method:

   - Replace missing values by median if the predictor is quantitative or by mode if the predictor is qualitative. This is also known as "`roughfix` ".
   - Apply standard analysis procedures to each imputed bootstrap sample and obtain the estimate $\hat{\theta}^{(b)}$.

3. Repeat Step 2 B times (e.g.B=2000).

4. Compute the bootstrap estimate $\hat{\theta}_{boot}$ and the estimate of its variance $\hat{V}_{boot}$

5. Study the empirical distribution of the estimates $\hat{\theta}^{(b)},\ b = 1, 2, \ldots, B$.

6. Construct confidence intervals.

Chapter 3

Classification and Regression Trees

We start with an introduction of the popular method for tree-based classification and regression called CART. This thesis work will mainly focus on classification trees.

3.1 Classification Trees

Classification trees has been around since the 1960s, but computational requirements limits their use until recently. Breiman et al.(1984) brought classification trees into the attention of statisticians. In physical anthropology, classification trees has been considered as an alternative to linear discriminant analysis [2] when missing data reduce the size of the data set.

Classification trees can be applied to classification problems when the response variable is categorical with each category representing one target class. The main purpose of classification trees is to obtain the most accurate

classification/prediction of an object based on the values of its explanatory variables.

3.1.1 Splitting A Tree And Decision Process

A binary tree T is split as follows. We start with the root node, which is equivalent to the predictor space R, and split it into two *daughter nodes*, say R_1 and R_2 according to a related splitting rule which is based on a single explanatory variable. We further split R_1 and R_2 into two parts to get R_1, R_2, R_3, R_4, and continue doing this until we have a collection of subsets $R_1, \ldots, R_b$ of R as illustrated in Figure 3.1. The terminal nodes form a partition of R. At each split, the data is partitioned into two mutually exclusive groups, each of which is as homogeneous as possible. The splitting procedure is then applied to each group separately. The objective is to partition the response into homogeneous groups, but also needs to keep the tree small. The size of a tree equals the number of terminal node.

The way that explanatory variables are used to form splitting rules depends on their types. For quantitative predictors, the splitting rule is of the form $\{x_i \leq s\}$, and we assign observations with $\{x_i \leq s\}$ or $\{x_i > s\}$ to the left or right daughter node respectively . For qualitative predictors, the splitting rule is based on a category subset C with the form of $\{x_i \in C\}$, and assign cases with $\{x_i \in C\}$ or $\{x_i \notin C\}$ to the left or right daughter node respectively. Each terminal subset will be assigned a class label and there may be two or more terminal nodes having the same class label.

Why binary splits [3]? We might consider multi-way splits into more than two groups, rather than splitting each node into just two groups at each

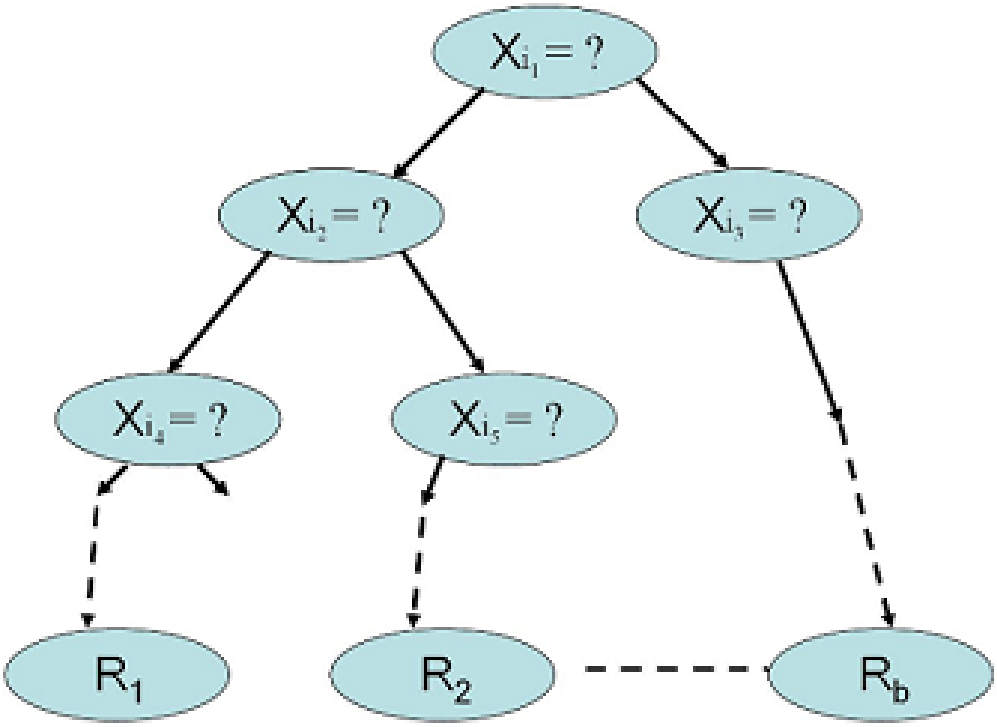


Figure 3.1: Split A Tree

step (as above). This can sometimes be very useful, but it is not a general strategy. The problem is that multi-way splits divide the data too quickly, leaving insufficient data at the next level down. Hence, we would want to use multi-way splits only when needed. The binary splits are preferred, since multi-way splits can be achieved by a series of binary splits.

Given a classification tree, the decision for a instance is made as follows. Beginning from the root node, at each node (including the root), a test is carried out on some predictor variable, and with the outcome, the instance is forward to one of the daughter nodes. This process will continue until a terminal node is reached and this terminal node contains the final classification/prediction we will make.

3.1.2 Growing A Classification Tree

We can use decision tree learning algorithms to produce classification trees from training data. At each split step, a decision tree learning algorithm searches over all possible splits and pick the best one in terms of the homogeneity of nodes. The homogeneity of nodes is measured by impurity, which takes the value zero for completely homogeneous nodes, and as homogeneity decreases, the value of impurity increases. Thus, maximizing the homogeneity of the groups is equivalent to minimizing their impurity measures. There are many measures of impurity for classification trees. We will discuss some of them briefly in the next section.

The CART algorithm is one of the popular decision tree learning algorithms and it's a form of binary recursive partition. At each node, the algorithm searches through the predictor variables one by one, and for each

variable, it finds the best split. For example, if we have p measurements or predictors, then we got p best splits. After that, the algorithm compares the p single variable splits and selects the best of the best. In order to select the best split at each node of the tree, the algorithm calculates the reduction in impurity from parent nodes to daughter nodes for each potential split. The purpose of the CART algorithm is to maximize this reduction at each step.

In general, four main steps are needed to grow a tree.

1. The first step is to select an appropriate measure of homogeneity or impurity.

2. The second step is to choose a methodical way to grow a tree and some stopping rule (for example, we can stop when the number of cases reaching each terminal node is very small, or the observations in each terminal node are homogeneous enough).

3. The third step is to prune the large tree we grew in the previous step which will be discussed later. As we prune the tree gradually, a sequence of subtrees from the largest to the smallest with root node only is obtained.

4. The fourth step is to choose the "best" tree according to some criteria. Our goal is to choose the tree with the smallest size, and also categorizing the training data effectively.

3.1.3 Impurity Measures

For classification trees, if response variable is categorical taking values $1, 2, \ldots, K$, and suppose we have partitioned the predictor space into b regions

$R_1, R_2, \ldots, R_b$. Conventional algorithm models the response in each region R_m as a constant c_m, which can be expressed as [3]:

$$f(x) = \sum_{m=1}^{b} c_m I\left(X \in R_b\right), \tag{3.1}$$

where $R_m, m = 1, 2, \ldots, b$ represents the space of b terminal nodes.

For classification trees, impurity measure $Q_m\left(T\right)$ is defined in terms of the proportions of responses in each category. We represent a region R_m with N_m observations, and let

$$\hat{p}_{k|m} = \hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = k\right) \tag{3.2}$$

be the proportion of class k observations in node m, $k \in (1, 2, \ldots, K)$. We use the short notation $\hat{p}_{mk}$ instead of $\hat{p}_{k|m}$ to denote the estimated probability of an observation in the class k given that it is in the node m ($\sum_{k=1}^{K} \hat{p}_{k|m} = 1$). We classify the observations in node m to the majority class $k\left(m\right) = \arg\max_k \hat{p}_{mk}$.

Define the impurity of a node m as

$$Q_m\left(T\right) = \sum_{k=1}^{K} f(\hat{p}_{mk}) \tag{3.3}$$

Since we would like $Q_m\left(T\right) = 0$ when node m is pure, the impurity function f must be concave with $f(0) = f(1) = 0$. Three commonly used criteria $Q_m\left(T\right)$ of node impurity are:

1) Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}\left(1 - \hat{p}_{mk}\right) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2 \tag{3.4}$$

2) Cross-entropy or deviance:

$$\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{3.5}$$

3) Misclassification error:

$$\frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i \notin k\left(m\right)\right) = 1 - \hat{p}_{m,k(m)} \tag{3.6}$$

For the special case, if the response variable only have two categories, and if p is the proportion in the second class, then these measures are $2p\left(1-p\right)$, $-p\log p-\left(1-p\right)\log\left(1-p\right)$, $1-\max\left(p,1-p\right)$, respectively. They are shown in Figure 3.2. In this case, all these three have similar properties. They take minimums at $p=0$ and $p=1$, and the maximum at $p=0.5$. But the cross-entropy and the Gini index are differentiable, and hence more amenable to numerical optimization. The CART algorithm adopted the Gini criteria as the default option for binary response Y.

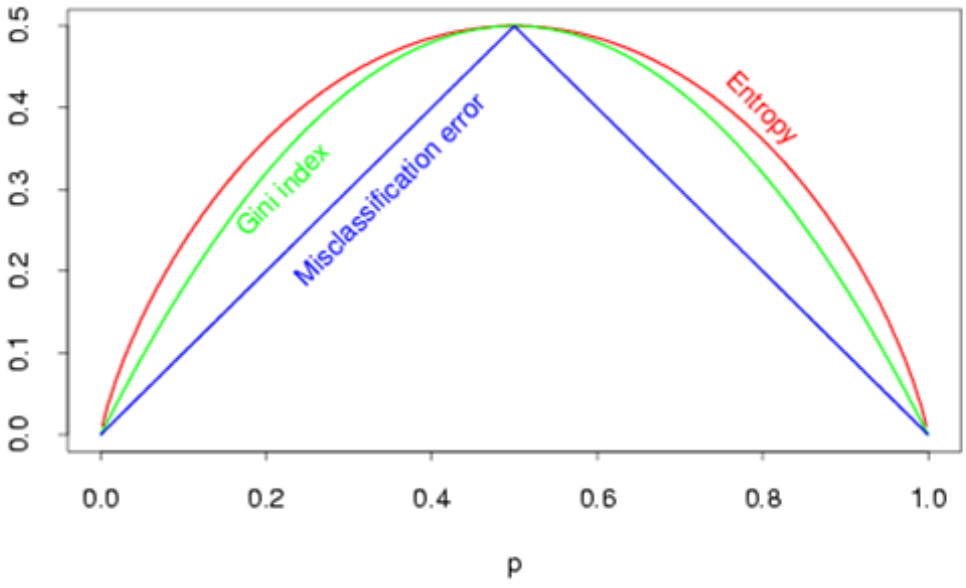


Figure 3.2: Node Impurity Measures For Two-class Classification

In addition, cross-entropy and the Gini index are more sensitive to changes in the node probabilities than the misclassification error. Assume a two-class classification problem with 400 observations in each class. This is denoted by(400/400). Suppose one split would produce two nodes, (300/100) and (100/300), and an alternative split would produce two nodes with (200/400) and (200/0). For both splits, the misclassification error rate is 0.25. Although both of the nodes produced by the former split need to be split further, the latter split produces only one node that needs further splitting and the other node that is pure. Both the Gini index and cross-entropy are lower for the second split. For this reason, in terms of tree construction, either the Gini index or cross-entropy should be used for the latter split.

As we defined above, for a classification tree, the impurity function $Q_m(T)$ of a node m could be any one of (2.3)-(2.5). We use m_L and m_R

to denote the left and right daughter nodes of m, respectively, and assume a candidate split s of node m sends a proportion p_{m_L} of the data in m to m_L and p_{m_R} to m_R. Define the reduction in impurity to be

$$\Delta Q(s, m) = Q_m(T) - p_{m_L} Q_{m_L}(T) - p_{m_R} Q_{m_R}(T) \tag{3.7}$$

Then the splitting criterion is that we choose the split s^* to maximize the impurity reduction in node m, i.e.,

$$\Delta Q(s^*, m) = \max_{s \in S} \Delta Q(s, m)$$

where S is the set of all possible splits for node m.

Suppose we have done some splitting and arrived at a current set of terminal nodes. Denote the current set of terminal nodes by $\widetilde{T}$, and define

$$I_m(T) = Q_m(T) p(m)$$

and define the *tree impurity* $Q(T)$ by

$$Q(T) = \sum_{m \in \widetilde{T}} I_m(T) = \sum_{m \in \widetilde{T}} Q_m(T) p(m) \tag{3.8}$$

It is easy to see that selecting the splits that maximize $\Delta Q(s, m)$ is equivalent to selecting those splits that minimize the overall tree impurity $Q(T)$. Take any terminal node $t \in \widetilde{T}$ and consider a candidate split s, which splits the

node into t_L and t_R. Then the resulting new tree T' has the impurity

$$Q(T') = \sum_{\widetilde{T}-t} I_{t'}(T) + I_{t_L}(T') + I_{t_R}(T')$$

The decrease in tree impurity is

$$Q(T) - Q(T') = I_t(T) - I_{t_L}(T') - I_{t_R}(T')$$

This only depends on the node t and split s. Thus, maximizing the decrease in tree impurity by splits s on node t is equivalent to maximizing the expression

$$I_t(T) - I_{t_L}(T') - I_{t_R}(T') \tag{3.9}$$

As we define before, p_{t_L} and p_{t_R} are the proportions of data in node t to t_L and t_R respectively. And we use $p(t)$ to denote the proportion of data in node t.

$$p_{t_L} = \frac{N(t_L)}{N(t)} = \frac{\frac{N(t_L)}{N}}{\frac{N(t)}{N}} = \frac{p(t_L)}{p(t)}, p_{t_R} = \frac{p(t_R)}{p(t)}$$

Then

$$p_{t_L} + p_{t_R} = 1$$

and Equation 3.9 can be written as

$$\begin{aligned} Q(T) - Q(T') &= I_t(T) - I_{t_L}(T') - I_{t_R}(T') \\ &= Q_t(T)p(t) - Q_{t_L}(T')p(t_L) - Q_{t_R}(T')p(t_R) \\ &= [Q_t(T) - Q_{t_L}(T')p_{t_L} - Q_{t_R}(T')p_{t_R}]p(t) \\ &= \Delta Q(s,t)p(t) \end{aligned}$$

Since $Q(T) - Q(T')$ differs from $\Delta Q(s,t)$ only by the factor $p(t)$, so the same split s^* maximizes both expressions. Thus, the split selection procedure can be considered as a repeated attempt to minimize overall tree impurity.

3.1.4 Pruning A Tree

A natural question is how large should we grow a tree? Apparently, a very large tree might overfit the data, while a small tree may not be able to capture the important structure of the data. Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen according to the data. One approach to grow a tree using a splitting criterion is to continue the splitting procedure until the improvement of homogeneity due to additional splits is less than a pre-specified cutoff, and then we take this resulting tree as the best tree. For example, the stopping rule, that is, on finding a criterion for declaring a node terminal, can be defined as follow: set a threshold $\beta > 0$, and state a node m terminal if

$$\max_{s \in S} \Delta Q(s, m) < \beta \tag{3.10}$$

However, Breiman et al. [1] points out the weaknesses of this approach. If β is set too low, then there is too much splitting and the tree is too large. But, if we increase the value of β, there may be such nodes m satisfying Equation 3.10. But the descendant nodes m_L, m_R of m may have splits with large decreases in impurity. If we declare m terminal, we lose the good splits on m_L or m_R.

To get an optimal solution for the problem of finding the best tree, Breiman et al. [1] introduced three basic ideas.

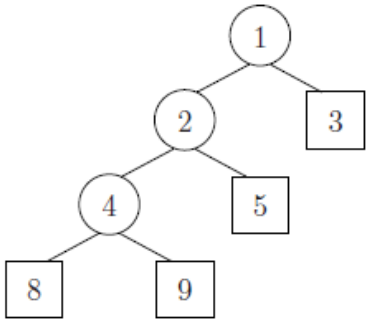
(1) The first idea is pruning trees: instead of stopping growth in progress, we grow an over-large tree and then cut it down. This can be impracticable in terms of computation, since the number of subtrees is usually very large. To solve this problem, Breiman et al. [1] introduced the second idea of *cost − complexity pruning*.

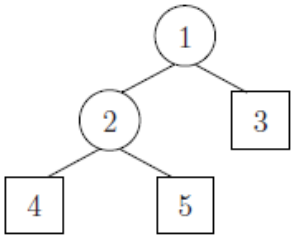
(2) The basic strategy of *cost − complexity pruning* is that we grow a large tree T_0, stopping the splitting process until the number of observations reaching each terminal node is small (say 10), and then getting the large tree pruned upward according to *cost − complexity* measure until we finally cut back to the root node.

Before we introduce the definition of *cost − complexity*, we first give a brief description of the notations that will be used here. We define that a *branch* of T_0 is a tree that has one of the internal nodes of T_0 as its root node, and contains all of the subsequent daughter nodes below that node in T_0. A *subtree* T of T_0 shares the root node of T_0, but may not share all the subsequent branches of T_0. See Figure 3.3. We also define $\widetilde{T}$ to be the set of terminal nodes of T. As before, the terminal nodes are indexed by m, $m = 1, 2, \ldots, b$ and node m represents the region R_m. Let $|\widetilde{T}|$ denotes the number of terminal nodes in T or the size of the tree T ($|\widetilde{T}| = b$), and define the risk of classification trees:

$$R(T) = \sum_{m \in \widetilde{T}} P(m) Q_m(T) \tag{3.11}$$

where $Q_m(T)$ measures the impurity of node m in a classification tree which can be any one in (2.3)-(2.5), and typically, we use the overall misclassification





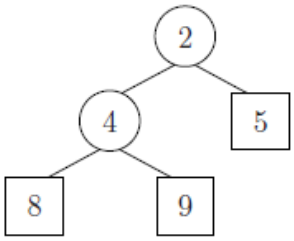


Figure 3.3: A tree, Subtree and Branch

error rate.

We define the *cost − complexity* criterion [1]

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}| \tag{3.12}$$

where $\alpha(\geq 0)$ is the complexity parameter. $R_\alpha(T)$ adds a penalty for complex trees to the impurity $R(T)$. The main idea is that for each α, find the subtree $T_\alpha \subseteq T_0$ which minimizes $R_\alpha(T)$, that is,

$$R_\alpha(T_\alpha) = \min_{T \subseteq T_0} R_\alpha(T) \tag{3.13}$$

The tuning parameter $\alpha \geq 0$ " governs the tradeoff between tree size and its goodness of fit to the data" [3]. If α is small, the penalty for having a large number of terminal nodes is small and T_α will be large. For example, if α is zero, T_0 is very large such that each terminal node only contains one case, then every case is classified correctly with misclassification rate $R(T_0) = 0$. So T_0 minimizes $R_0(T)$. As the value of α (or the penalty per terminal node) increases, the size of tree T_α decreases. So large values of α, smaller subtrees T_α of T_0. For α sufficiently large, the optimal subtree T_α will consist of the root node only. Brieman et al. [1] had proved the following useful results. For details, refer to [1].

Theorem 3.1.1. *If $\alpha_2 > \alpha_1$, the optimal subtree corresponds to α_2 is a subtree of the optimal subtree corresponding to α_1.*

Theorem 3.1.2. *For any value of complexity parameter α, there is a unique*

smallest subtree of T_0 *that minimizes the cost-complexity. Moreover, there exists an increasing sequence* α_k *and a positive integer* K*, such that*

$$-\infty < 0 = \alpha_0 < \alpha_1 < \alpha_2 < \cdots < \alpha_K < +\infty$$

and the corresponding optimal subtrees

$$T_{\alpha_0} \supset T_{\alpha_1} \supset T_{\alpha_2} \supset \ldots \supset T_{\alpha_K} = \{root\ of\ T_0\}$$

where $T_{\alpha_0} \supset T_{\alpha_1}$ means that T_{α_1} is a subtree of T_{α_0}. The sequence $\{T_{\alpha_0}, T_{\alpha_1}, \ldots, T_{\alpha_m}\}$ is also called *nested optimal subtrees*.

According to Theorem 3.1.1 and Theorem 3.1.2, as we increase α from 0 to large enough, the desired sequence of nested trees of decreasing size is obtained, beginning with the initial full tree T_0 and ending with the root tree with no split at all. Each tree T_α in the sequence is the best of all trees of its size with respect to α. It may prune more than one node at a time. For details, refer to [1] and [7] Chapter 7.

(3) In (2), we have got a decreasing sequence of subtrees $T_{\alpha_0} \supset T_{\alpha_1} \supset T_{\alpha_2} \supset \ldots \supset T_{\alpha_K} = \{root\ of\ T_0\}$. To complete the process of choosing our final tree, we need a good way to choose the degree of pruning, that is, select one of these as the final optimal-sized tree. Breiman et al. [1] came out two methods: using an independent test sample and cross-validation. If enough data are available, we can select a random sample from the data set to form a test sample, and the remainder forms the new training sample. The original

over-large T_0 is grown using the new training sample and pruned upward to get the decreasing sequence $T_{\alpha_0} \supset T_{\alpha_1} \supset T_{\alpha_2} \supset \ldots \supset T_{\alpha_K} = \{root\ of\ T_0\}$. For each tree in the sequence, predict the classifications for the test sample, and compute the misclassification error or deviance. Take the misclassification error for instance, for any tree T_{α_k} in the sequence, if we use $N^{(1)}$ to denote the total number of cases in the test sample, and $N_{ij}^{(1)}$ denote the number of class j cases in the test sample whose predicted classification by T_{α_k} is class i. Then the misclassification error for the tree T_{α_k} is

$$R^{ts}(T_{\alpha_k}) = \frac{\sum_i \sum_j N_{ij}^{(1)}}{N^{(1)}}.$$

Thus, for the sequence of trees, we can plot the errors or deviance versus α. This will often have a minimum within the range of trees considered, and we can choose the smallest tree with error or deviance close to the minimum value. One drawback of this method is that usually there are not sufficient data to build good trees based on only a subset of the data. However, this method is computationally efficient and is preferred when the training sample contains a large number of cases.

Unless the sample size of the training sample is quite large, cross-validation is the preferred method. In V-fold cross-validation, we divide the original training sample, denoted by S, by a random selection into V equally sized (as nearly as possible) subsets, S_v, $v = 1, 2, \ldots, V$, each containing the same number of cases. Then we take one part of S_v to form the testing sample and the remaining $V - 1$ parts to form the new training sample. The vth test sample is S_v and the vth new training sample is $S^{(v)}$. Usually, V is taken as 10 (10-fold cross-validation), so each new training sample contains 9/10 of

the data. Suppose the minimal cost-complexity sequence of trees based on the whole training data are $\{T_{\alpha_k}\}_{k=0}^{K}$, and the corresponding minimal cost-complexity trees based on $V-1$ parts of the data are $\{T_{\alpha_k}^{(v)}\}_{k=0}^{K}$. Suppose N_{ij} is the total number of class j cases classified as i by T_{α_k}, and $N_{ij}^{(v)}$ is the number of class j cases in S_v with predicted classification i by $T_{\alpha_k}^{(v)}$. Then

$$N_{ij} = \sum_v N_{ij}^{(v)}.$$

Each case in S appears in one and only one test sample S_v. Thus, the total number of class j cases in all test samples is N_j, which is the number of class j cases in the original training sample S. Then the misclassification error for tree T_{α_k} is

$$R^{CV}(T_{\alpha_k}) = \frac{\sum_i \sum_j N_{ij}}{N},$$

where N is the total number of cases in S. Comparing to the independent test sample method, cross-validation is more computationally expensive, but it uses all cases in the training sample more effectively.

Breiman et al. [1] also suggested the $1-SE$ *rule* to choose the right sized tree.

Definition 3.1.1. *Suppose $T_{\alpha_{k_0}}$ is the tree with minimum misclassification error $R(.)$, that is,*

$$R(T_{\alpha_{k_0}}) = \min_{\alpha_k} R(T_{\alpha_k}).$$

Then the tree selected is $T_{\alpha_{k'}}$, where $\alpha_{k'}$ is the maximum α_k satisfying

$$R(T_{\alpha_{k'}}) \leq R(T_{\alpha_{k_0}}) + SE(R(T_{\alpha_{k_0}}))$$

The motive of this selection rule is that, in most of the examples, the misclassification error estimates $R(T_{\alpha_k})$ as a function of the tree size will have a fairly rapid decrease at the initial. After that there is a long, and flat valley region such that $R(T_{\alpha_k})$ is almost constant except for changing up and down within $\pm$SE range. The standard error of the estimates can calculated for each size of the tree. See Breiman et al. [1] for details.

3.2 Missing Data in CART

In the tree building process, we split data at each node according to the value of some variable. This needs the values to be available. In the decision making process, a test is carried out on some variable at each node. This, again, requires that the value of that variable be available. Thus, for classification trees, missing data are a really big concern and we desire to find optimal solutions to handle missing values. Some treatments of missing data we discussed in the last chapter can be applied on tree-based models, such as procedures based on completely recorded units or imputation methods, and so on. For example, if the number of cases with missing data is less than five percent of the total number of cases, deleting all cases with missing values may be a possible choice. Another option is to replace missing data with imputation values and then apply tree-based procedures on the imputed data. Other than these methods in statistics, tree-based models have some unique and simple ways to deal with missing data. Here, we focus on the CART algorithm and consider the attractive strategy adopted in CART to split cases with missing values.

3.2.1 Choosing A Split

Recall that at each step, the CART algorithm tries to find the predictor and the split rule that gives the maximal reduction in impurity:

$$\Delta Q(s,m) = Q_m(T) - p_{m_L} Q_{m_L}(T) - p_{m_R} Q_{m_R}(T) \tag{3.14}$$

where $Q_m(T)$ is the impurity of the parent node m, p_{m_L} is the probability of a case going to the left daughter node of m, p_{m_R} is the probability of a case going to the right daughter node of m, $Q_{m_L}(T)$ is the impurity of the left daughter node and $Q_{m_R}(T)$ is the impurity of the right daughter node.

To split the parent node m into two daughter nodes, we require predictors. If the values of some predictors are missing, the CART algorithm still select the best split at each step according to Equation 3.14, however, some terms are somewhat modified. Firstly, the impurity indices $Q_{m_L}(T)$ and $Q_{m_R}(T)$ are calculated only over the observations which are not missing. Secondly, the two probabilities p_{m_L} and p_{m_R} are also calculated only over the relevant observations, but they are then adjusted so that they sum to 1. Thus, for each predictor, we find the optimal split that maximizes the quantity $\Delta Q(s,m)$. Then we pick out the best one with the maximum value of $\Delta Q(s,m)$ among the optimal splits.

Once a splitting variable and a split point for it have been decided, what is to be done if the predictor values of some cases are missing? CART use *surrogate* variables to allocate the missing cases to the daughter nodes.

3.2.2 Surrogate Variables

Surrogate splits are chosen to match as well as possible the primary split. The main idea is that if a case has missing value on the splitting variable, we consider other predictors as surrogate variables and rank them according to the measure of agreement. Then assign the case to one of the daughter nodes using the best surrogate variable. If there are missing data on the best surrogate variable, the second best surrogate variable is used instead. And so on. If all surrogate variables of a case are missing, the algorithm has two options: stop or send the case in the majority direction. This can be done through defining control parameters in the algorithm. The measure of agreement is the number of non-missing cases the surrogate split classified the same way as the primary split variable does, possibly after swapping 'left' and 'right' for the surrogate. It can be shown in the following Table 3.1.

Table 3.1: The Measure of Agreement

	Primary Split left	Primary Split right	NA
Surrogate Split left	$N_{(left,left)}$	$N_{(right,left)}$	a
Surrogate Split right	$N_{(left,right)}$	$N_{(right,right)}$	b

Thus, the number of cases going to the same directions, no matter which split is used, is $(N_{(left,left)} + N_{(right,right)})$.

3.3 Implementation in R

All the decision-tree learning algorithms share the same ideology but have different ways to realize it. A simpler and faster realization of CART in Splus or R is via the library section `rpart` (Recursive Partitioning and

Regression Trees) by Terry Therneau and Beth Atkinson [8].

The basic functions needed to grow and prune classification trees:

- `rpart`: fit a '`rpart`' model and grow an over-large tree
- `print`: print a text version of a tree
- `printcp`: display complexity parameter (`cp`) table for fitted '`rpart`' object
- `prune`: prune the over-large tree to the final optimal subtree based on `cp`

The basic steps needed to use `rpart`:

Step 1. Attach the library so that the functions can be found.

```
library(rpart)
```

Step 2. Decide what type of endpoint we have.

- Categorical ⇒ method = 'class'
- Continuous ⇒ method = 'anova'
- Poisson Process/Count ⇒ method = 'poisson'
- Survival ⇒ method = 'exp'

Step 3. Fit the model using the standard Splus modeling language.

```
fit<-rpart(formula, data, method='class'...)
```

Step 4. Print a text version of the tree.

```
print(fit)
```

Step 5. Prune to the optimal tree.

```
fit1<-prune(fit,cp)
```

Step 6. Print a summary which examines each node in depth.

```
summary(fit1)
```

Step 7. Plot a standard version of the plot with some basic information.

```
plot(fit1)
text(fit1,use.n=T)
```

Model options in `rpart`:

`parms`: For classification, the list of optional parameters can contain any of:

- `prior`– the vector of prior probabilities (positive, sum to 1)
- `loss`– the lost matrix (zeros on diagonal, positive off-diagonal elements)
- `split`– the splitting index ('gini' or 'cross-entropy')

`na.action`: The default action deletes the observations for which the response y is missing or *all* of the predictors are missing, but keeps those in which one or more predictors are missing.

`control`: A list of control parameters. The list contains

- `minsplit`: The minimum number of observations that must exist in a node, in order for a split to be attempted. The default is 20.

- `minbucket`: The minimum number of observations in a terminal node. This defaults to `minsplit/3`.

- `xval`: The number of cross-validation to be done. The default number is 10.

- `cp`: Complexity parameter. Any split that does not decrease the overall lack of fit by a factor of '`cp`' is not attempted. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially,the user informs the program that any split which does not improve the fit by '`cp`' will likely be pruned off by cross-validation, and that hence the program need not pursue it.

- `maxsurrogate`: The maximum number of surrogate splits retained at each node in the output. (No surrogate that does worse than "go with the majority" is printed or used). The default is 5.

- `usesurrogate`: How to use surrogates in the splitting process.

  - 0=display only; an observation with a missing value for the primary split rule is not sent further down the tree.

  - 1=use surrogates, in order, to split subjects missing the primary variable; if all surrogates are missing the observation is not split.

  - 2(default)=if all surrogates are missing, then send the observation in the majority direction.

- `surrogatestyle`: Controls the selection of a best surrogate.

  - 0(default)=use the total number of correct classification for a potential surrogate variable.

- 1=use the percent correct, calculated over the non-missing values of the surrogate.

The first option more severely penalizes covariates with a large number of missing values.

Chapter 4

Linear Methods and Extensions for Binary Classification

In this chapter, we will focus on linear methods for binary classification problems, when the response variable is categorical and can only take two possible values. Typical examples of binary responses include disease status (diseased/not diseased) and survival status (dead/alive). Many linear techniques have been developed to discriminate observations into two different groups, for instance, linear regression, linear discriminant analysis, logistic regression and so on.

It has been shown that if the response only has two classes, and we code them -1 and 1, then the intersection of the fitted plane using linear regression with 0 provides a separating hyperplane, which is the same separating hyperplane as given by the linear discriminant analysis, if the prior probabilities for the two classes are both 1/2, and both classes have the same number of elements [4]. Thus, in this chapter, we will only review linear regression and logistic regression. However, a drawback of these two methods is that they

delete too many observations when missingness is encountered. In the last part of this chapter, we will define a weighted-averaging classifier to deal with missing values.

4.1 Linear Regression Model

Linear regression is probably the most widely used regression models. It is simple, easy to understand, easy to interpret and also easy to use. It can deal with not only the categorical outcomes also the continuous response variables.

Suppose we have n observations $\{y_i; x_{i1}, \ldots, x_{ip}\},\ i = 1, \ldots, n$. Mathematically, the linear regression model has the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i,\ i = 1, \ldots, n. \tag{4.1}$$

1. In the above equation, $\beta_0, \ldots, \beta_p$ are unknown parameters;

2. $\epsilon_1, \ldots, \epsilon_n$ are n random error terms, which are purely due to randomness and can not be observed.

We could denote linear regression in matrix form

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix}}_{Y_{n\times 1}} = \underbrace{\begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}}_{X_{n\times(p+1)}} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \cdots \\ \beta_p \end{pmatrix}}_{\beta_{p+1\times 1}} + \underbrace{\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdots \\ \epsilon_n \end{pmatrix}}_{\epsilon_{n\times 1}}$$

or

$$Y = X\beta + \epsilon. \tag{4.2}$$

Several assumptions are made on the error terms, which are called Guass-Markov (GM) assumptions.

- (A1) $E(\epsilon_i) = 0, i = 1, \ldots, n$;

- (A2) $Var(\epsilon_i) = \sigma^2,\ i = 1, \ldots, n$ and $Cov(\epsilon_i, \epsilon_j) = 0,\ 1 \leq i \neq j \leq n$;

- (A3) $\epsilon_i' s$ are normally distributed.

Thus, $\epsilon_1, \ldots, \epsilon_n$ are independent and follow $N(0, \sigma^2)$.

In matrix form, the GM assumptions can be written as

- (A1) $E(\epsilon) = 0_{n\times 1} \Rightarrow E(Y) = X\beta$;

- (A2) $Var(\epsilon) = \sigma^2 I_{n\times n} \Rightarrow Var(Y) = \sigma^2 I_{n\times n}$;

- (A3) ϵ is multivariate normally distributed, which implies that Y is multivariate normally distributed.

In a word, $\epsilon \sim MN(0_{n\times 1}, \sigma^2 I_{n\times n})$ or $Y \sim MN(X\beta, \sigma^2 I_{n\times n})$.

4.1.1 Fitting the Linear Regression Model

- Definition of Least Square Estimate (LSE) The loss function

$$S(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2 = \sum_{i=1}^{n} \epsilon_i^2 = (Y - X\beta)^T (Y - X\beta).$$

The least square estimate of $\beta = (\beta_0, \ldots, \beta_p)^T$, denoted by $\hat{\beta} = (\hat{\beta}_0, \ldots, \hat{\beta}_p)^T$, is to minimize the loss function $S(\beta)$. Mathematically,

$$\hat{\beta} = \arg\min_{\beta} S(\beta) = \arg\min_{\beta} (Y - X\beta)^T (Y - X\beta).$$

- Deriving LSE There are several ways to get the least square estimate for β. The common way for deriving $\hat{\beta}$ is to take the first derivative of $S(\beta)$, set it to 0, and solve the equation to get the estimates.

$$S(\beta) = (Y - X\beta)^T (Y - X\beta) = \beta^T X^T X\beta - 2Y^T X\beta + Y^T Y$$

Take the first derivative of $S(\beta)$ with respect to β:

$$\partial S(\beta)/\partial\beta = 2X^T X\beta - 2X^T Y$$

Set it to 0 and solve the equations

$$\partial S(\beta)/\partial\beta|_{\beta=\hat{\beta}} = 2X^T X\hat{\beta} - 2X^T Y = 0 \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T Y.$$

- Fitted Values $\hat{Y}$ and Its Standard Error

1. Fitted values $\hat{Y}$. The fitted value vector

$$\hat{Y} = X\hat{\beta} = X(X^T X)^{-1} X^T Y = HY.$$

where $H = X(X^T X)^{-1} X^T$

2. Standard error of $\hat{Y}$.

$$se(\hat{Y}) = se(X\hat{\beta}) = \sqrt{X\hat{Var}(\hat{\beta})X^T} = \hat{\sigma}\sqrt{X(X^TX)^{-1}X^T}.$$

Thus, we need to obtain the estimate $\hat{\sigma}$ of σ. An unbiased estimation of σ^2 is

$$\hat{\sigma}^2 = \frac{1}{n-1-p}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n-1-p}Y^T(I-H)Y.$$

In R, we can use $lm()$ to fit the linear regression model.

The properties of $\hat{\sigma}^2$ and $\hat{\beta}$ are given in the following theorem

Theorem 4.1.1. *Under GM Assumptions A1-A3, we have*

1. $\hat{\beta} \sim MN_{p+1}(\beta, \sigma^2(X^TX)^{-1})$;
2. $E(\hat{\sigma}^2) = \sigma^2$ *and* $(n-1-p)\hat{\sigma}^2/\sigma^2 \sim \chi^2_{n-1-p}$;
3. $\hat{\beta}$ *and* $\hat{\sigma}^2$ *are independent.*

Based on the above results, we could construct confidence interval and do hypothesis testing for a single parameter or a linear combination of parameters in linear regression model.

4.2 Logistic Regression Model

Logistic Regression are mainly used to deal with categorical responses, especially the binary outcomes. Over the past decade, the logistic regression

model has become the standard method of analysis in many field, such as biomedical studies, social sciences and marketing.

4.2.1 Model Setup

In any regression problem the key quantity is the mean value of the response variable, given the values of predictors, which is expressed as $E(Y|x)$. In linear regression we assume that this mean may be expressed as a linear equation in x, such as $E(Y|x) = \beta_0 + \beta_1 x$. This expression implies that it is possible for $E(Y|x)$ to take any value as x ranges from $-\infty$ to $+\infty$. However, if the response variable is binary, and the conditional mean lies between 0 and 1, that is $0 \leq E(Y|x) \leq 1$, and the plot of $E(Y|x)$ versus x is as follows:

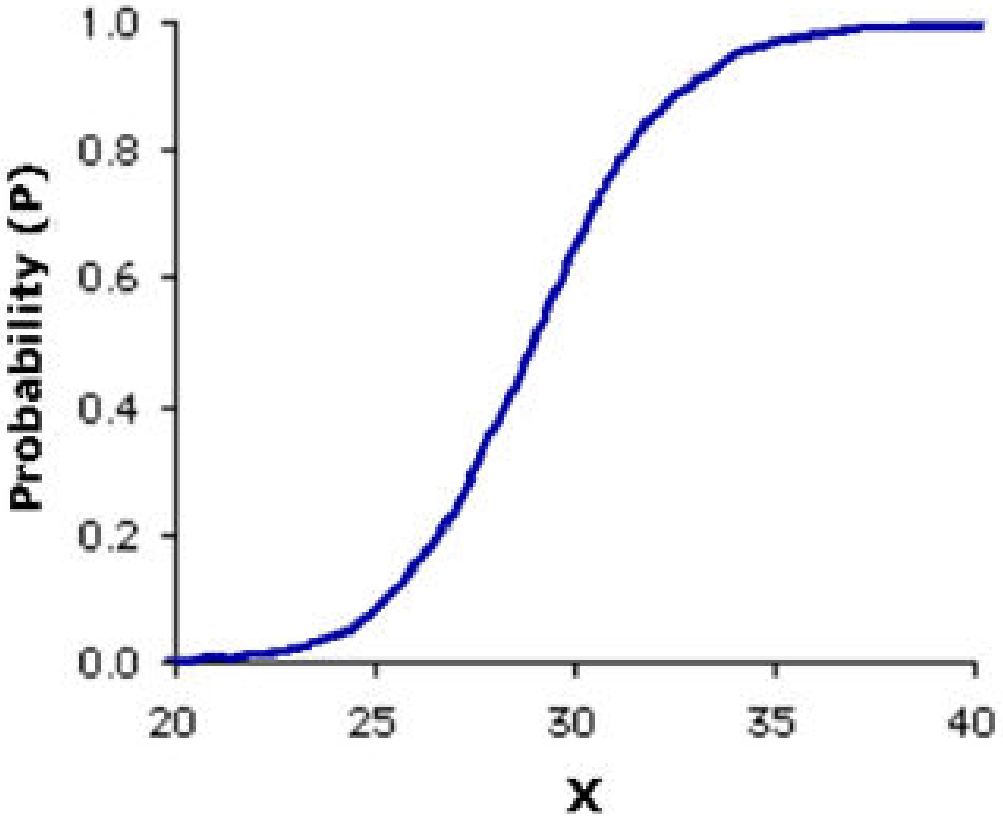


Figure 4.1: Plot of $E(Y|x)$ Versus x

The curve is said to be $S-$*shaped*. In this case, the model we will use is that of the logistic distribution. Because of its extremely flexible and easily used function in mathematics, logistic regression becomes a standard analysis method.

For a binary response variable Y and an explanatory variable X, let $\pi(x) = E(Y|x) = P(Y = 1|X = x) = 1 - P(Y = 0|X = x)$. The logistic regression model is

$$\pi(x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)} \tag{4.3}$$

Equivalently, the log odds, called that *logit*, has the linear relationship

$$logit[\pi(x)] = \log \frac{\pi(x)}{1-\pi(x)} = \alpha + \beta x \tag{4.4}$$

Another important difference between the linear and logistic regression models is the conditional distribution of the response variable. In the linear regression model, we assume that $y = E(Y|x) + \varepsilon$ with ε follows a normal distribution with mean zero and some constant variance. Then, the conditional distribution of Y is also a normal distribution with mean $E(Y|x)$ and constant variance. However, this is not the case when the response variable is binary. In this situation, $y = E(Y|x) + \varepsilon = \pi(x) + \varepsilon$ and if $y = 1$ then $\varepsilon = 1 - \pi(x)$ with probability $\pi(x)$, and if $y = 0$ then $\varepsilon = -\pi(x)$ with probability $1 - \pi(x)$. Therefore, ε has a distribution with zero mean and $\pi(x)(1 - \pi(x))$, so the conditional distribution of the response variable follows a binomial distribution with probability $\pi(x)$.

Like ordinary regression, logistic regression can extend to models with multiple explanatory variables. For instance, the model for $\pi(\mathbf{x})$=$P(Y = 1)$ at values $\mathbf{x}$=$(x_1, \ldots, x_p)$ of p predictors is

$$logit[\pi(\mathbf{x})] = \log \frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})} = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{4.5}$$

The alternative formula, directly specifying $\pi(\mathbf{x})$, is

$$\pi(\mathbf{x}) = \frac{\exp(\alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)}{1 + \exp(\alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)} \tag{4.6}$$

4.2.2 Fitting the Logistic Regression Model

Suppose we have the following observations: $\{y_i, m_i : x_{i1}, \ldots, x_{ip}\}$, $i = 1, 2, \ldots, n$. We assume that

$$y_i \sim Binomial(m_i, \pi_i) \tag{4.7}$$

with

$$logit(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} = x_i^T \beta \tag{4.8}$$

and $y_1, \ldots, y_n$ are independent. Here $x_i = (1, \ldots, x_{i1}, \ldots, x_{ip})^T$, $\beta = (\alpha, \beta_1, \ldots, \beta_p)$. Equivalently, we have

$$\pi_i = \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)} \tag{4.9}$$

With the n observations, our purpose here is to make inference on β. Since we have a parametric model, the MLE is a natural choice we may consider. The likelihood function of β is given as

$$\begin{aligned} L(\beta) &= \prod_{i=1}^{n} Pr(Y_i = y_i) = \prod_{i=1}^{n} \binom{m_i}{y_i} \pi_i^{y_i} (1-\pi_i)^{m_i - y_i} \\ &= \prod_{i=1}^{n} \binom{m_i}{y_i} (\frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)})^{y_i} (\frac{1}{1 + \exp(x_i^T \beta)})^{m_i - y_i} \end{aligned}$$

and the log-likelihood function of β is given as

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{n} \{y_i \log(\pi_i) + (m_i - y_i) \log(1 - \pi_i)\} + \sum_{i=1}^{n} \log \binom{m_i}{y_i} \\ &= \sum_{i=1}^{n} \{y_i \log(\pi_i/(1 - \pi_i)) + (m_i - y_i) \log(1 - \pi_i)\} + \sum_{i=1}^{n} \log \binom{m_i}{y_i} \\ &= \sum_{i=1}^{n} \{y_i x_i^T \beta - m_i \log(1 + \exp(x_i^T \beta))\} + \sum_{i=1}^{n} \log \binom{m_i}{y_i} \end{aligned}$$

The MLE of β is defined as

$$\hat{\beta} = \arg\max L(\beta) = \arg\max l(\beta)$$

The idea of MLE is to find β which makes the observed data most likely.

Here are some useful notations.

1. Score function: $S(\beta) = l'(\beta)$, the first derivative of log-likelihood function with respect to β. Note that

$$S(\hat{\beta}) = 0 \Rightarrow \hat{\beta}.$$

2. Observed Fisher information function: $I(\beta) = -l''(\beta) = -S'(\beta)$, minus the 2nd derivative of the log-likelihood function.

3. Expected Fisher information function: $J(\beta) = E\{I(\beta)\}$, the expectation of the observed Fisher information function.

To find $\hat{\beta}$, we first find the score function:

$$S(\beta) = \sum_{i=1}^{n} \{y_i x_i - m_i \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)} x_i\} = \sum_{i=1}^{n} (y_i - m_i \pi_i) x_i$$

and then solve the equation $S(\hat{\beta}) = 0$ to get $\hat{\beta}$. Unfortunately, we can not find a close form for $\hat{\beta}$. To solve the problem, we apply some numeric algorithms: Newton Raphson algorithm and Fisher Scoring algorithm. The observed Fisher information function

$$I(\beta) = -\frac{\partial S(\beta)}{\partial \beta^T} = \sum_{i=1}^{n} m_i \{\frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)^2} x_i x_i^T\} = \sum_{i=1}^{n} m_i \pi_i (1-\pi_i) x_i x_i^T.$$

Note that $J(\beta) = I(\beta) = Var\{S(\beta)\}$. The iterative way of Newton Raphson and Fisher Scoring algorithm to calculate $\hat{\beta}$

1. Choose a starting value for $\beta^{(0)}$.

2. Update the value of β by

$$\beta^{(r+1)} = \beta^{(r)} + I(\beta^{(r)})^{-1} S(\beta^{(r)}) = \beta^{(r)} + J(\beta^{(r)})^{-1} S(\beta^{(r)}), i = 0, 1, 2, \ldots.$$

3. Iterate until $|\beta^{(r+1)} - \beta^{(r)}|$ is quite small.

Note that if $\beta^{(r+1)} - \beta^{(r)} = 0$, then $S(\beta^{(r)}) = 0$ and we find $\hat{\beta}$. However, in any computer, it is hard to get $\beta^{(r+1)} = \beta^{(r)}$. We can stop the algorithm if $\beta^{(r+1)} - \beta^{(r)}$ is very close to 0. In R, we can use $glm()$ to fit the logistic regression model.

4.2.3 Statistical Inference

Theorem 4.2.1. *When sample size is large, under certain conditions, approximately*

$$\hat{\beta} - \beta \sim MN(0, I^{(-1)}(\hat{\beta})) \text{ or } \hat{\beta} - \beta \sim MN(0, J^{(-1)}(\hat{\beta})).$$

Thus, the ML estimators $\hat{\beta}$ have a large-sample normal distribution with estimated covariance matrix

$$I^{(-1)}(\hat{\beta}) = \sum_{i=1}^{n} m_i \hat{\pi}_i (1 - \hat{\pi}_i) x_i x_i^T$$

with

$$\hat{\pi}_i = \frac{\exp(x_i^T \hat{\beta})}{1 + \exp(x_i^T \hat{\beta})}.$$

The fitted value of $P(Y = 1|X_i = x_i)$is

$$\hat{P(Y = 1|X_i = x_i)} = \hat{\pi}_i = \frac{\exp(x_i^T \hat{\beta})}{1 + \exp(x_i^T \hat{\beta})}.$$

To calculate the standard error of the predicted probability, we have to use the delta method.

Theorem 4.2.2. *For some sequence of random variables X_n satisfying*

$$\sqrt{n}[X_n - \theta] \xrightarrow{D} N(0, \Sigma),$$

where n is the number of observations and Σ is a covariance matrix. Then

$$\sqrt{n}[g(X_n) - g(\theta)] \xrightarrow{D} N(0, \nabla g(\theta)^T \cdot \Sigma \cdot \nabla g(\theta))$$

for any function g satisfying the property that $\nabla g(\theta)$ exists and is non-zero valued. Especially, in univariate terms, if

$$\sqrt{n}[X_n - \theta] \xrightarrow{D} N(0, \sigma^2),$$

then

$$\sqrt{n}[g(X_n) - g(\theta)] \xrightarrow{D} N(0, \sigma^2[g'(\theta)]^2)$$

Thus, the function we used in logistic regression is $g(t) = (1+exp(-t))^{-1}$. Applying the delta method, we get

$$\hat{\pi}_i = g(x_i^T \hat{\beta}) = g(linear\ combination)$$

$$se(\hat{P(Y = 1|X_i = x_i)}) = g'(x_i^T \hat{\beta}) * se(x_i^T \hat{\beta}) = \hat{\pi}_i(1 - \hat{\pi}_i) * \sqrt{x_i^T [I^{(-1)}(\hat{\beta})] x_i}$$

Moreover, we could construct the Confidence Interval and conduct hypothesis testing for a single parameter or a linear combination of parameters. The output of $glm()$ automatically produce the test statistic for the hypothesis $H_0 : \beta_j = 0$ and the corresponding p-value based on $N(0, 1)$ distribution. If we test the hypothesis that some of the coefficients equal to 0, then we can use likelihood ratio test with the approximate distribution χ^2_{df}. In $glm()$ output, it does not provide the log-likelihood but the *deviance*. As mentioned,

$$l(\beta) = \sum_{i=1}^{n} \{y_i \log(\pi_i) + (m_i - y_i) \log(1 - \pi_i)\} + \sum_{i=1}^{n} \log \binom{m_i}{y_i}.$$

The deviance of a model is defined to be

$$D(\hat{\beta}) = 2 \sum_{i=1}^{n} \{y_i \log(\frac{y_i/m_i}{\hat{\pi}_i}) + (m_i - y_i) \log(\frac{(m_i - y_i)/m_i}{1 - \hat{\pi}_i})\},$$

which is two times log-likelihood difference between the saturated model and the given model. With the deviance statistics for two nested models, the

likelihood ratio test statistic

$$R_n = D(\hat{\beta_{H_0}}) - D(\hat{\beta_{H_A}}).$$

4.2.4 Model/Variable Selection

When there are missing fields in data, the default option in linear and logistic regression algorithms is to discard cases with missing values in covariates. This will definitely reduce the sample size and lose information. In such situation, one plausible strategy is to do model/variable selection according to some criteria. In this way, the model will become simpler and smaller and more units in the sample will be included in the analysis. We will cover one procedure: all-subset selection and two criteria for selecting candidate models: AIC and BIC.

All-Subset Procedure

In some situations, there are many covariates in data. The true model, or the model only containing important variables is unknown. Suppose there are p covariates, then the number of important variables is less than or equal to p. The main idea of all-subset procedure is that all the possible models with the number of covariates less than or equal to p will be considered and compared. Thus, we need to compare

$$\binom{p}{1} + \binom{p}{2} + \cdots + \binom{p}{p} = 2^p - 1$$

models to obtain the "best" model. For example, if $p = 7$, there are $2^7 - 1 = 127$ candidate models. Then we pick the "best" one according to some criteria.

Criteria: AIC and BIC

For model with q covariates, the value of AIC is defined as

$$AIC = -2 \max \log(likelihood) + 2(q+1)$$

and the value of BIC is defined as

$$BIC = -2 \max \log(likelihood) + \log(n)(q+1).$$

After simplified, the AIC and BIC reduce to the following formulas:

$$AIC = n \log(2deviance) + n + 2(q+1)$$

$$BIC = n \log(2deviance) + n + \log(n)(q+1).$$

where *deviance* is the deviance of model. Among all the candidate models, AIC and BIC aims to select the model with minimum value of AIC or BIC.

In $glm()$ output, the value of AIC is directly provided. In $lm()$ output, the deviance of model is provided and the value of AIC or BIC is calculated according to the formula.

4.3 Weighted-Averaging Classifier

Both linear and logistic regression are modeled on some subset of covariates. If there are missing values on the covariates, the default approach in *lm* and *glm* is just to ignore the units and base the analysis on complete cases only. This will lead to loss of information, lack of efficiency due to dis-

carding the incomplete cases and biases in estimates when data are missing in a systematic way. Even a smaller model can be selected according to AIC or BIC criteria, we believe too many observations have been deleted in these methods. Furthermore, when it comes to classify any new instance, the conventional regression approaches would be unapplicable if some missing values exist in the covariates used in the model.

A natural way to make a better use of data with missingness is to consider all-subset possible models. For example, suppose we regress the response Y on the important covariates $X = (X_1, X_2, \ldots, X_{10})$, which is also an AIC optimal model. Assume a subject (in the training data or in the validation data) has missing values on the first 5 covariates $(x_1, \ldots, x_5)$. If we only consider the model with all of the ten variables, then this subject would either be eliminated from the analysis (if it is in the training data) or could not be classified (if it is in the validation data), both of which would sacrifice its information on the left 5 variables $(x_6, \ldots, x_{10})$. However, if we consider all $2^{10} - 1$ submodels, then the subject would be either included to fit some submodels or its prediction would be obtainable based. In this sense, the available information of this case has been well used instead of being deleted.

We describe the method we developed here, weighted-averaging classifier, in a formal way. Weighted-averaging classifier fits all-subset possible models (except the model only having intercept) and obtain the prediction of an instance from each submodel. The final prediction of this instance is defined as weighted averaging the available prediction values. Non-available predictions are excluded. Since standard error is a common measurement of the accuracy of a prediction and the smaller the standard errors, the more reliable the predictions. Therefore, weights are chosen to be functions inverse

proportional to standard errors, such as $f(se) = 1/(se)^2$, $f(se) = 1/(se)$, or $f(se) = 1/\sqrt{se}$. Predictions with smaller standard errors will be assigned larger weights. The performance of these different functions will be examined and compared in the empirical studies of next chapter.

Suppose the original full model include p covariates $x_1, \ldots, x_p$. The principal procedures to realize the weighted-averaging classifier are described below:

- **Step 1:** Fit all $2^p - 1$ linear regression or logistic regression submodels on the training data;

- **Step 2:** Predict a new instance on each model. Denote the $(2^p - 1) \times 1$ prediction vector by $\mathbf{prd} = (prd_1, \ldots, prd_{(2^p-1)})^T$. Some of the elements in **prd** would be NA, if the instance has missing values on some explanatory variables. This will not influence the analysis since we only consider the available prediction values. Denote the vector with non-NA predictions by $\widetilde{\mathbf{prd}} = (prd_{i_1}, \ldots, prd_{i_k})^T$. Thus, as long as the instance has at least one non-missing dependent variables, the procedure continues to the next step. The extreme case is that if all covariates of the instance are missing, then we could do nothing since no information of the instance is konwn.

- **Step 3:** The corresponding standard error of each prediction forms a $(2^p - 1) \times 1$ vector $\mathbf{se} = (se_1, \ldots, se_{(2^p-1)})^T$. If the prediction is NA, then its standard error is also NA, which will not be taken into account. Suppose the non-NA standard error vector is denoted as $\widetilde{\mathbf{se}} =$

$(se_{i_1}, \ldots, se_{i_k})^T$. Compute the $k \times 1$ weight vector

$$\mathbf{W} = f(\widetilde{\mathbf{se}})/(\sum_{j=1}^{k} f(se_{i_j})) = (W_1, \ldots, W_k)^T.$$

For example, if $f(x) = 1/x$, then

$$\mathbf{W} = (1/se_{i_1}/\sum_{j=1}^{k}(1/se_{i_j}), \ldots, 1/se_{i_k}/\sum_{j=1}^{k}(1/se_{i_j}))^T$$

.

- **Step 4:** Compute the final prediction of the instance using the following formula:

$$prd_{final} = \sum_{j=1}^{k}(W_j \cdot prd_{i_j}) = \mathbf{W} \bullet \widetilde{\mathbf{prd}}$$

.

- **Step 5:** Determine the prediction class. For linear regression, we classify $prd_{final} < 0$ to the class coded as -1 and $prd_{final} > 0$ to the class coded as 1. For logistic regression, we classify $prd_{final} < 0.5$ to the class coded as 0 and $prd_{final} > 0.5$ to the class coded as 1.

Chapter 5

Comparison of Methods and Empirical Studies

Five methods of dealing with missingness in binary classification problems, linear and logistic regression with all important covariates, smaller models selected according to the AIC criteria, nonparametric bootstrap method to impute missing values, surrogate split in Classification and Regression Trees (CART) and weighted-averaging linear and logistic classifiers, have been discussed in the last three chapters. In this chapter, we will compare these methods on several data sets regarding to misclassification error.

5.1 Data Sets

The following data sets will be used:

Pima.tr, Pima.te, Pima.tr2

Skull

5.1.1 Data of Diabetes

The first three data sets of *diabetes*, "Pima.tr", "Pima.te", and "Pima.tr2" are available in the "MASS" library under R. These data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases. A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to World Health Organization criteria.

The training set "Pima.tr" contains 200 subjects, and the testing data "Pima.te" contains additional 332 subjects, neither of which has missing values. "Pima.tr2" contains "Pima.tr" plus 100 subjects with missing values in the explanatory variables. The variables in these three data frames contain:

- "npreg": number of pregnancies.
- "glu": plasma glucose concentration in an oral glucose tolerance test.
- "bp": diastolic blood pressure (mm Hg).
- "skin": triceps skin fold thickness (mm).
- "bmi": body mass index (weight in kg/(height in m)^2).
- "ped": diabetes pedigree function.
- "age": age in years.
- "type": "Yes" or "No", for diabetic according to WHO criteria. It is the response variable.

5.1.2 Fossil Skull Data

The data of skull comes from Henke's database of fossil skulls in 1989 [11]. The variables codes are given in the Martin-Saller ([6]) system used by Henke. Another system is defined by Howells 1973 ([12]). Both are listed below:

- "M1/H-GOL": Maximum Cranial Length.
- "M8/H-XCD": Maximum Cranial Breadth.
- "M9/H-FMB": Minimum Frontal Breadth.
- "M10/H-XFB": Maximum Frontal Breadth.
- "M17/H-BBH": Basion-bregma Height.
- "M44/H-EKB": Biorbital Breadth.
- "M45/H-ZYB": Bizygomatic Breadth.
- "M46/H-ZMB": Bimaxilla Breadth.
- "M47": Facial Height (Nasion-gnathion).
- "M48": Upper Facial Height(Nasion-alveolare).
- "M49": Posterior Interorbital Breath.
- "M50": Anterior Interorbital Breath.
- "M51a": Orbital Breath (left).
- "M52": Orbital Height.
- "M54/H-NLB": Nasal Breadth.

- "M55/H-NLH": Nasal Height.
- "M62": Palatal length.
- "M63/H-MAB": Palatal Breadth.
- "M66": Bigonial Breadth.
- "M69": Mental Height.
- "M70": Height of Mandibular Ramus.
- "M71": Breadth of Mandibular Ramus.

The sex in Henke's database was determined according to [11]. Many measurements in Henke's collection are not available. The number of completed cases with all the above variables non-missing is less than 5. Therefore, missing values is the biggest concern in the fossil skull data set. To get a reasonable number of complete training data set, we can only resort to several variables in the data. The original recommendations of Henke [11] are M1, M17, M45 or M1, M8, M45 and M52. In [10] and [9], the variables actually selected are M1 and M45 since the discriminatory value of M8 turned out to be poor and on the other hand, inclusion of M17 and/or M52 would undesirably downsize the sample. The choice of M1, M45 will result in the largest sample size with available Henke's data. However, this selection was too simple to definitely lose lots of information in the data. Thus, after compromising number of associated variables with the completed sample size, we choose Henke's recommendation to include M1, M8, M17, M45 and M52 as predictions in the models.

5.1.3 Missing Values within the Data Sets

- "Pima.tr2" contains 100 subjects (100/300=33.3%) with missing values in the explanatory variables. 10-fold cross validation is applied to "Pima.tr2" and misclassification error is computed.

- In the training data "Pima.tr", there are no missing values. In order to compare the abilities to deal with missing values, various proportions, 0%, 5%, 10%, 14.3%, 25%, 30% of missingness are randomly generated in "Pima.tr". Apply the 5 methods to the incomplete "Pima.tr" and calculate misclassification error on the testing data "Pima.te" without missingness. Repeat this process 100 times and average error rates.

- In the fossil skull data, based on variables M1, M8, M17, M45 and M52, the data contains a total number of 397 subjects and 250 subjects (250/397=62.97%) have missing values in one of the five predictors. 11-fold cross validation is applied and misclassification error is calculated.

The following methods are performed and compared for data "Pima.tr2", "Pima.tr" and "Pima.te".

- Logistic and Linear regression with all covariates.

- Logistic and Linear regression with few covariates. A smaller model is selected according to AIC criteria.

- Logistic and Linear regression with Bootstrap imputation.

- Classification and Regression Trees (CART).

- Weighted-averaging Logistic and Linear classifiers with weights $1/se^2$, $1/se$ and $1/\sqrt{se}$.

The following analysis are carried out for the fossil skull data.

- Logistic and Linear regression with M1 and M45 only.
- Logistic and Linear regression with M1, M8, M17, M45 and M52.
- Logistic and Linear regression with few covariates (selected from M1, M8, M17, M45 and M52). A smaller model is selected according to the AIC criteria.
- Logistic and Linear regression with Bootstrap imputation.
- Classification and Regression Trees (CART).
- Weighted-averaging Logistic and Linear classifiers with weights $1/se^2$, $1/se$ and $1/\sqrt{se}$.

5.2 Results of Diabetes Data

5.2.1 Results of "Pima.tr2"

The following tables summarize the results of five different methods mentioned above. CART1 stands for the classification and regression trees based on minimum cross-validation error criteria and CART2 means the CART algorithm based on 1-SE rule. 10-fold cross-validation is applied to "Pima.tr2".

Table 5.1: Methods Comparison on "Pima.tr2"

	All Covar	AIC	Bootstrap
Linear(NO.out of 300)	0.245(200)	0.2239(201)	0.2733(300)
Logistic(NO.out of 300)	0.24(200)	0.2338(201)	0.27(300)

Table 5.2: Methods Comparison on "Pima.tr2"

	CART1	CART2
Misclassification Error(NO.out of 300)	0.29(300)	0.303(300)

Table 5.3: Methods Comparison on "Pima.tr2"

Weighted-Averaging	$f = 1/se^2$	$1/se$	$1/\sqrt{se}$
Linear(NO.out of 300)	0.2933(300)	0.2667(300)	0.2633(300)
Logistic(NO.out of 300)	0.28(300)	0.26(300)	0.2567(300)

The "NO. out of 300" in the bracket of each table is the number of cases tested in the 10-fold cross-validation process.

The results based on "Pima.tr2" show that the weight $1/\sqrt{se}$ achieved to have the smallest misclassification errors both for linear and logistic regression. Furthermore, the linear and logistic regression with all covariates or with AIC criteria have smaller errors than other methods. However, we should notice that only around 200 out of 300 cases have been tested since 100 cases include missing values on some explanatory variables. The CART, bootstrap and weighted-averaging methods did test all the cases and weighted-averaging classifiers have a better performance.

CART1 and CART2 use *surrogate* variables by default, which is discussed in Section 3.3. To see the splitting process more clearly, the R code and corresponding output are as follows:

```
> library(rpart)
> library(MASS)
```

```
> set.seed(1204)
> fit<-rpart(type~.,data=Pima.tr2,cp=0.001)
> printcp(fit)
Classification tree:
rpart(formula = type ~ ., data = Pima.tr2, cp = 0.001)

Variables actually used in tree construction:
[1] age bmi glu ped
Root node error: 106/300 = 0.35333
n= 300

        CP nsplit rel error  xerror     xstd
1 0.235849      0   1.00000 1.00000 0.078107
2 0.132075      1   0.76415 0.86792 0.075346
3 0.025157      2   0.63208 0.71698 0.071066
4 0.015723      5   0.55660 0.77358 0.072823
5 0.009434      9   0.47170 0.75472 0.072259
6 0.001000     11   0.45283 0.79245 0.073367
> plotcp(fit)
```

The last two columns in the `cp` table are somewhat important.

- The table is printed from the smallest tree (no splits) to the largest one (11 splits).
- The number of splits is listed, rather than the number of terminal nodes. The number of terminal nodes (tree size) is always 1+ the number of splits.

- The columns `xerror` and `xstd` are the scaled misclassification error and its standard error of the cross-validation, respectively. They are random, depending on the random partition used in the 10-fold cross-validation that has been computed within `rpart`.

- For easier reading, the error columns have been scaled so that the first node has an error of 1. Since in this example, the model with no splits must make 106/300 misclassifications, multiply columns 3-5 by 106 to get a result in terms of absolute error. Computations are done on the absolute error scale, and printed on relative scale. The complexity parameter column (CP) has been similarly scaled.

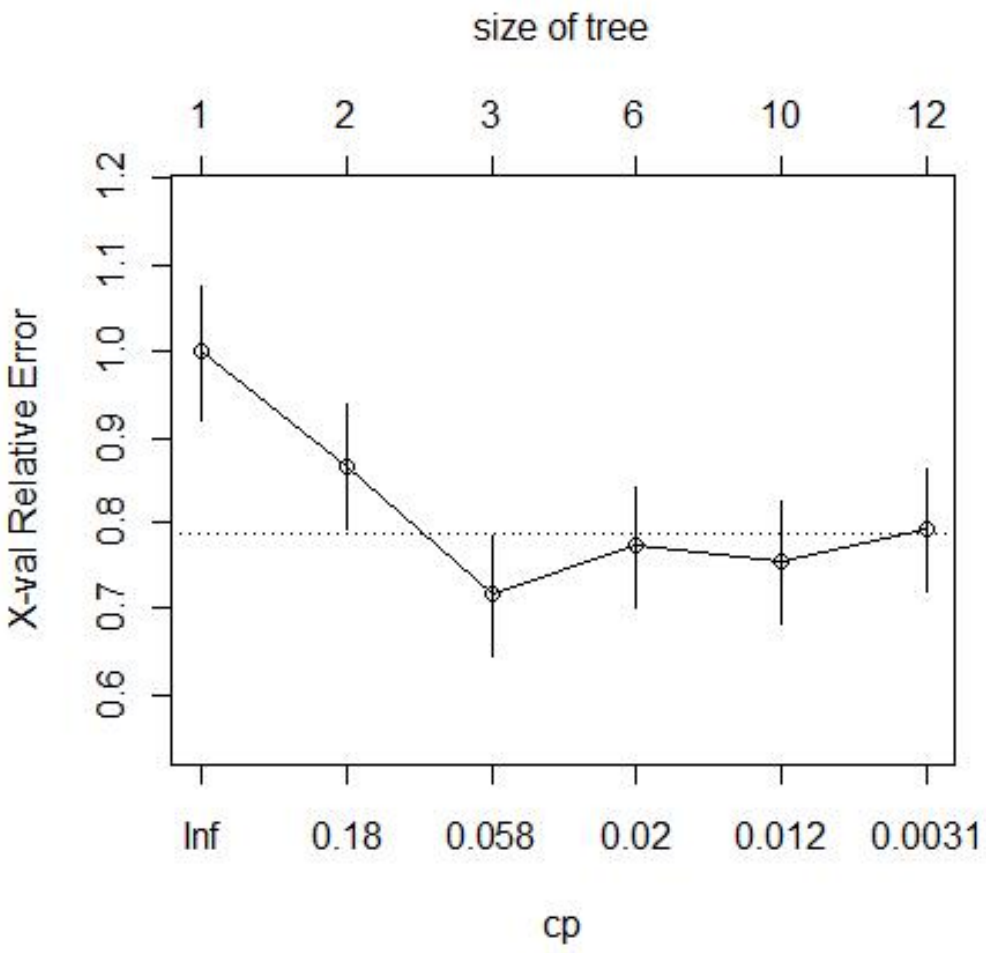


Figure 5.1: Plot of Complexity Parameter

The complexity parameter (as plotted in Figure 5.1) may then be chosen to minimize `xerror`. An alternative procedure is to use the 1-SE rule, the largest value with `xerror` within one standard deviation of the minimum. In this case the 1-SE rule gives 0.71698+0.071066=0.788046, so we choose line 3, a tree with 2 splits and hence size 3. We can examine the pruned tree (Figure 5.2)

```
> fit1<-prune(fit,cp=0.03)
```

```
> print(fit1,digits=3)
n= 300
node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 300 106 No (0.647 0.353)
  2) glu< 128 177  32 No (0.819 0.181) *
  3) glu>=128 123  49 Yes (0.398 0.602)
    6) bmi< 28.8 28   7 No (0.750 0.250) *
    7) bmi>=28.8 95  28 Yes (0.295 0.705) *
> plot(fit1,branch=0.3,uniform=T)
> text(fit1,digits=3)
```

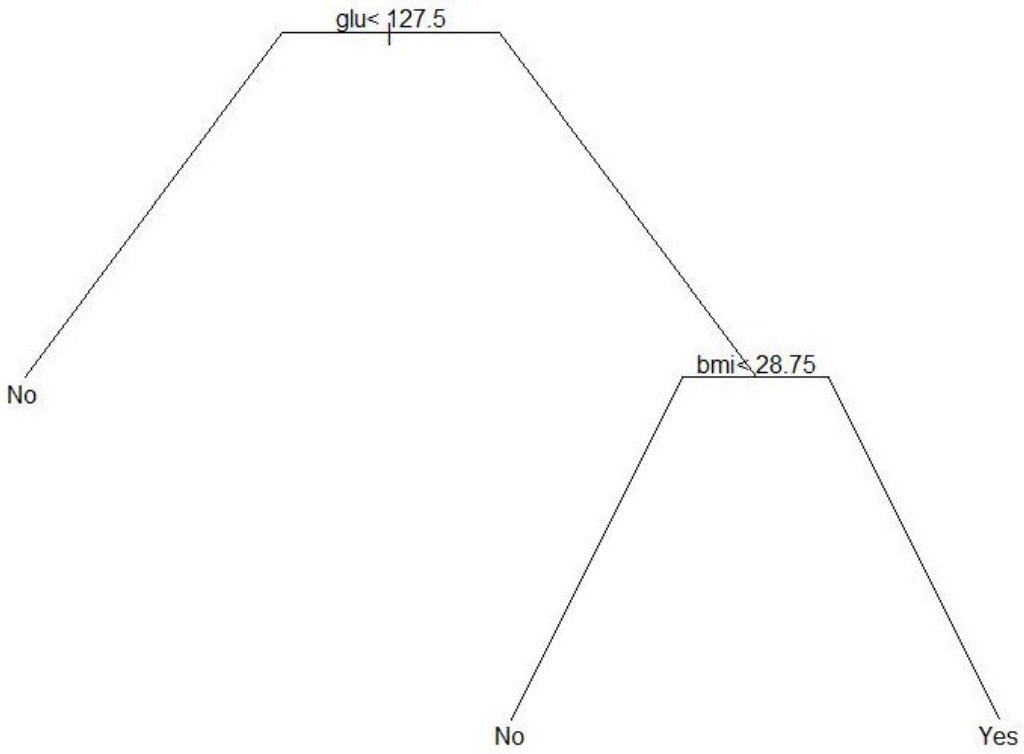


Figure 5.2: Plot of Pruned Tree

The `summary` method, `summary.rpart`, produce voluminous output.

```
> summary(fit1)

......

Node number 1: 300 observations,    complexity param=0.2358491
  predicted class=No   expected loss=0.3533333
    class counts:   194   106
   probabilities: 0.647 0.353
  left son=2 (177 obs) right son=3 (123 obs)
  Primary splits:
    glu   < 127.5  to the left, improve=25.704610, (0 missing)
    bmi   < 27.35  to the left, improve=16.586400, (3 missing)
    age   < 28.5   to the left, improve=14.360940, (0 missing)
    npreg < 6.5    to the left, improve= 9.263027, (0 missing)
    skin  < 22.5   to the left, improve= 7.920797, (98 missing)
  Surrogate splits:
    age   < 32.5   to the left, agree=0.640, adj=0.122,(0 split)
    npreg < 6.5    to the left, agree=0.623, adj=0.081,(0 split)
    bp    < 77     to the left, agree=0.623, adj=0.081,(0 split)
    bmi   < 39.15  to the left, agree=0.617, adj=0.065,(0 split)
    ped   < 1.172  to the left, agree=0.603, adj=0.033,(0 split)

Node number 2: 177 observations
  predicted class=No   expected loss=0.180791
    class counts:   145    32
   probabilities: 0.819 0.181
```

```
Node number 3: 123 observations,    complexity param=0.1320755
  predicted class=Yes  expected loss=0.398374
    class counts:    49    74
   probabilities: 0.398 0.602
  left son=6 (28 obs) right son=7 (95 obs)
  Primary splits:
    bmi  < 28.75  to the left, improve=9.240146, (1 missing)
    glu  < 157.5  to the left, improve=6.373984, (0 missing)
    ped  < 0.3085 to the left, improve=6.281245, (0 missing)
    skin < 20     to the left, improve=4.557608, (42 missing)
    age  < 27.5   to the left, improve=2.139111, (0 missing)
  Surrogate splits:
    age < 21.5   to the left, agree=0.779, adj=0.036, (1 split)

Node number 6: 28 observations
  predicted class=No   expected loss=0.25
    class counts:    21     7
   probabilities: 0.750 0.250

Node number 7: 95 observations
  predicted class=Yes  expected loss=0.2947368
    class counts:    28    67
   probabilities: 0.295 0.705
```

- There are 194 non-diabetes (class 1) and 106 diabetes (class 2), so the

first node has an expected loss of 106/300=0.3533333.

- The improvement is n times the change in impurity index. In this example, the largest improvement is for the variable `glu`, with an improvement of 25.704610. The next best choice is `bmi`, with an improvement of 16.586400.

- For node 3, `bmi` is missing for 1 observation. (5+90)/122=0.779 of the observations that have both `age` and `bmi` agree at their respective splits, thus `age` is chosen as the best surrogate for `bmi`. The `adj` indicates how much is gained over simply choosing a "go with the majority" rule (=94/122). `adj` is calculated as (95/122-94/122)/(1-94/122)=0.036.

Table 5.4: The Agreement

	bmi<28.75	bmi>28.75	NA
age<21.5	5	4	0
age>21.5	23	90	1

5.2.2 Results of "Pima.tr" and "Pima.te"

Main steps of analyzing algorithms:

1. Randomly generate m missing fields out of 1400($= 200 \times 7$) in "Pima.tr". 200 is the number of cases in "Pima.tr" and 7 is the number of variables (other than the response variable). Thus, in total, there are 1400 fields in the data. The proportions of missingness ($100(m/1400)\%$) are taken 0%, 5%, 10%, 14.3%, 25% and 30%, respectively.

2. Apply each method to the new incomplete training data "Pima.tr" and calculate misclassification errors for the testing data "Pima.te" without missingness.

3. Repeat the above steps 100 times. Average the 100 errors as the final results. The corresponding standard errors are also obtained.

Table 5.5: Methods Comparison on "Pima.tr" and "Pima.te"(Logistic)

%(Ave.ComCase)	0%	5%(139.3)	10%(96.51)	14.3%(67.78)	25%(26.35)	30%(11.27)
All Covar(se)	0.1988	0.2039(0.0099)	0.2171(0.0152)	0.2253(0.0193)	0.2878(0.0504)	0.3678(0.0942)
AIC	0.1988	0.2069(0.0101)	0.2177(0.0138)	0.2254(0.0193)	0.2822(0.0491)	0.3249(0.0724)
Bootstrap	0.1988	0.2017(0.0076)	0.2054(0.0074)	0.2075(0.0083)	0.2156(0.0102)	0.2208(0.0116)
CART1	0.2439	0.2543(0.0125)	0.2592(0.0167)	0.2619(0.0255)	0.2686(0.0278)	0.2737(0.0331)
CART2	0.2711	0.2598(0.0146)	0.2636(0.0231)	0.2666(0.0303)	0.2764(0.0367)	0.2826(0.0388)
Wgt-ave-$\frac{1}{se^2}$	0.2379	0.2360(0.0102)	0.2362(0.0118)	0.2287(0.0151)	0.2661(0.0487)	0.3166(0.0553)
Wgt-ave-$\frac{1}{se}$	0.2169	0.2218(0.0104)	0.2253(0.0108)	0.2273(0.0133)	0.2610(0.0505)	0.3054(0.0572)
Wgt-ave-$\frac{1}{\sqrt{se}}$	0.2018	0.2144(0.0077)	0.2205(0.0089)	0.2218(0.0118)	0.2576(0.0512)	0.3001(0.0569)

Table 5.6: Methods Comparison on "Pima.tr" and "Pima.te"(Linear)

%(Ave.ComCase)	0%	5%(139.3)	10%(96.51)	14.3%(67.78)	25%(26.35)	30%(11.27)
All Covar(se)	0.2018	0.2057(0.0110)	0.2175(0.0164)	0.2248(0.0189)	0.2638(0.0331)	0.3564(0.0894)
AIC	0.2018	0.2064(0.0110)	0.2179(0.0158)	0.225(0.0193)	0.2634(0.0333)	0.3556(0.0879)
Bootstrap	0.2018	0.2060(0.0076)	0.2080(0.0083)	0.2093(0.0102)	0.2146(0.0100)	0.2196(0.0095)
CART1	0.2439	0.2543(0.0125)	0.2592(0.0167)	0.2619(0.0255)	0.2686(0.0278)	0.2737(0.0331)
CART2	0.2711	0.2598(0.0146)	0.2636(0.0231)	0.2666(0.0303)	0.2764(0.0367)	0.2826(0.0388)
Wgt-ave-$\frac{1}{se^2}$	0.2469	0.2496(0.0110)	0.2528(0.0129)	0.2579(0.0178)	0.2627(0.0229)	0.2677(0.0238)
Wgt-ave-$\frac{1}{se}$	0.2139	0.2277(0.0114)	0.2305(0.0118)	0.2349(0.0153)	0.2414(0.0205)	0.2447(0.0215)
Wgt-ave-$\frac{1}{\sqrt{se}}$	0.2038	0.2191(0.0101)	0.2254(0.0108)	0.2269(0.0143)	0.2346(0.0195)	0.2396(0.0212)

The "Ave.ComCase" in the bracket of each table is the average number of complete cases of the 100 generated incomplete data "Pima.tr". "se" is the standard error of the 100 misclassification errors for each proportion.

Logistic regression and linear regression were used in Table 5.5 and Table 5.6, respectively.

Methods Comparison:

1. The last three rows in both tables are the weighted-averaging logistic or linear classifiers with three different weights. After comparison, we find that the weight $1/\sqrt{se}$ achieves to have the smallest misclassification errors and standard errors. Thus, $1/\sqrt{se}$ would be the optimal weight we should choose. This is consistent with the results of "Pima.tr2" in the last section.

2. If there are no missing values in "Pima.tr", that is, the second column in both tables, model with all covariates, with AIC criteria and model with bootstrap imputation reduce to the same and they have smaller misclassification errors 0.1988. $0.1988 * 200 = 66$ out of 332 cases are misclassified while the weighted-averaging logistic classifier with weight $1/\sqrt{se}$ misclassifies $66.9976 = 0.2018 * 332$ cases and the corresponding linear classifier partitioned $67.6616 = 0.2038 * 332$ cases into the wrong groups. The differences are no more than two cases. Thus, the weighted-averaging classifiers are good competitors to conventional models with all covariates. The latter is considered as the perfect situation since there is no missing values at all and all variables are modeled. As the proportion of missingness increases, the former does even better. If missingness does exist, a smaller model chosen according to AIC criteria instead of including all covariates would be more reasonable. It has smaller errors when the missingness percentage is below 14.3%.

3. In classification problems, the objective is to discriminate observations into different groups. If some new instance has missing values in explanatory variables, the first two approaches in the tables might not be applicable. From this point of view, bootstrap imputation, CART and

weighted-averaging classifiers would be better choice. After comparison of misclassification errors, the three methods are ranked as follows, bootstrap imputation, weighted-averaging classifier with $1/\sqrt{se}$, CART with minimum validation error, and CART with 1-SE rule.

4. Bootstrap imputation had smaller errors in this analysis which is contradictory to the results of "Pima.tr2" in the last section. The difference is that the testing data we currently used is "Pima.te", which does not contain any missing values. However, in "Pima.tr2", we use 10-fold cross validation. There might be missing values in the validation sets. What was done is that if the validation set has missing field, then pool the bootstrap training data and validation set together to replace the missing place by median if the variable is numerical or replace it by mode if the variable is categorical. In this way, both the training and validation data are complete. Thus, models with bootstrap imputation is able to predict any case. However, sometimes, it may be not reasonable to do imputation. For example, if gender is one of the important explanatory variables in a study and an new instance is tested with gender missing. According to the bootstrap imputation algorithm, we use the mode of gender to replace the missing place. This may be totally wrong, which has large impact on the final predictions.

5. CART1 and CART2 have unique and simple ways to deal with missing values, although their misclassification errors are somewhat higher than other methods. When the proportion of missingness increases to 30%, the performance of CART is better than that of weighted-averaging logistic classifiers. If the data contain even more missing values, CART

can easily handle, on the contrary, weighted-averaging classifiers may be failure since to estimate certain number of coefficients the data has to meet some requirements regarding to the number of completed cases.

5.3 Results of "Fossil Skull Data"

If different analytical methods are applied to the fossil skull data with M1, M8, M17, M45 and M52 as explanatory variables, the results are described in the following tables. CART1 stands for the classification and regression trees based on minimum cross-validation error criteria and CART2 means the CART algorithm based on 1-SE rule. Since the data includes 397 observations with 147 completed cases, we use 11-fold cross validation to calculate misclassification error rates.

Table 5.7: Methods Comparison on "Fossil Skull Data"

	M1,45	M1,8,17,45,52	AIC	Bootstrap
Linear(NO.out of 396)	0.1818(242)	0.1837(147)	0.1837(147)	0.2323(396)
Logit(NO.out of 396)	0.1777(242)	0.1769(147)	0.1769(147)	0.2298(396)

Table 5.8: Methods Comparison on "Fossil Skull Data"

	CART1	CART2
Misclassification Error(NO.out of 396)	0.2374(396)	0.2248(396)

Table 5.9: Methods Comparison on "Fossil Skull Data"

Weighted-Averaging	$f = 1/se^2$	$1/se$	$1/\sqrt{se}$
Linear(NO.out of 396)	0.25(396)	0.2424(396)	0.245(396)
Logistic(NO.out of 396)	0.2525(396)	0.2475(396)	0.2424(396)

1. Table 5.9 also shows that the weight $1/\sqrt{se}$ has the smallest misclassification errors than other weights both for linear and logistic regression.

2. In Table 5.8, the first column is the model used in [10] with only M1 and M45 as predictors. Comparing to the model with M1, M8, M17, M45, M52 and the model with AIC criteria, the error rates are very close. However, the key disadvantage of these three models is that they can only be tested on part of the data. 242 for M1 and M45 models and 132, 147 for the larger models and AIC models, respectively. The CART, bootstrap and weighted-averaging classifiers have been tested on all cases.

3. Although the error rates of CART, Bootstrap and weighted-averaging classifiers are quite close, the CART with 1-SE rule has a better performance in the fossil skull data, followed by bootstrap imputation, CART with minimum error criterion and weighted-averaging with weight $1/\sqrt{se}$. Thus, when there is a large amount of missingness in data (e.g. 250/397=62.97% in this case), classification and regression trees would be a better option.

To compare the results of CART with 1-SE rule, Bootstrap imputation and weighted-averaging classifier with weight $1/\sqrt{se}$ more clearly, cross

tabulation are provided.

Table 5.10: CART2 vs. Bootstrap Logistic

CART2 \ Bootstrap Logit	Female	Male
Female	130	39
Male	9	218

Table 5.11: CART2 vs. Weighted-averaging $1/\sqrt{se}$ Logistics

CART2 \ Wgt-ave Logit	Female	Male
Female	111	58
Male	5	222

Table 5.12: Weighted-averaging $1/\sqrt{se}$ Logistics vs. Bootstrap Logitstic

Bootstrap Logit \ Wgt-ave Logit	Female	Male
Female	113	26
Male	3	254

Table 5.13: CART2 vs. Bootstrap Linear

CART2 \ Bootstrap Linear	Female	Male
Female	127	42
Male	9	218

Table 5.14: CART2 vs. Weighted-averaging $1/\sqrt{se}$ Linear

CART2 \ Wgt-ave Linear	Female	Male
Female	110	59
Male	5	222

Table 5.15: Weighted-averaging $1/\sqrt{se}$ Linear vs. Bootstrap Linear

Bootstrap Linear \ Wgt-ave Linear	Female	Male
Female	111	25
Male	4	256

The summary of the final optimal tree with 1-SE rule is described as follows:

```
> summary(tr.rp2)
......
```

```
Node number 1: 361 observations,    complexity param=0.4459459
  predicted class=1  expected loss=0.4099723
    class counts:   148   213
   probabilities: 0.410 0.590
  left son=2 (152 obs) right son=3 (209 obs)
  Primary splits:
    V2 < 187.5 to the left,  improve=50.290200, (27 missing)
    V5 < 138.5 to the left,  improve=21.774910, (138 missing)
    V4 < 141.5 to the left,  improve=10.264200, (172 missing)
    V3 < 144.5 to the left,  improve= 4.723763, (26 missing)
    V6 < 30.5  to the left,  improve= 3.633681, (131 missing)
  Surrogate splits:
    V3 < 129.5 to the left,  agree=0.551, adj=0.007, (16 split)

Node number 2: 152 observations
  predicted class=0  expected loss=0.2828947
    class counts:   109    43
   probabilities: 0.717 0.283

Node number 3: 209 observations
  predicted class=1  expected loss=0.1866029
    class counts:    39   170
   probabilities: 0.187 0.813
```

Chapter 6

Conclusion

By the empirical studies in the last chapter, we summarize the following conclusions.

- Among the three studies, the weight $1/\sqrt{se}$ in weighted-averaging classifiers tends to have smaller error rate than other weights. Thus, if we choose to use the weighted-averaging classifier as the analysis method, $1/\sqrt{se}$ should be the optimal weight option.

- The models with all important covariates and models selected according to the AIC criteria have smaller errors than other methods when the amount of missingness is moderate. However, the key disadvantage of these methods is that classification/prediction of a new instance may not be obtainable if there is missingness in predictors. The objective of classification problems is to discriminant observations into different groups. Thus, from this point of view, bootstrap imputation, CART and weighted-averaging classifier would be better choices. Moreover, as the proportion of missingness increases and exceeds 25%, the superior

performance of these three methods appears.

- If we compare the tree methods: bootstrap imputation, CART, and weighted-averaging classifier with weight $1/\sqrt{se}$, we would say that bootstrap imputation sometimes may be unstable and not rationale. When there's no missing value in the testing data (study 2), bootstrap performs the best among the three competitors. When the missingness in data is not large (study 1), the weighted-averaging classifier performs the best. When the missingness is very large, as the situation in the fossil skull data, CART would be the best choice. It misclassified less cases than other methods.

References

[1] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and regression trees*, Wadsworth, Belmont, CA, USA, 1984.

[2] Marc R. Feldesman, *Classification trees as an alternative to linear discriminant analysis*, American Journal of Physical Anthoropology **119** (2002), 257–275.

[3] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning*, 1st ed., Springer-Verlag, Berlin, Heidelberg, New York, USA, 2001.

[4] Richard A. Johnson and Dean W. Wichern, *Applied multivariate statistical analysis*, 6th ed., Prentice Hall, Pearson, 2008.

[5] Roderick J. A. Little and Donald B. Rubin, *Statistical analysis with missing data*, 2nd ed., Wiley-Interscience, New York, USA, 2002.

[6] Martin R and Saller K, *Lehrbuch der anthropologie*, Bd 1. Gustav Fisher, Stuttgart, 1957.

[7] Brian D. Ripley, *Pattern recognition and neural networks*, 1st ed., Cambridge University Press, Cambridge, UK, 1996.

[8] Terry M. Therneau and Elizabeth J. Atkinson, *An introduction to recursive partitioning using the rpart routines*, 1997.

[9] Šefčáková A., *A new find of upper palaeolithic skull in Slovakia.*, Anthropologie (Brno) **35(2)** (1997), 233.

[10] Šefčáková A., Mizera I., and Thurzo M., *New human fossil remains from Slovakia. The skull from Moča (late upper palaeolithic, south Slovakia)*, Bulletin Slovenskej Antropologickej Spoločnosti **2** (1999), 55–63.

[11] Henke W., *Jungpaläolithiker und mesolithiker: Beiträge zur anthropologie*, Tech. report, Johannes Gutenberg-Universität, Mainz Habilitationsschrift, 1989.

[12] Howells W. W., *Cranial variation in man: a study by multivariate analysis of patterns of difference among recent human populations*, Papers of Peabody Museum of Archaeology and Ethnology **67** (1973).