2013

Statistical classification problems in assessment of teachers

Xuan Wang
Louisiana State University and Agricultural and Mechanical College, baobeiwangxuan@hotmail.com

STATISTICAL CLASSIFICATION PROBLEMS IN ASSESSMENT OF TEACHERS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
In partial fulfillment of the
requirements for the degree of
Master of Science

in

The Department of Mathematics

by
Xuan Wang
B.S., Louisiana State University, 2011
August 2013

ACKNOWLEDGMENTS

I would like to thank many people who give me help and supporting. First, I take this opportunity to thank my mentor Dr. P. Sundar for the expert help through the whole process of my master's thesis. Furthermore I would like to thank Dr. A. Sengupta and Dr. J. N. Shreve for their help and suggestions in my thesis. Also, I like to thank Dr. J. Madden for providing the data. Finally, I want to appreciate my family supporting me throughout the time in graduate school.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Classification and regression trees form an important and indispensable tool in data analysis and classification problems. Class trees are described in detail with examples. The method is applied to a data set pertaining to evaluation of teachers. In addition, two other classification methods, bagging and AdaBoost are explained. These methods improve existing classifiers to nearly optimal classifiers.

CHAPTER 1. INTRODUCTION

According to Breiman (1984) "the basic purpose of a classification study can be either to produce an accurate classifier or to uncover the predictive structure of the problem". In classification problems, a large number of measures have been created to evaluate the accuracy of classifiers. A good classifier should not only be accurate in terms of minimizing misclassification errors, but also be a good predictor of future data.

For the model-based methods in statistics, there are a variety of classification procedures such as linear discriminant analysis, quadratic classifier, logistic regression and variable kernel density estimation. Linear discriminant analysis abbreviated as LDA is a statistical method used widely, in pattern recognition and machine learning. LDA is related to principal component analysis since both methods use linear combinations of features to classify data. (Martinez, 2011) Quadratic classifier is another method of statistical classification that uses a quadratic surface to classify data for pattern recognition (Cover, 1965). Logistic regression is a statistical method that predicts the outcome as a categorical variable, and "focuses instead upon the relative probability (odds) of obtaining a given result category." (Guido, 2006) Variable kernel density estimation is the method of using the size of kernels to estimate the locations. (Terrell, 1992)

For the algorithmic methods in statistics, decision tree, bootstrap aggregation and boosting are all typical algorithmic methods in classification problems. There are two kinds of decision trees. One type is called classification trees, and another is called regression trees. CART is the acronym for decision trees, and stands for classification and regression trees. The method consists in using a tree structure to classify data and build the most accurate tree, in order to classify or predict the numerical value of a response variable. Section 2.1 will briefly discuss the algorithm for building a classification tree. In section 2.2, there is an illustration of how to use

the program Rattle in R to build the classification tree in order to evaluate the teachers' efficiency based on the test scores of students. "R is a programming language and environment developed for statistical analysis by practicing statisticians and researchers" (Williams, 2011), and Rattle is a graphical user interface for data mining that is created by the programming language R. In section 2.3, the algorithm for building a regression tree by finding the optimal splitting points will be presented. Section 2 explains the ways to build classification as well as regression trees, and the differences between them. CART was invented 30 years ago, and recently, several of these methods have been proposed. Bootstrap aggregation is a method that can improve the performance of an existing classifier. Bagging predictors introduced in Breiman(1996), is a method based on bootstrap aggregation in order to overcome limitations of data and attain better classification. In section 3.1, the algorithm for bagging predictors will be discussed. The principle of bagging predictors in classification and numerical prediction will be addressed in section 3.2. Boosting algorithm is a method that improves the performance of weak classifiers; it uses an average of weak classifiers to perform the classification. Adaboost is the most popular of the boosting algorithms (Hertzmann& Fleet, 2011). In section 4.1, Adaboost algorithm will be discussed. An example that illustrates the Adaboost algorithm is given in section 4.2. Finally, Section 5 details the advantages and disadvantages of CART, bagging and Adaboost with a full comparison of these three methods.

CHAPTER 2. DECISION TREES

Consider a data set in which there is a target attribute and at least one conditional attribute. The target attribute is the response variable that one needs to classify or predict. Conditional attributes are the input variables that help us to build a model to classify or predict the value of a response variable. If the target attribute is numerical, the model is called a regression tree; if the target attribute is categorical or labeled by classes, the model is called a classification tree. To build trees, one needs to separate data into a learning set and a testing set. (Breiman, 1984) recommends setting up 70% of original data as learning set, and the rest of the data as a testing set before building a tree. In building a decision tree, we need a rule for selecting the best split at any node and a criterion for choosing the right-sized tree. In the section 2.1, we start with a discussion on ways to find the best split at any node.

2.1 Algorithm for Classification Tree

Definition 2.1.1 *(Impurity Function)*

Let J be a positive integer, and let D denote set of all J-tuples of numbers $(p_1, p_2, \dots, p_J)$ *satisfying* $p_j \geq 0, j = 1,2, \dots, J, \sum_j p_j = 1$. *A non-negative function* $\emptyset$ *defined on D is called an impurity function if the following properties hold:*

(i) $\emptyset$ *attains maximum at the point* $(\frac{1}{J}, \frac{1}{J}, \dots, \frac{1}{J})$,

(ii) $\emptyset$ *achieves its minimum at the points* $(1,0, \dots ,0), (0,1, \dots ,0), \dots , (0,0, \dots ,1)$,

(iii) $\emptyset$ *is a symmetric function of* $p_1, p_2, \dots, p_J$.

Definition 2.1.2 *(Impurity Measure)*

Given an impurity function $\emptyset$ *the corresponding impurity measure* $i(t)$ *at any node t is defined by*

$$i(t) = \emptyset(p(1|t), p(2|t), \dots p(J|t)),$$

where $p(j|t)$is the estimated probability of class j given at node t.

If a split s of a node t sends a proportion p_R of the data already in t to t_R and the proportion p_L to t_L, define the decrease in impurity to be

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

Maximizing the decrease in $\Delta\, i(s,t)$ impurity is a criterion for split selection.

In classification trees, entropy, Gini index and misclassification error are the most popular impurity functions. Entropy and Gini index are differentiable and are more sensitive to change in the node probabilities. Hence, they are preferred in finding splitting points. Though entropy as an impurity function was well-known for a longtime, Gini index used by Breiman is the measure most commonly chosen for classification trees. And misclassification error is used to prune a tree to find the right-size of a tree after we build the model by using the learning data set. (Tan, 2006) Next, the three impurity functions are introduced, and the relationship between entropy and Gini index will be shown.

Entropy is a measure of randomness, and is defined by the impurity function of entropy

$$i(t) = -\sum_{k=1}^{k} p_k \log(p_k),$$

where p_k is the probability of class k, in a multinomial convex t.

Next, an example is given of how to use entropy impurity function to find the split points. Suppose there is a root node t that includes all the learning set of data with the sample size is 10, and this data is classified into two classes where the class 1 has the sample size is 4, and the class 2 has the sample size is 6. Let us look at the Figure 1.

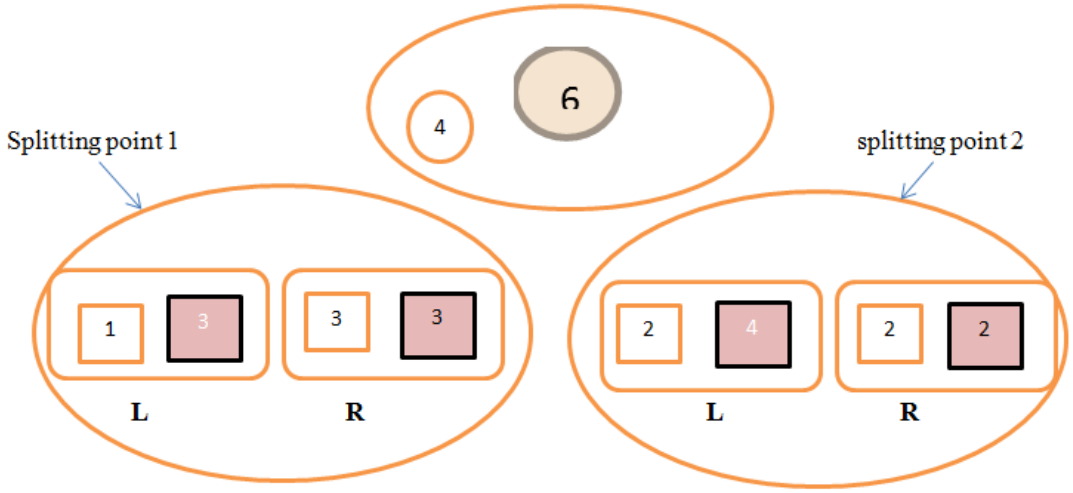


Figure 1. Illustration for Finding a Splitting Point

The impurity function of entropy at node t is

$$i(t) = -\left(\frac{4}{10}\right)\log\left(\frac{4}{10}\right) - \left(\frac{6}{10}\right)\log\left(\frac{6}{10}\right) = 0.2922.$$

Consider two possible splitting points to separate the root node into two parts. We below describe the rule to choose the optimal splitting points, among the two choices. Consider splitting point 1 where the left node contain four observations with one observation belonging to class 1 and three observations belonging to class 2. The right node contains six observations with three belong to class 1 and three observations belong to class 2. At splitting point 1, the impurity of the left side node with $p_L = \frac{4}{10}$ is

$$i(t_L) = -\left(\frac{1}{4}\right)\log\left(\frac{1}{4}\right) - \left(\frac{3}{4}\right)\log\left(\frac{3}{4}\right) = 0.2442\,.$$

The impurity of the right side node with $p_R = \frac{6}{10}$ is

$$i(t_R) = -\left(\frac{3}{6}\right)\log\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log\left(\frac{3}{6}\right) = 0.3010.$$

So the decrease of impurity

$$\Delta\, i(s,t) = i(t) - p_R i(t_R) - p_L i(t_L) = 0.2922 - \left(\frac{6}{10}\right) * 0.3010 - \left(\frac{4}{10}\right) * 0.2442 = 0.01392.$$

Now consider splitting point 2 where the left node contains six observations with two in class 1 and four in class 2; the right node contain two observations in class 1 and two observations in class 2. Repeat the same steps of splitting point 2, the

$$i(t_L) = 0.2764\ with\ p_L = \frac{6}{10},$$

$$i(t_R) = 0.3010\ with\ p_R = \frac{4}{10},$$

so

$$\Delta\, i(s,t) = i(t) - p_R i(t_R) - p_L i(t_L) = 0.00596\,.$$

As the $\Delta\, i(s,t)$ of splitting point 1 is greater than that of splitting point 2, splitting point 1 is preferred.

Gini index will be introduced next, which is the most important and widely used impurity function. It is also the default option in most of the programs to run the classification tree, especially the rpart package in R. First, the idea of how to use Gini index to find the splitting points will be elaborated.

Gini measure, used by Breiman (1984) is defined below:

$$i(t)_{Gini\ index} = \sum_{i \neq j} p(i \mid t) p(j \mid t),$$

$$p(j \mid t) = p(j,t)/p(t),$$

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

Where $p(j \mid t)$ is the estimated probability given in node t of a sample in class j. p(j, t) is the estimated probability of a sample in group j at node t. $p(t)$ is the probability of a sample at node t. π(j) is called the prior probability of class j, it is used to calculate the proportion of data in

every class. $N_j(t)$ is the number of samples in class j at node t. N_j is the number of samples in each class.

The Figure 2 and the example below give a brief explanation of the notations shown in the formulas of Gini measure:

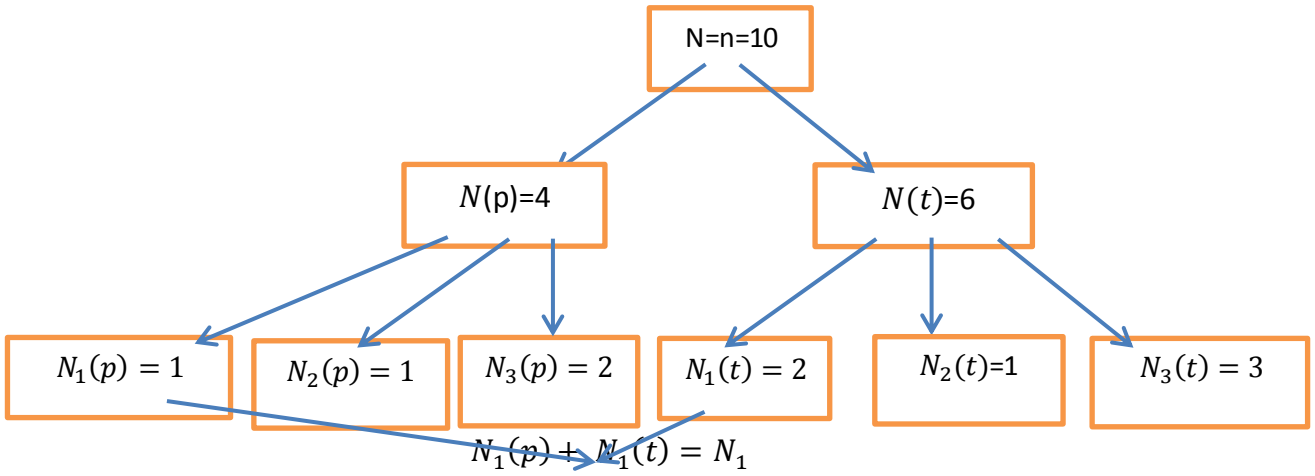


Figure 2. Explanation of Notations in Gini Measure

Suppose there are n independent observations, the total number of samples is n=10 that indicates N, which is also called the root node. If N is separated into 3 classes, so the number of samples in each class indicates as N_j.

If the root node is separated to two notes, one is called node t, and another is called node p, the number of samples going to node t is defined by

$$N(t) = 6,$$

and the number of samples going to node p is indicated by $N(p) = 4$.

As N is separated to three classes, the node t and node p also will be separated to three classes. Let j=1, 2, 3, suppose the number of samples in

$$N_1(t) = 2, N_2(t) = 1, N_3(t) = 3,$$

so they represent as $N_j(t)$.

Suppose the number of samples in

$$N_1(p) = 1, N_2(p) = 1, N_3(p) = 2 ,$$

so they represent as $N_j(p)$, and it follows

$$N_j = N_j(t) + N_j(p),$$

with

$$N_1 = 3, N_2 = 2, N_3 = 5.$$

If N is separated by k parts,

$$\sum_{j=1}^{k} N_j = N .$$

And at node t, it also needs to be satisfy with

$$\sum_{j=1}^{k} N_j (t) = N(t).$$

$\pi(j)$ is called the prior probability of class j, and the prior probability is used to estimate the proportion of data in every class.

In figure 1, if the prior probability of class 1 is wanted, so

$$\pi(j) = \pi(1) = \frac{[N_1(t)+N_1(p)]}{N} = \frac{[2+1]}{10} = 0.3.$$

By the formulas given by Breiman, $p(j,t)$ is the estimated probability of a sample in group j and at node t. For example,

$$p(1,t) = \frac{\pi(1)N_1(t)}{N_1} = 0.3 * \frac{2}{3} = 0.2.$$

As $p(t)$is the estimated probability of a sample at node t, so it follows

$$p(t) = N(t)/N.$$

And $p(j \mid t)$ is the estimated probability of a sample in group j given at node t, so $p(j \mid t) = p(j,t)/p(t)$.

For example,

$$p(1 \mid t) = \frac{p(1,t)}{p(t)} = \frac{\pi(1)N_1(t)}{N_1} * \frac{N}{N(t)} = 0.2 * \frac{10}{6} = 1/3. \qquad \text{(Breiman, 1984)}$$

In fact, the most common and equal way using Gini index is written as

$$i(t)_{Gini\ index} = \left(\sum_j p(j|t)\right)^2 - \sum_j p^2(j|t)$$

$$= 1 - \sum_j p^2(j|t)\ ,$$

where k is the number of classes for the response variable.

Let us consider the example in Figure 1. The impurity by using Gini index at node t is

$$i(t) = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2 = 0.48\ .$$

Considering the splitting point 1, the impurity of the left side node with $p_L = \frac{4}{10}$ is

$$i(t_L) = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.375\ .$$

The impurity of the right side node with $p_R = \frac{6}{10}$ is

$$i(t_R) = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5.$$

So the decrease of impurity

$$\Delta\ i(s,t) = i(t) - p_R I(t_R) - p_L I(t_L) = 0.48 - \left(\frac{6}{10}\right) * 0.5 - \left(\frac{4}{10}\right) * 0.375 = 0.03.$$

Repeat the steps and do the splitting point 2, the $i(t_L) = 0.444$ with $p_L = \frac{6}{10}$, $I(t_R) = 0.5$ with $p_R = \frac{4}{10}$, so the decrease impurity

$$\Delta\ i(s,t) = i(t) - p_R I(t_R) - p_L I(t_L) = 0.0136\ .$$

As the $\Delta\ i(s,t)$ of splitting point 1 is greater than that of splitting point 2, the splitting point 1 is also preferred to be selected by the impurity measures defined by Breiman (1984). This is the same result obtained by using entropy.

Now let us look at the relationship between entropy and Gini index working under 2 classes.

Let k=2, the impurity function of Entropy is

$$i(t)_{impurity} = -\sum_{i=1}^{2} p_i \, log_2 \, p_i = -p_i log_2 p_i - (1-p_i) log_2 (1-p)$$

$$= -p_i \left(\frac{\ln p_i}{\ln 2}\right) - (1-p_i)\left(\frac{\ln(1-p_i)}{\ln 2}\right)$$

As we know, $\ln x = \ln[1-(1-x)]$ approximates $-(1-x)$ because expanding of the Taylor series when$|(1-x)| < 1$, so

$$i(t)_{impurity} = \frac{1}{\ln 2}[-p_i(p_i - 1) - (1-p_i)(1-p_i-1)]$$

$$= \frac{1}{\ln 2}[-{p_i}^2 + p_i - (1-p_i)^2 + 1 - p_i]$$

$$= \frac{1}{\ln 2}[1 - {p_i}^2 - (1-p_i)^2]$$ → This is the Gini index under 2 classes.

$$= \frac{1}{\ln 2}\, i(t)_{Gini\ index}$$

As $\frac{1}{\ln 2}$ is greater than 1, under the 2 classes, the impurity in using Gini index is a little smaller than using entropy. As the two impurity functions do not make big difference between each other, this is also why entropy and Gini index are two popular ways using in CART, however, Gini index is more sensitive to change in the node probability. And Breiman (1984) found "The Gini index is simple and quickly computed, it can also incorporate symmetric variable misclassification costs in a natural way." (Breiman, 1984)

As a tree cannot grow infinitely, the stopping criterion can depend on the number of samples, the depth of the tree growing, the probability of accuracy and the number of classes in terminal nodes. These criterions can be specified and set up in the program while running CART and will be discussed in section 2.2.

The last impurity function is Misclassification rate that is defined as:

$$i(t)_{misclassification\ rate} = 1 - max_j p(j|t),$$

Where $p(j|t)$ is the estimated probability of class j at node t, and misclassification rate is preferred to use in pruning the tree. (Tan, 2006) When the classification tree is developed, the misclassification rate will initially decrease, but it will hit a minimum rate while the tree is growing. After the tree reaches a minimum misclassification rate, it will increase, so the part of the tree after the misclassification rate has hit the minimum is called over fitting. Hence it is necessary to prune the tree. Cross validation is the most popular method to prune the tree and is the default method in most programs. Cross-validation is a node validation technique that can estimate the accuracy and performance of a model build. K-fold cross-validation randomly divides the data into k parts with the same size, and uses these parts to test the estimated accuracy. However, 10-fold cross-validation is more preferred in forming a good classifier method (Kohavi, 2005). Further methods about pruning classification trees are discussed in Breiman (1984).

2.2 Example of Building Classification Tree

The data included in Appendix A is provided by Dr. Madden in the mathematics department at Louisiana State University. The data includes 728 teachers containing a different number of students in each class, and the students' pretest and posttest score. The data is organized by each teacher and is provided in Appendix B. It is clear to understand pretest average score is the average of the students' pretest score for each teacher, and pretest sd is the stand deviation of the students' pretest scores for each teacher. It is similar as the explanations of posttest average score and posttest sd. The difference sd is the stand deviation for the difference of two average scores. The percentage going up is the percentage of the students who make the improvement from pretest score to posttest score. And the number of students is the class size for each teacher in the data. With the limitation of data, the difference of the two average scores becomes a condition to

assume the classification of the teachers. In cases where the difference is greater than 5, we classify these teachers as having made improvements with the students, and this is indicated as 1. If the difference is less than or equal to 5, we classify that these teachers did not make improvements with the students and this is indicated as 0. It is clear to see $j = 2$ for this data as the target attribute in 2 classes.

Figure 3 is the interface of Rattle in R. The partition command refers to the ratio of splitting the learning set and testing set. Rattle defaults that the learning set is 70% of the original data, and the remaining 30% sets up as the testing data. As the target attribute is class labeled, the categorical target type was checked on the interface. The rows shown on the interface are the attributes in the organized data. Bin is the classifier for the teachers, so it puts in XXX as the target. As we use the difference of two average scores to assume the classification of the teachers, the attribute of 'post-pre' should be ignored. Presume the three conditional attributes are checked in the Figure to be used in building the classification tree.

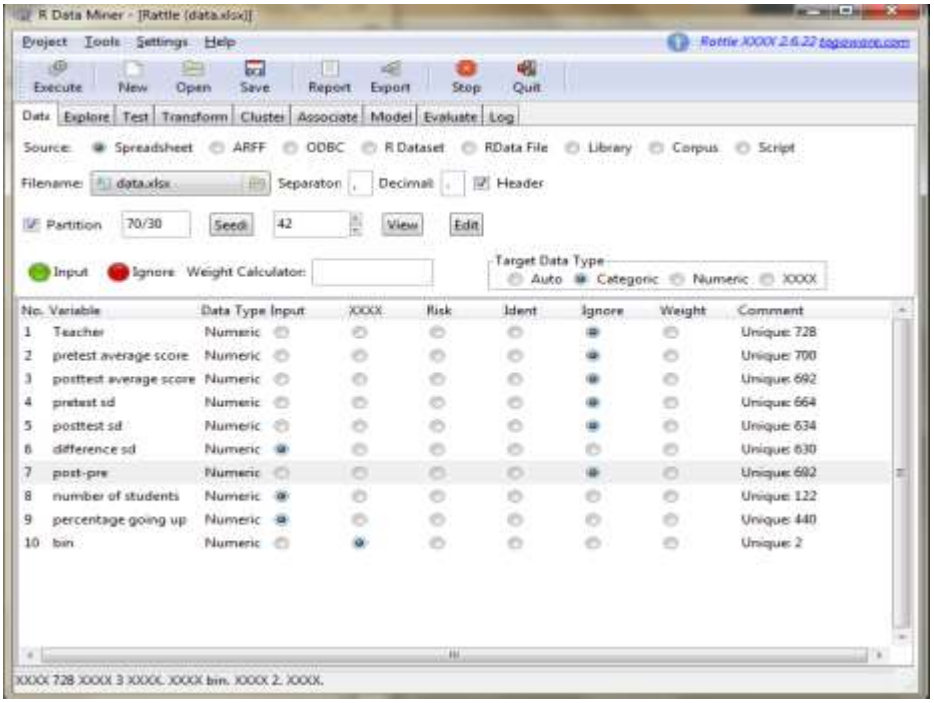


Figure 3. Interface for Rattle in R

After executing the Rattle in R, Figure 4&5 shows the information and classification tree. Graham Williams (2011) suggests in his book *Data Mining with Rattle and R* that the defaults in Rattle are all based on rpart's defaults. The min split argument specifies the minimum observations of doing the splitting in a node. The min bucket argument designates the minimum observations of each leaf nodes or terminal nodes. The complexity argument is utilized for controlling the pruning of the decision tree; it will provide the most optimal tree. The default of complexity is 0.0100, indicating that there is at least 1% probability gain in every continuous splitting of the nodes. Max Depth is used for limiting the depth of a decision tree. The defaults for these arguments can be changed to control the size of the tree, however, in the data; the defaults are used for the programming of the decision tree. (Williams, 2011)

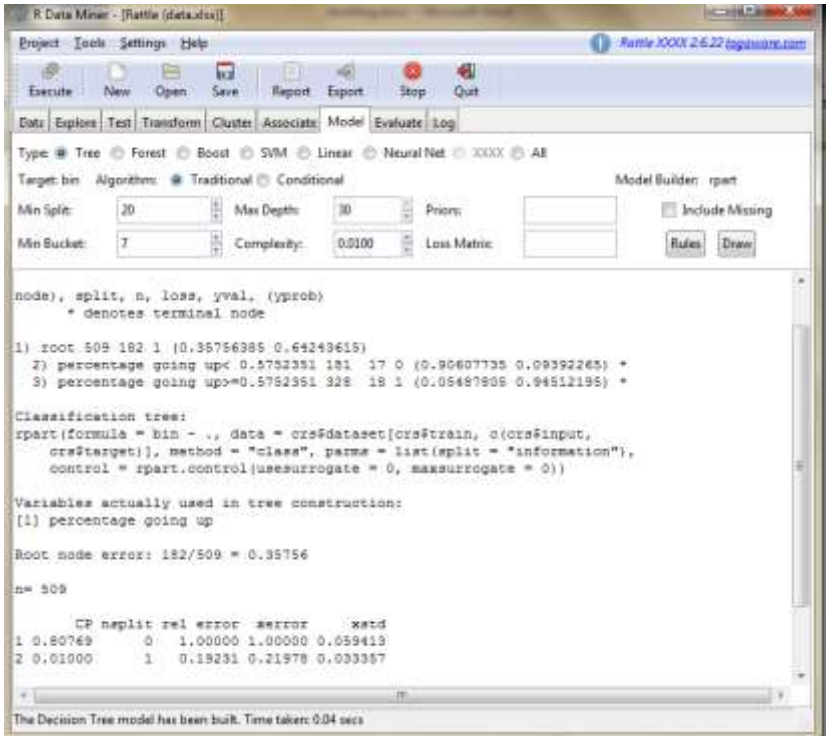


Figure 4. Output for Running the Data by Using Rattle

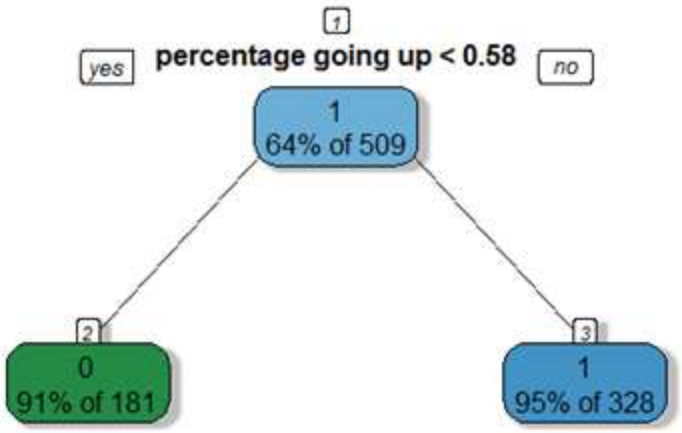
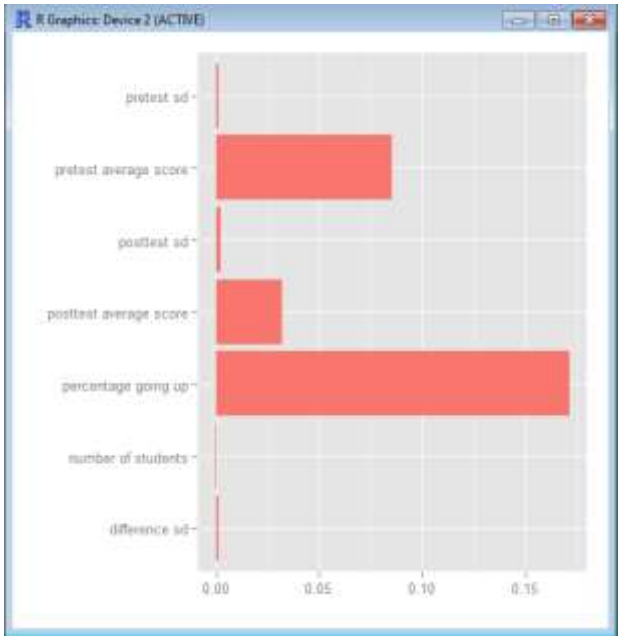


Figure 5. Classification Tree for the Data

After executing the Rattle, the classification tree is shown in Figure 5. It is clear to see that three conditional attributes were chosen to build the classification tree. The program only uses the percentage going up as the conditional attribute because this conditional attribute will give the best probability of classification in the terminal nodes. In node 2, 91% of teachers 181, were classified as class 0, having made no improvements and in node 3, 95% of the teachers,328, classified as the class 1, having made improvements with the students. The correct rate in terminal nodes is significant enough to say this is an adequate classification tree even utilizing one conditional attribute.

The Figure 6 is showing the importance of each conditional attribute with the target attributes:




```
Variable Importance
===================

                            Importance
percentage going up         0.17286631016
pretest average score       0.08516577540
posttest average score      0.03191443850
posttest sd                 0.00204278075
difference sd               0.00170053476
pretest sd                  0.00096256684
number of students         -0.00007486631
```

Figure 6. Importance of the Conditional Attributes

In Figure 6, the importance of each conditional attribute is illustrated by running the program; the percentage going up affects the target attributes the most. This is also an important reason that the decision tree only has percentage going up as the conditional attribute to do the separation. The classification tree shown in Figure 5 is the final tree that is already pruned by the testing set. The cross-validation is the default method to prune the tree in Rattle. The cross-validation measures the relative error; this is the default method in rpart package in R to prevent over fitting. (Williams, 2011)

If the conditional attribute of percentage going up is ignored and the separation of a big class and small class is based on the number of students, the standard deviation in average scores can measure the level of improvement of the students for each teacher, and the standard deviation in

pretest scores can indicate the level of the students in each class for the teacher. So after executing the Rattle, the classification tree will be shown as Figure 7:

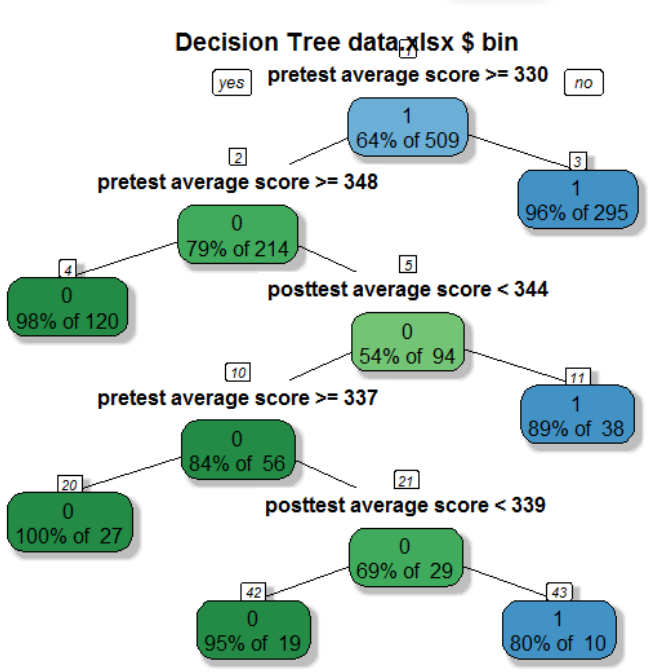


Figure 7. New Classification Tree for the Data

Based on the tree built above, it can be concluded that this classification tree is sufficient as the terminal nodes that all have very high percentage have correctly classified the teachers. Suppose we pick the terminal 4 to do the explanation. There are two splitting points are showed in the classification tree which are pretest average scores greater or equal to 330 and the pretest average score greater or equal to 348, altogether, if a teacher is classified into this node, it indicates that the teacher has a probability of 98% that there is no improvement by the students.

2.3 Algorithm for Regression Tree

A Regression tree is used when the target attribute is numerical. The tree can be used to predict the value of the response variable y. Suppose the learning set is given by $\{x_i, y_i\}$, where $i = 1,2, \dots N$. Let the response variable y be numerical. Suppose $n(t)$ is the number of data elements that fall in the node t, define

$$\overline{y(t)} = \frac{1}{n(t)} \sum_{x_i \in t} y_i \,,$$

where n(t) is the total number of data in node t. Let $\tilde{T}$ denotes all the nodes in the regression tree, so the average of the total sum of squared errors for the tree is

$$R(t) = \frac{1}{N} \sum_{t \in \tilde{T}} \sum_{x_i \in t} (y_i - \overline{y(t)})^2$$

Breiman (1984) defined the best split of building regression tree as

Definition 2.3.1 *(Best split for regression tree)*

The best split S of t is that split in S which most decreases $R(t)$.

More precisely, for any split S of t into t_L and t_R, let

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

Take the best split s to be a split such that

$$\Delta R(s,t) = max_{s \in S} \Delta R(s,t).$$

Next, the steps of building a regression tree are introduced. The first step to set up a regression tree is the same as for classification trees, to separate the data to learning set and testing set. Breiman referred that the learning set is around 70% of the original data, and the remaining 30% is the testing set in order to prune the tree. Then start with a single node that contains all the data in the learning set. The key difference of splitting in a regression tree is using the mean squared error to measure the variance reduction. The largest decrease in the variance will be the best splitting point in building a regression tree. (Breiman, 1984) The stopping criterion can depend on the same principles that were used for classification trees, and cross-validation can also be employed to prune the tree.

The decision tree algorithm can be used extensively in mixed types of variables, even where some values are missing. This section has introduced how to identify splitting points in either

classification tree or regression tree. The algorithm is easy to follow and the models are generally simple to interpret. Since the decision tree has been very popular for 30 years, the power of prediction tends to be inferior and there are more and more classification methods developed to improve the accuracy of the data. (Williams, 2011) In the next section, bagging will be introduced.

CHAPTER 3. BAGGING

Breiman (1996) referred to the classification and regression trees as unstable, if the classifier is sensitive to small changes in the sample. "Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor." Breiman (Breiman, 1996) mentioned in his paper of "Bagging Predictors", is the procedure of "bootstrap aggregation".

3.1 Algorithm for Bagging Predictor

Consider a learning set $L = \{x_i, y_i\}, i = 1,2, \dots n$, n can be any positive integer. The sample size of this learning set is n, and the response variable of y is either categorical or numerical. Suppose $\emptyset(x, L)$ is the predictor from the learning set and there are a sequence of learning sets $\{L_k\}$, each also contains n independent observations were given from the same learning set L. For each learning set of $\{L_k\}$, there is a new predictor $\emptyset(x, L_k)$ for the response variable. The goal for the Bagging is using these new predictors to find a better predictor than using a single learning set predictor.

If the response variable y is categorical and is separated by $j \in \{1,2, \dots, J\}$ classes, it indicates $\emptyset(x, L)$ predicts a class label. For each predictor $\emptyset(x, L_k)$ in the sequence of learning sets $\{L_k\}$, we are going to take the voting of these predictors to aggregate the predictor $\emptyset(x, L)$. Let $N_j = nr\{k; \emptyset(x, L_k) = j\}$and the new predictor denotes as $\emptyset_A(x) = \arg max_j N_j$, the results of class label will follow the maximum of N_j.

If the response variable of y is numerical, the $\emptyset_A(x)$ equals the average of the sequences of predictors$\emptyset(x, L_k)$, it denotes

$$\emptyset_A(x) = E_L \emptyset(x, L_k) = av_A \emptyset(x, L_k),$$

with the expectation over L is denoted by E_L .

However, there are not many replicates of L in each dataset, so bootstrap samples become a good method to assume the L_k in a single learning set. Pick up random n observations with the replacement from the learning set every time that is denoted by L^B, so the new predictor of each bootstrap sample denotes as $\emptyset(x, L^B)$ and the final aggregate predictor denoted as $\emptyset_B(x)$. If the response variable y is class labeled, the $\emptyset_B(x)$ will be voted by $\emptyset(x, L^B)$.

For example, the table 1 below is a learning set where the response y is categorical and indicates the sample size is 4 in 2 classes:

Table 1. An Example for Bagging Predictor with Response Variable is Categorical

index	1	2	3	4
Value of x	2	4	6	8
Value of y	1	1	0	1

Suppose it is resampled 3 times, the first instance includes the samples indexed as 1,1,2,3, and the procedure method for the $\emptyset(x, L^B)$ =1; the second sampling includes the samples indexed as 1,3,3,4, and the procedure method for the $\emptyset(x, L^B) = 0$; the final instance includes the samples indexed as 1,2,3,4, and the procedure method for the $\emptyset(x, L^B) = 1$. In the aggregate for the three predictors, it is clear to see $\emptyset_B(x) = 1$ as class 1 have more instances than class 0.

If the response variable of y is numerical, the $\emptyset_B(x)$ equals the average of the sequences of predictors$\emptyset(x, L^B)$, it denotes

$$\emptyset_B(x) = E_L\emptyset(x, L^B) = av_B\emptyset(x, L^B).$$

For example, the table 2 below is a learning set where the response variable y is numerical and is given with the sample size of 4:

Table 2. An Example for Bagging Predictor with Response Variable is Numerical

Index	1	2	3	4
Value of x	2	4	6	8
Value of y	6	8	9	7

Suppose these are also resampled 3 times, the first time includes the samples that are indexed as 1,1,2,3, and the procedure method form the $\emptyset(x, L^B)$ =6.7; the second sample includes the samples that are indexed as 1,3,3,4, and the procedure method for the $\emptyset(x, L^B) = 8.6$; the third sampling includes the samples that are indexed as 1,2,3,4, and the procedure method for the $\emptyset(x, L^B) = 7.5$. In the aggregate of the three predictors, it is clear to see

$$\emptyset_B(x) = av_B\emptyset(x, L^B) = \frac{6.7+8.6+7.5}{3} = 7.6\,,$$

the new predict value of y for the input x is 7.6.

3.2 Principle in Classification

Suppose a learning set $L = \{x_i, y_i\}, i = 1,2, \dots n$, n can be any positive integer and the response variable of y is class label $j \in \{1,2, \dots, J\}$. Let a predictor $\emptyset(x, L)$ of this learning set predicts a class label j$\in \{1,2, \dots, J\}$, and set up a new notation $\varphi(j|x)$, it denotes the relative frequency of the predictor $\emptyset(x, L)$ predicts input x in class j within the number of independent resampling from the learning set L. Let

$$\varphi(j|x) = P(\emptyset(x, L) = j).$$

If $P(j|x)$ indicates the probability of input x that can lead to the class j, then $\sum_j \varphi(j|x)P(j|x)$ is the probability of predictor $\emptyset(x, L)$ correctly classify class label of the input x. Set up $\varphi(j|x)$ is an indicator function which is defined as

$$\varphi(j|x) = \begin{cases} 1 \; if P(j|x) = max_i P(i|x) \\ 0 \; if \; P(j|x) \neq max_i P(i|x) \end{cases},$$

So it is easy to obtain

$$\sum_j \varphi(j|x)P(j|x) \leq max_j P(j|x) \quad (1)$$

Let $P(x)$ denotes the probability distribution of x in each class and make

$$Q(x) = \left[\sum_j \varphi(j|x)P(j|x)\right] p(x).$$

Integrating the above equation on both sides, it becomes

$$q(x) = \int \left[\sum_j \varphi(j|x)P(j|x)\right] p(x)dx\,.$$

From the inequality (1), it follows that

$$q(x) \leq \int max_j P(j|x)\; p(x)dx\,,$$

and this is the maximum correct classification probability of input x. It also gives us the equality of

$$argmax_j\; \varphi(j|x) = argmax_j\; p(j|x),$$

this indicates the predict class label of aggregating the predictors is the same as the classifier with the theoretical probability. That is, if the relative frequency of predictors make more predictions in class j than other classes, the actually input x also has more probability in class j than other classes.

However, it is not every input x will be in the right classifier by predictor $\emptyset$. Define the indicator function,

$$I = \begin{cases} 1, & if argmax_i\; \varphi(i|x) = j \\ 0, & if\; argmax_i\; \varphi(i|x) \neq j \end{cases}.$$

So the correct overall classification probability of aggregating of predictors is

$$g(x) = \int \sum_j I\,(argmax_i\; \varphi(i|x) = j)p(j|x) * p(x)dx\,.$$

Let M be the set of inputs x in the correct classifier with the predictor $\emptyset$, and M' is the set of inputs x in the other classifiers with the predictor $\emptyset$, the correct classification probability of input x by using aggregate predictor is

$$\int max_j P(j|x)\ p(x)dx(x \in M) + \int \sum_j I\,(argmax_i\ \varphi(i|x) = j)p(i|x) * p(x)dx\ (\mathrm{x} \in \mathrm{M}').$$

This means if there are more input x are predicted in the correct classification, it indicates the predictor tends to be more optimal.

3.3 Principle in Numerical Prediction

Suppose a learning set $L = \{x_i, y_i\}, i = 1,2, \ldots n$, n can be any positive integer and the response variable of y is numerical. Let a predictor $\emptyset(x, L)$ of this learning set predicts number for input x. As discussed above, the aggregate predictor $\emptyset_A(x)$ equals the average of the predictors $\emptyset(x, L)$, it denotes $\emptyset_A(x) = E_L\emptyset(x, L)$, with the expectation over L is denoted by E_L. Suppose y is the output value for the input x, the expectation

$$E_L[y - \emptyset(x, L)]^2 = E_L\left[y^2 - 2 * y * \emptyset(x, L) + \left(\emptyset(x, L)\right)^2\right]$$

$$= E_L[y^2] - 2E_L[y] * E_L[\emptyset(x, L)] + E_L[\emptyset(x, L)]^2.$$

As y is numerical, it changes to

$$E_L[y - \emptyset(x, L)]^2 = y^2 - 2yE_L[\emptyset(x, L)] + E_L[\emptyset(x, L)]^2 \qquad (1)$$

Since $variance(x) = E[x^2] - [E[x]]^2 \geq 0$, we obtain

$$E_L[y - \emptyset(x, L)]^2 - [E_L[y - \emptyset(x, L)]]^2 \geq 0,$$

With $\emptyset_A(x) = E_L\emptyset(x, L)$, it will change to $E_L[y - \emptyset(x, L)]^2 - [y - \emptyset_A(x)]^2 \geq 0$, thus

$$E_L[y - \emptyset(x, L)]^2 \geq [y - \emptyset_A(x)]^2 \qquad (2)$$

Applying (1) in (2),

$$y^2 - 2yE_L[\emptyset(x, L)] + E_L[\emptyset(x, L)]^2 \geq y^2 - 2y\emptyset_A(x) + [\emptyset_A(x)]^2,$$

The inequality becomes

$$E_L[\emptyset(x,L)]^2 \geq [\emptyset_A(x)]^2,$$

it also equals

$$E_L[\emptyset(x,L)]^2 \geq [E_L\emptyset(x,L)]^2 \qquad (3)$$

Let $p(x,y)$ is the joint distribution, integrating (3) by both sides,

$$\iint E_L[\emptyset(x,L)]^2 p(x,y)\, dxdy \geq \iint [E_L\emptyset(x,L)]^2\, p(x,y) dxdy$$

The new inequality becomes

$$E_L[\gamma(x,L)]^2 \geq [E_L\gamma(x,L)]^2$$

If $\gamma(x,L)$ changes substantially, there will be an improvement in classification. This is a case of an unstable regression tree. Additional information can be found in (Breiman, 1996).

CHAPTER 4. ADABOOST

AdaBoost is very simple to use and often improves given weak classifiers, and there are also tremendous weak classifiers can be chosen. The algorithm of AdaBoost and how to use the algorithm to calculate will be introduced in the following section. (Hertzmann & Fleet, 2011)

4.1 AdaBoost Algorithm

Definition 4.1.1 (*AdaBoost)*

AdaBoost is an algorithm for construction a "strong" classifier as linear combination

$$H(x) = sign[\sum_{t=1}^{T} \alpha_t h_t(x)]$$

of "simple" "weak" classifiers $h_t(x)$.

Suppose there is given a learning set $L = \{x_i, y_i\}, i = 1,2, \dots m$, m can be any positive integer, where $x_i \in R^k$ and $y_i \in \{-1,1\}$. Let $h_t(x)$ denotes the best weak classifiers at each iteration t, and make $h_t(x_i) \in \{-1,1\}$ with $t = 1,2, \dots T$.

As $t = 1\ to\ T$, the initial initially assign uniform weights $w_1(\mathrm{i}) = 1/m$ when $t = 1$. At each iteration t, find the best weak classifier $h_t(x)$ using weights $w_t(\mathrm{i})$.

Suppose there is a set of weak classifiers $\delta = \{h_j(x)\}$. Compute the error rate:

$$\in_t = \in_j = \sum_{i=1}^{m} w_t(i)[\![y_i \neq h_j(x_i)]\!],$$

where $$h_t(x) = argmin_{h_j \in \delta}(\in_j)\ .$$

When $\in_t \geq \frac{1}{2}$, it will stop to use the new method to classify.

Let $\alpha_t = \frac{1}{2} ln \frac{1-\in_t}{\in_t}$, the weights for each new weak classifier is

$$w_{(t+1)}(i) = \frac{w_t(\mathrm{i}) \exp(-\alpha_t y_i h_t(x_i))}{z_t},$$

Where z_t is the new normalization factor, and then the output for the final classifier:

$$H(x) = sign[\sum_{t=1}^{T} \alpha_t h_t(x)],$$

Where $h_t(x)$ denotes the number of weak classifiers at each iteration t.

As the algorithm is introduced above, the weights for each weak classifier is given by

$$w_{(t+1)}(i) = \frac{w_t(\mathrm{i})\exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}.$$

And ϵ_t needs to be less than ½ in the algorithm, $\alpha_t > 0$ is the condition to be satisfied with. It is easy to follow the inequality of $\exp(-\alpha_t y_i h_t(x_i))$.

$$\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} > 1, if\ y_i \neq h_t(x_i) \\ < 1, if\ y_i = h_t(x_i) \end{cases}.$$

The inequality follows the rule that if $y_i \neq h_t(x_i)$, it will increase the weight of wrong classifiers; if $y_i = h_t(x_i)$, it will decrease the weight of right classifiers. So AdaBoost focus on the informative examples.

So the following upper bound theorem holds. The theorem is due to maximize the training error in order to control the accuracy of the Algorithm. (Yoav Freund, Robert E. Schapire, 1996)

Definition 4.3.1 *(Upper bound Theorem)*

The following upper bound holds on the training error of H

$$\frac{1}{m}\sum_{i} [\![H(x_i) \neq y_i]\!] \leq \prod_{t=1}^{T} Z_t$$

Proof: As the weights function is

$$w_{(t+1)}(i) = \frac{w_t(\mathrm{i})\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$= \frac{w_{t-1}(i)\exp(-\alpha_{t-1} y_i h_{t-1}(x_i))}{Z_{t-1}} * \frac{\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$= \cdots$$

$$= \frac{w_1(\mathrm{i})\exp(\sum_t -\alpha_t y_i h_t(x_i))}{\prod_t Z_t},$$

As $w_1(\mathrm{i})$is the initial weight with $w_1(\mathrm{i}) = 1/\mathrm{m}$.

$$w_{t+1}(\mathrm{i}) = \frac{1}{m} * \frac{\exp(\sum_t -\alpha_t y_i h_t(x_i))}{\prod_t Z_t}.$$

Using the inequality above, when $h_t(x_i) \neq y_i$, so $y_i h_t(x_i) < 0$, and $-\alpha_t y_i h_t(x_i) > 0$, it is easy to follow

$$\exp(\sum_t -\alpha_t y_i h_t(x_i)) > 1.$$

Now we have

$$[\![H(x_i) \neq y_i]\!] \leq \exp(\sum_t -\alpha_t y_i h_t(x_i)),$$

Multiply $\frac{1}{m}$ both sides, it becomes

$$\left(\frac{1}{m}\right)[\![H(x_i) \neq y_i]\!] \leq \left(\frac{1}{m}\right)\exp\left(\sum_t -\alpha_t y_i h_t(x_i)\right)$$

$$\left(\frac{1}{m}\right)\sum_i [\![H(x_i) \neq y_i]\!] \leq \left(\frac{1}{m}\right)\sum_i \exp\left(\sum_t -\alpha_t y_i h_t(x_i)\right)$$

$$\left(\frac{1}{m}\right)\sum_i [\![H(x_i) \neq y_i]\!] \leq \prod_t Z_t * \sum_i w_{t+1}(\mathrm{i}).$$

As $\sum_i w_{t+1}(\mathrm{i}) = 1$, so

$$\left(\frac{1}{m}\right)\sum_i [\![H(x_i) \neq y_i]\!] \leq \prod_t Z_t.$$

Where does α_t come from? As we know, Z_t is the new normalization factor for each t, the function for Z_t is

$$Z_t = \sum_{i=1}^m w_t(i)\exp((-\alpha_t y_i h_t(x_i)),$$

If we take the derivative respect to each α_t and find the minimum value of Z_t

$$\frac{dZ}{d\alpha_t} = \sum_{i=1}^m w_t(i)[-y_i h_t(x_i)]\exp((-\alpha_t y_i h_t(x_i)) = 0$$

Using the equality function

$$y_i h_t(x_i) = \begin{cases} -1, if\ y_i \neq h_t(x_i) \\ 1, if\ y_i = h_t(x_i) \end{cases},$$

The derivative becomes

$$\sum_{y_i \neq h_t(x_i)} w_t(i)\exp(\alpha_t) - \sum_{y_i = h_t(x_i)} w_t(i)\exp(-\alpha_t) = 0\ .$$

Using the error rate,

$$\epsilon_t = \sum_{i=1}^{m} w_t(i)[y_i \neq h_t(x_i)]\ .$$

It becomes

$$\epsilon_t \exp(\alpha_t) - (1 - \epsilon_t)\exp(-\alpha_t) = 0\ ,$$

Solve for the equation, we can get

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}\ .$$

4.2 Example of Using AdaBoost

Below is an example of how the AdaBoost algorithm can be used to classify the teachers' efficiency. Suppose there is a given training data with six teachers. The Y value of -1 and classifies that the teacher does not make an improvement, and the Y value of 1 classifies the teacher does make an improvement. The X value can include any features that can be used to classify the teachers.

Table 3. An Example for AdaBoost

number	1	2	3	4	5	6
X value	1	2	3	4	5	6
Y value	-1	1	1	-1	-1	1

The weak learner $h_1(x)$ that minimizes the probability of error over the entire data, and the $y_1(x_i)$ are

Table 4. Weak Classifier Predicts the Value of Response Variable

Y value	-1	-1	1	-1	1	1

With initial weights are

Table 5. Initial Weights for the Response Variable

Initial weights	1/6	1/6	1/6	1/6	1/6	1/6

There are 2 teachers are not correctly classify by the first method, so

$$\in_1 = \sum_{i=1}^{m} w_1(i)[\![H(x_i) \neq y_i]\!] = 1/3,$$

$$\alpha_1 = \frac{1}{2} ln \frac{1-\in_1}{\in_1} = 0.3466,$$

And $$f_1(x) = 0.3466 * h_1(x).$$

The pre-normalized p_i is

Table 6. Pre-normalized Probability for the Response Variable

Pre-normalized p_i	Index1= 0.1178	Index2= 0.2358	Index3= 0.1178	Index4= 0.1178	Index5= 0.2358	Index6= 0.1178

Let us pull out the index 1 and index2 to show how the calculation working in this problems.

Pre-normalized

$$p_i = \frac{w_1(\mathrm{i})\exp(-\alpha_1 y_i h_t(x_i))}{1},$$

And

$$p_1 = \left(\frac{1}{6}\right) * \exp\left(-0.3466 * (-1) * (-1)\right) = 0.1178,$$

$$p_2 = \left(\frac{1}{6}\right) * \exp\left(-0.3466 * (-1) * (1)\right) = 0.2358.$$

The normalization factor $z_1 = \sum_{i=1}^{6} p_i$= 0.9468, the new weight is

$$w_2(i) = \frac{w_1(\mathrm{i})\exp(-\alpha_1 y_i h_t(x_i))}{z_1}$$

Showed in the table 7 below:

Table 7. The New Weights for the Response Variable

$w_2(i)$	Index1= 0.1244	Index2= 0.2490	Index3= 0.1244	Index4= 0.1244	Index5= 0.2490	Index6= 0.1244

If the above steps are repeated till the $\in_t \geq \frac{1}{2}$, it will stop. Suppose there are 3 methods used for this example, the final classifier: $H(x) = sign[\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)]$, If $H(x) > 0$, it is a strong classifier and indicates this teacher does make an improvement; if $H(x) < 0$, it is a weak classifier and indicates this teacher does not make an improvement with the students.

CHAPTER 5. DISCUSSION AND CONCLUSION

Decision trees are widely used for solving classification problems as they can handle mixed types of variables to classify data or perform numerical predictions. Breiman addressed "it has the potential for being a powerful and flexible classification tool". Since the final decision trees are in a simple form which includes the original data and also can do the efficiency classifiers for the new data, it is very useful and competitive to use the conditional information to do the classifiers (Breiman, 1984). Deng et al (2011) pointed out that if the data includes the categorical variable with the number of levels in decision trees, not in binary levels, the information gain will be biased and calculation can be very complex if many values are uncertain. However, Nayab(2011) suggests that the instability is also another problem for a decision tree, even if there is a small change in the input x, it can change the tree significantly by using the original data. In order to avoid the unstable problem in decision trees, Brieman (1996) explored the method called bagging predictor. It can improve an unstable and weak classifier to a better one. Dong et al (2006) indicated that the bagging predictor focuses on the global accuracy to get the average of classification accuracy; it may obtain less over fitting. However, Breiman (1996) provided evidence that bagging predictors do not perform well with stable data. In order to make a better frame for doing classification, AdaBoost became the most popular algorithm in Boosting; it is good for outliers. It was formed by Yoav Freund and Robert Schapire. Freund and Schapire (2011) indicated AdaBoost is simple and easy to program and can provide consistently effective results according to the rules. It can combine many classification methods and be "less susceptible to the over fitting problems than most learning algorithm." AdaBoost with decision tree is considered to be the best classifier. As AdaBoost is used to form a linear combination of weak classifiers, if weak learners are quite strong, the AdaBoost may not be effective to do the

classification. If hypotheses are too complex, the test error might be much larger than the training error (Freund, 2011). Comparison of the above three popular methods in doing classification, the properties of the data decide which methods will be most accurate and efficient. With the limitations of the data, and based on the different criteria for the school districts across the nation, the methods described can be used to classify teacher efficiency, and to improve existing classifiers.

REFERENCES

Abdi, H. (2007). discriminant Correspondence. *Encyclopedia of Measuremnt and Statistics*, 270-275.

Aleix M. Martinez, A.C. Kak. (Feb 2001). PCA versus LDA. *IEEE transactions on Pattern Analysis and Machine Intelligence, 23*, 228-233.

Cpver, T. (June 1965). Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognioin. *Electronic Computers, IEEE Transactions on, 14*, 326-334.

George R. Terrell, David W. scott. (1992). variable Kernel Density Estimation. *annals of Statistics, 20*, 1236-1265.

Houtao Deng, George runger, Eugene Tuv. (2011). Bias of importance measures for multi-valued attributes and solutions. *Proceedings of he 21st international conference on Artificial neural networks. 2*, pp. 293-300. Heidelberg: Springer-Verlag Berlin.

Introduction to Data Mining. Pang-Ning Tan, Michael Steinbach,Vipin Kumar. (March 25th, 2006). chapter 4 Classification,basic concepts, decision trees and model evaluation. In M. S. Pang-Ning Tan, *Introduction to data mining* (pp. 155-164). Minnesota : Addison Wesley company.

Joseph J Guido, Paul C Winters, Adam B Rains. (2006). Logistic Regression Basics. *NESUG*.

kohavi, R. (1995). a study of cross-validation and bootsrp for accuracy estimation and model selection. *Proceedings of the 14th international join conference on Artificial intelligence. 2*, pp. 1137-1143. San Francisco: Morgan Kaufmann Publishers Inc.

Leo Breiman, Jerome H. Freidman, Richard A. Olshen, Charles J. Stone. (1984). *Classification and Regression trees*. Belmont, California: Wadsworth International Group.

Leo Breiman (1996). Bagging Predictor. *Machine Learning, 24*, 123-140.

Liuhuan Dong, Yuan yuan, Yudong Cai. (2006). Using baggin Classifier to Predict Protein Domain Structural Class. *Biomolecular Structure & Dynamics, 24*, 241-242.

Williams, G. (2011). *Data Mining with Rattle and R*. New York: Springer Science+Business Media, LLC.

Yoav Freund, Robert E. Schapire. (1996). Experiments with a New soosting Algorithm. *Machine Learning: Proceedings of he Thirteenth Internatinal Conference*.

APPENDIX A. EXAMPLE OF ORIGINAL DATA

{14003433283, {{"733421221", {{119, 276}, {175, 245}, {204, 221}, {225, 274}, {234, 281}, {234, 306}, {242, 255}, {249, 284}, {249, 287}, {249, 292}, {255, 301}, {255, 310}, {261, 274}, {261, 290}, {261, 306}, {261, 332}, {267, 264}, {267, 315}, {273, 307}, {273, 308}, {273, 316}, {283, 307}, {283, 308}, {283, 309}, {283, 316}, {283, 318}, {283, 329}, {288, 301}, {288, 306}, {288, 331}, {293, 311}, {293, 331}, {298, 318}, {298, 336}, {298, 338}, {298, 344}, {302, 297}, {302, 309}, {302, 315}, {302, 342}, {307, 306}, {307, 318}, {307, 325}, {311, 327}, {316, 304}, {316, 329}, {325, 277}, {325, 317}, {325, 326}, {325, 328}, {325, 334}, {325, 334}, {325, 358}, {329, 340}, {329, 363}, {333, 325}, {333, 333}, {338, 334}, {338, 346}, {338, 363}, {342, 328}, {342, 334}, {342, 339}, {351, 367}, {351, 380}, {356, 346}, {356, 346}, {356, 375}, {365, 344}, {370, 344}, {370, 353}, {370, 367}, {370, 371}, {381, 367}, {395, 347}, {402, 343}, {402, 380}, {450, 393}}}, {"smcl", {{267, 274}}}}}}

APPENDIX B. ORGANIZED DATA IN SPREADSHEET

Teacher	pretest average score	posttest average score	pretest sd	posttest sd	difference sd	post-pre	number of students	percentage going up	bin
1	474.7	421.4	46.58	46.38	49	-53.3	10	0.3	0
2	453.73	400.52	42.71	54.43	63.57	-53.21	33	0.181818182	0
3	428.51	384.56	44.83	32.27	35.74	-43.95	71	0.112676056	0
4	447.37	407.03	42.74	43.28	37.23	-40.34	76	0.131578947	0
5	414.31	374	42.94	22.67	32.57	-40.31	16	0.125	0
6	428.69	388.58	54.71	43.44	49.13	-40.11	26	0.230769231	0
7	403.23	363.77	24.41	21.48	20.57	-39.46	13	0	0
8	428.59	390.76	47.89	46.02	35.45	-37.83	68	0.147058824	0
9	409.9	373.73	55.09	33.3	42.6	-36.17	79	0.215189873	0
10	424.2	388.96	43.86	39.98	34.21	-35.24	25	0.08	0
11	416.82	382.18	47.51	37.33	46.08	-34.64	22	0.272727273	0
12	387.85	353.31	36.01	16.8	25.63	-34.54	13	0	0
13	407.28	372.78	47.27	24.91	43.02	-34.5	18	0.166666667	0
14	455.14	420.72	50.72	49.33	39.36	-34.42	36	0.25	0
15	403.36	369.06	49.8	33.5	42.59	-34.3	124	0.193548387	0
16	409	375	54.87	15.67	47.35	-34	14	0.214285714	0
17	397.96	364.06	42.18	26.14	30.72	-33.9	78	0.076923077	0
18	437.65	404.06	54.95	48.72	53.05	-33.59	17	0.352941176	0
19	409.9	376.8	59.55	53.17	27.28	-33.1	10	0	0
20	399.26	366.17	54.08	28.86	42.2	-33.09	23	0.173913043	0
21	428.79	396.32	43.37	37.11	45.12	-32.47	28	0.214285714	0
265	338.5	344.08	36.65	37.04	19.42	5.58	12	0.75	1
266	326.7	332.3	33.98	15.25	22.85	5.6	10	0.6	1
267	320.36	325.98	39.38	23.85	26.01	5.62	88	0.579545455	1
268	355.44	361.17	49.3	43.55	34.21	5.73	18	0.5	1
269	333.76	339.59	42.88	33.72	29.17	5.83	59	0.593220339	1
270	333.64	339.57	50.24	33.6	32.15	5.93	75	0.6	1
271	301.29	307.43	41.35	28.05	22.35	6.14	14	0.642857143	1
272	335.67	341.93	52.16	35.22	31.91	6.26	30	0.533333333	1
273	345.93	352.2	50.72	37.25	30.34	6.27	107	0.663551402	1
274	320.36	326.75	31.06	23.05	18.81	6.39	28	0.642857143	1
275	344.03	350.44	52.71	42.78	36.16	6.41	32	0.5625	1
276	327	333.41	37.9	30.33	22.64	6.41	17	0.588235294	1
277	320.71	327.18	32.74	28.76	22.99	6.47	34	0.558823529	1
278	333.8	340.32	45	25.28	29.93	6.52	119	0.579831933	1
279	352.3	359.09	56.47	37.42	30.08	6.79	23	0.652173913	1
280	326.53	333.34	54.63	29.38	38.62	6.81	100	0.56	1
281	338.77	345.59	57.99	30.66	34.14	6.82	22	0.590909091	1

VITA

Xuan Wang, a native of Jiangsu, China, received her bachelor's degree at the Louisiana State University and Agricultural and Mechanical in 2011. Thereafter, as her great interest in mathematics, she made the decision to enter graduate school to keep studying in the Department of Mathematics at Louisiana State University. She will receive her master's degree in August 2013 and plans to begin work after she graduate.