Assignment 3: Decision Tree Classification Using R

Written by Daanish Ahmed

DATA 630 9041

Semester Summer 2017

Professor Edward Herranz

UMUC

July 3, 2017

Introduction

Decision trees are an extremely effective method for predicting outcomes within a dataset. They consist of nodes representing certain variables that are split into branches indicating the possible values of that variable. By traversing the tree from the root to the terminal nodes (or leaves), we can predict the value of the dependent variable and see the impact that other variables have towards its outcome. Some of the benefits to using decision trees are that they are computationally cheap to build, easy to understand, and can handle issues such as missing values and irrelevant data (Bati, 2015). However, this method has three shortcomings: a vulnerability to overfitting, bias when handling unbalanced datasets, and instability due to variance in the data (Bati, 2015). These issues will be addressed when I apply this model to a dataset on bank marketing information using R software. The goal of my model is to evaluate the likelihood of a client subscribing for a term deposit ("Bank Marketing Data Set," 2012) and determine the impact that other variables have towards the outcome. I will address overfitting by building the model on training and test data and pruning unnecessary branches. Afterwards, I will evaluate the model's accuracy and handle any bias through oversampling the minority class of the dependent variable. Finally, I will address the issue of high variance by implementing the random forest algorithm on the oversampled dataset. I expect that each of these steps will improve the accuracy of my model, resulting in a set of rules that contains useful insights for banking strategies.

Data Exploration

I will begin my analysis with a description of the original dataset. This dataset contains 17 variables with 4521 observations (see Figure 1 in Appendix B). Some of these variables include

the client's age, job, marital status, education level, default credit, yearly balance, housing loans, personal loans, last contact day and month, and previous campaigns ("Bank Marketing Data Set," 2012). The dependent variable in my model is "subscribed term deposited," which is a factor with the values "yes" and "no." Five of the variables, such as age and last contact day, are integer variables. The remaining variables are all factors, and most of them have between 2 and 12 levels. Two of the variables have more than 200 levels—this can be problematic for the model and it will be addressed during preprocessing. For further exploration, I generated a set of descriptive statistics from the dataset (see Figure 2 in Appendix B). We see that the median client age is 39, most clients are married and have attained at least a secondary education, and most have housing loans but no personal loans. These statistics reveal that there are no missing values in any of the variables, which would not have mattered to our model since decision trees are designed to handle missing values (Bati, 2015). Furthermore, we notice that the vast majority of clients have not subscribed for a term deposit, meaning that the dependent variable is unbalanced. This will likely result in a decision tree that is biased towards predicting the most common value of "no." I will address this issue in another section of the paper, in which I create a new model that oversamples the minority class and compare the outcome to that of my initial model.

Preprocessing

Before building the model, I performed several preprocessing steps to prepare the data for implementation with the algorithm. Firstly, the decision tree classification method in R requires us to use the "party" package. I installed this package and activated it alongside some of the other packages I am using in my code. The next step would be to remove unique identifiers, since they are not useful to the analysis and can even affect the results of the decision tree algorithm ("Cars

Classification," 2017). But since there are no ID variables in the dataset, we do not need to worry about this step. Additionally, the dependent variable "subscribed term deposited" is already a factor type, thus we do not need to convert it into one. As stated before, the decision tree algorithm is not affected by missing values (Bati, 2015), meaning that we would not have to handle them if they were present in this dataset. The last step is to remove variables that can cause problems with the decision tree algorithm. The variables "yearly balance" and "passed days" are factors with 2294 and 292 levels respectively, meaning that they are less useful for the analysis and may impact the quality of the decision tree. Additionally, not all of their values are in number format—some of the values consist of question mark symbols. Therefore, these variables cannot easily be converted into numeric form and it is better to delete them. With this step completed, the dataset is ready for implementation with the decision tree method.

Decision Tree Model

I will now describe the process of creating the decision tree classification model using my dataset. The first part of this process is to split the dataset into a training set containing 70% of the data and a test set with 30% of the data. This is necessary because it prevents overfitting, which occurs when we include too many branches in the tree—including outliers and branches with unnecessary information. Overfitting is a problem because it impacts the accuracy of our decision tree model (Han, Kamber, & Pei, 2011). But by splitting the data into training and test sets, we reduce the amount of data included in the tree—therefore lowering the chance of overfitting. It is still possible that overfitting may occur after splitting the dataset, thus it is necessary to prune the model of outliers and unnecessary branches. However, the "ctree" method in R can automatically prune the decision tree. According to the R documentation on Conditional

Inference Trees (obtained using the "help" command in RStudio), the "ctree" method builds the tree to its optimal size such that no form of pruning is required. Therefore, we can use this method to easily implement our decision tree model on the training data. For the input parameters, I used "subscribed term deposited" as the dependent variable and all remaining variables as independent variables. The reason why I included all remaining variables is that I already removed the unnecessary or problematic variables during preprocessing.

Now that the decision tree has been built, I printed the entire model in textual format (see Figure 3 in Appendix B). We see that there are 15 terminal nodes (or leaves) in the tree, and there are 3181 observations in the training set. We also see that there are 29 nodes in the model, and each node shows a description of how the tree was split and the nodes that each split lead to. For a more visual depiction of the model, I created a graph of the decision tree (see Figure 4 in Appendix B). This graph contains much of the information that was included in the textual version of the tree—such as the splitting variables and the branches that each split lead to. But for each terminal node, we also see the percentage of instances that have the values of "yes" (dark) and "no" (light) for the dependent variable. A more simplified version of this tree shows the exact percentages for the values in parenthesis under each leaf, with "no" on the left and "yes" on the right (see Figure 5 in Appendix B). In node 5, for instance, there is a 79% chance that that a client will subscribe for a term deposit and a 21% chance that they will not. In node 16, however, there is a 0% chance that the client will subscribe for a term deposit. Finally, I generated confusion matrices on the training and test datasets to evaluate the classification accuracy of the model (see Figures 6 and 7 in Appendix B). Based on these figures, we find that the accuracy on the training data is 89.8%, while the accuracy on test data is 90.2%. These results seem to indicate that the model is very accurate, but I will describe the details in the next section.

Initial Results

By examining these findings, we can gain insight on the relationships between the variables as well as the accuracy of the overall model. I will begin by examining the results from the decision tree visualizations (see Figures 4 and 5 in Appendix B). Firstly, we see that the variable "last contact duration in seconds" is one of the most important splitting variables since it is used for the root node as well as nodes 3, 7, 12, and 26. This suggests that the length of the phone call is a good indicator of whether the marketing effort succeeded at persuading a customer to subscribe for a term deposit. For calls lasting over 645 seconds (or roughly 10 minutes), there is a 50% chance that the client will subscribe for a term deposit. The second node in the tree implies that the outcome of the previous marketing campaign is also a key variable for predicting the value of the dependent variable. Node 5, which contains 66 records, has a 79% chance that that a client will subscribe for a term deposit—giving it the highest likelihood of success out of all leaves in the tree. Achieving this rate of success requires the previous marketing campaign to have been successful and for the previous contact call to have lasted between 159 and 645 seconds. But even if the previous campaign was a success, there is only an 18% chance of the client subscribing if the last phone call was less than 159 seconds long. Other interesting findings in this model come from the marital status and job variables. Married customers only have a 5% chance of subscribing while single and divorced customers have a 40% chance of doing so, given that the conditions for node 8 are satisfied. And according to node 14, students are more likely to subscribe than individuals working in any paying job (see Figures 3 and 5 in Appendix B).

There are two leaves that indicate a 0% chance of a client subscribing and one with only a 3% chance of this happening. These leaves (nodes 16, 17, and 27) have between 183 and 1128 observations, meaning that they are more likely to be accurate. Some of the variables that

contribute to these low values are job type, last contact type, last contact duration, and last contact month. Based on my previous findings, I believe that job type and last contact duration are good indicators of whether a client will subscribe. However, I do not think that last contact type is a strong metric for measuring this value. Last contact type has three possible outcomes (cellular, telephone, and unknown), and the three nodes with the lowest success rates fall under each of these three outcomes. Last contact month might be an effective metric for measuring client subscription, but it is unclear why this is the case. The value of the dependent variable differs according to the last contact month, but some of the months that are grouped together in the decision tree (such as March and October) are very far apart. This seems to imply that the dependent variable may not depend on the number of months since the last contact attempt, but rather on the specific time of year. Finally, there were several independent variables that were not included as splitting variables in the creation of this tree. For instance, the client's age, default credit, housing loans, and personal loans were all omitted from the model. This suggests that such variables were less effective for building an accurate model compared to the variables that were used.

The final part of analyzing the initial model is to examine the confusion matrices for the training and test datasets. I found that the classification accuracies of the training and test sets were 89.8% and 90.2% respectively. These results seem to indicate that the model is very accurate at predicting the values of the dependent variable. But when looking at the matrices themselves (see Figures 6 and 7 in Appendix B), we see that there are very few predictions that accurately predicted "yes." The vast majority of cases accurately predicted the value "no." This suggests that the model is biased towards predicting the majority class of "no," which explains its high accuracy rate. To address this issue, I will recreate this model and oversample the minority class in my dataset to see if this will produce a more accurate decision tree.

Oversampling

Oversampling is a process that involves adding multiple instances of the minority class into the dataset ("Class Imbalance Problem," 2013). As a result, this process will reduce the bias of the decision tree by increasing the likelihood that the algorithm will predict the minority class. Implementing the oversampling method in R requires users to install and activate the "unbalanced" package (Pozzolo, Caelen, & Bontempi, 2015). For preprocessing, the function requires the dependent variable to be a factor with the values 0 and 1 for the majority and minority classes respectively (Pozzolo et al., 2015). Once I converted the values of my dependent variable, I implemented the oversampling function on my dataset using "subscribed term deposited" as the response variable and all remaining variables as predictors. Next, I split the oversampled dataset into training and test sets and implemented the decision tree algorithm—following the same steps that I described when building the original decision tree. After building the tree, I printed the model's textual format (see Figures 8 and 9 in Appendix B). This model is many times more complex than my original tree, since it has 93 leaves and a total of 185 nodes. Variables that were used as splitting variables in the original model—such as last contact duration and the outcome of previous campaigns—are still used as splitting variables throughout the new model. However, some variables that were not used as splitting variables in the previous model (such as education and current campaign contacts) are now being used as such.

To visualize the results, I created a simplified graph of the oversampled decision tree (see Figure 10 in Appendix B). This tree is extremely complex compared to the original model, and it is difficult to read due to the number of nodes and branches. But as stated before, the "ctree" method in R automatically prunes the branches—meaning that there is no need to remove any additional branches. And though the tree is more complex, the results indicate a much greater

likelihood of predicting customers who will subscribe for a term deposit. To evaluate the accuracy of my new model, I generated confusion matrices for the training and test datasets (see Figure 11 and 12 in Appendix B). Based on these figures, we see that the classification accuracies for the training and test sets are 89.8% and 87.6% respectively. The training set's accuracy is identical to that of the original model, but the test set's accuracy is slightly lower. However, these matrices indicate that the model is no longer biased, since there are roughly equal numbers of predictions for the majority and minority classes. Overall, this approach seems to be an effective way to deal with bias within this algorithm. But the consequence of using this method is that the model loses some of its accuracy. Since oversampling consists of duplicating records with the minority class, it can cause the algorithm to overfit the data to a handful of records—therefore causing the predictions to move further from the expected value ("Class Imbalance Problem," 2013). For future research, I would recommend exploring other approaches for handling unbalanced datasets to find a method that maximizes the accuracy of the model.

Random Forests

The last issue with decision trees is that they tend to be unstable due to high variance. This means that the algorithm often produces different results for the training and test datasets (James, Witten, Hastie, & Tibshirani, 2013). One way to reduce the variance of my model is through using random forests. According to James et al. (2013), random forests involve building multiple decision trees using repeated samples from the training set and choosing a random selection of variables as splitting variables. The authors state that this method generates trees that are not correlated with each other, resulting in a lower average variance between the trees. Use of random forests in R requires the "randomForest" package to be installed and activated. After enabling this

package, I implemented the algorithm using the training data from the oversampled dataset. For the input values, I used "subscribed term deposited" as the dependent variable, all other variables as independent variables, and a forest size of 25 trees. I generated the model's output, showing many of the forest's properties and a confusion matrix for the training data (see Figure 13 in Appendix B). The figure indicates that the forest is a classification model, there are 25 trees, and there are 3 random variables used for each split. Additionally, we see that the error rate is 4.32%, and the confusion matrix indicates that the model has a 95.7% accuracy rate.

Finally, I produced a confusion matrix for the random forest model on test data (see Figure 14 in Appendix B). This matrix reveals that the model's classification accuracy on the test set is 95.7%, which is equivalent to the accuracy on the training set. This indicates that the results are very similar between the training and test sets, which means that the variance has been greatly reduced. We can also see that the model's accuracy has been significantly improved, especially when compared to the previous two models. Additionally, there are roughly even numbers of predictions for the majority and minority classes in both matrices. This reflects the properties of the oversampled dataset that was used to create this model. Altogether, the random forest method has been successful in not only lowering the model's variance, but also improving its accuracy. Though there may be more accurate models for handling variance, the current method has nevertheless proven effective in this study.

Conclusion

The results of my analysis have fulfilled many of the goals that I had described at the beginning of the paper. First, I addressed the issue of overfitting during the model's creation by

implementing the decision tree algorithm on training and test data. By analyzing the initial tree, I found that the model provided many useful insights for improving bank marketing strategies. Certain variables such as the last contact duration, the outcome of the previous marketing campaign, and the customer's job and marital status can help to determine the likelihood of a customer subscribing for a term deposit. Next, I reduced the model's bias by overfitting the minority class of the dependent variable and implementing the decision tree algorithm on the new balanced dataset. Finally, I reduced the model's variance by implementing a random forest algorithm on the overfitted data. I found that using these steps not only helped to eliminate the issues mentioned, but also improved the model's accuracy as well.

One of the limitations in this study was the fact that the original dataset was unbalanced, which I handled by oversampling the minority class. However, I also found that the oversampling method itself had limitations with regards to accuracy. Since oversampling involves creating copies of the minority class, it can cause the predictions to differ from the expected value ("Class Imbalance Problem," 2013). This occurred to some degree in my model, as evidenced by the fact that the predictions for the oversampled model were slightly less accurate than those of the original model. Therefore, I would suggest exploring other methods for dealing with unbalanced datasets for future research. It is likely that using a different sampling method, or even a combination of methods, may yield more accurate predictions. And even though my random forest model was extremely accurate, it is possible that there still might be more accurate methods for reducing variance as well. I expect that the exploration of additional techniques will eventually result in a more accurate model that can yield vital information on bank marketing strategies.

References

Bank Marketing Data Set. (2012, February 14). Retrieved July 1, 2017, from https://archive.ics.uci.edu/ml/datasets/Bank Marketing

Bati, F. (2015, Fall). Classification using Decision Tree. Lecture presented at UMUC. Retrieved June 28, 2017.

Cars Classification via Decision Tree – Conditional Inference Tree Model (2017). Retrieved June 26, 2017.

Class Imbalance Problem. (2013, August 30). Retrieved July 05, 2017, from http://www.chioka.in/class-imbalance-problem/

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques* (3rd ed.). Retrieved June 27, 2017.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. Retrieved June 28, 2017.

Pozzolo, A. D., Caelen, O., & Bontempi, G. (2015, June 26). Package 'unbalanced'. Retrieved July 3, 2017, from https://cran.r-project.org/web/packages/unbalanced/unbalanced.pdf

Appendix A

R Script for Assignment 3

```
# DATA 630 Assignment 2

# Written by Daanish Ahmed

# Semester Summer 2017

# June 17, 2017

# Professor Edward Herranz


# This R script creates a decision tree classification model using a dataset

# containing information on bank marketing data.  The purpose of this assignment

# is to generate a model that predicts the value of the dependent variable

# "suscribed.term.deposited" using the other variables as independent variables.

# Additionally, the script will evaluate the model's accuracy and attempt to

# improve it by oversampling the minority class (to deal with bias) and construct

# a random forest (to deal with high variance).  The script consists of several

# components, including opening and initializing the data, exploration and

# preprocessing, building and analyzing the decision tree, oversampling, and

# applying the random forest algorithm to the oversampled dataset.


# This section of code covers opening the dataset and initializing the packages

# that are used in this script.
```

```
# Sets the working directory for this assignment.  Please change this directory
# to whichever directory you are using, and make sure that all files are placed
# in that location.
setwd("~/Class Documents/2016-17 Summer/DATA 630/R/Assignment 3")

# In order to run the commands for the decision tree, oversampling, and random
# forests, we need to use the following packages:

# If you have not installed these packages yet, remove the three # symbols below.
# install.packages("party")
# install.packages("unbalanced")
# install.packages("randomForest")

# Loads the necessary packages into the system.
library("party")
library("unbalanced")
library("randomForest")

# Opens the CSV file "bank-marketing.csv".
bank <- read.csv("bank-marketing.csv", head=TRUE, sep=",")

# End of opening the dataset.
```

```
# This section of code covers data preprocessing.  It includes exploration of
# the original dataset, removing variables, and dealing with missing values.

# Previews the bank marketing dataset.
View(bank)

# Shows the descriptive statistics for all variables in the dataset.
summary(bank)

# Displays the structure of the data.  This is necessary to see if there are
# any unique identifiers (IDs) or other variables that can be removed.
str(bank)

# There are no unique identifier variables in this dataset, thus we do not
# need to remove them.

# The dependent variable "suscribed.term.deposited" is already a factor,
# thus we do not need to convert it into one.

# The variables "yearly.balance" and "passed.days" are factors with 2294 and
# 292 levels respectively, meaning that they aren't useful for the decision
```

```
# tree algorithm.  Also, not all of their values are numbers (some of them
# have "?" symbols as values).  Thus, we remove them.
bank <- bank[, -c(6, 14)]

# Verifies that the unwanted variables have been removed.
str(bank)

# The decision tree algorithm is able to handle missing values.  Therefore,
# we do not need to check for missing values.

# End of data preprocessing.


# This section of code covers the creation of the decision tree classification
# model.  It includes dividing the data into training and test datasets,
# creating and visualizing the model, and analyzing the model on the training
# and test datasets.

# Generates a random seed to allow us to reproduce the results.
set.seed(1234)

# The following code splits the bank marketing dataset into a training set
# consisting of 70% of the data and a test set containing 30% of the data.
```

```
ind <- sample(2, nrow(bank), replace = TRUE, prob = c(0.7, 0.3))
train.data <- bank[ind == 1, ]
test.data <- bank[ind == 2, ]

# The following code creates the decision tree classification model using the
# training data with suscribed.term.deposited as the dependent variable and all
# other variables as independent variables.
myFormula <- suscribed.term.deposited ~ .
model <- ctree(myFormula, data = train.data)

# Prints the decision tree model.
print(model)

# Displays the visualization of the decision tree.
plot(model)

# Plots a simplified version of the decision tree.
plot(model, type="simple")

# Displays the confusion matrix on the model for training data, allowing
# us to see the model's accuracy.
table(predict(model), train.data$suscribed.term.deposited)
```

```
# Displays the confusion matrix on the model for test data.
testPred <- predict(model, newdata = test.data)
table(testPred, test.data$suscribed.term.deposited)

# End of creating the decision tree model.


# This section of code covers the oversampling of our original dataset.
# The original model appears to be biased towards predicting the majority
# class of the dependent variable.  We will oversample the minority class
# to make the dataset less biased and apply the decision tree algorithm
# to this dataset.

# Converts the dependent variable into a character type so that we can
# replace the variable's values.
bank$suscribed.term.deposited <- as.character(bank$suscribed.term.deposited)

# The 'ubOver' function in the 'unbalanced' package requires the dependent
# variable to be a factor with the values 0 and 1.  Thus, we replace the
# values of our dependent variable.
bank$suscribed.term.deposited[bank$suscribed.term.deposited == 'no'] <- '0'
bank$suscribed.term.deposited[bank$suscribed.term.deposited == 'yes'] <- '1'
```

```
# Converts our dependent variable back into a factor.
bank$suscribed.term.deposited <- as.factor(bank$suscribed.term.deposited)

# The following code is from Package 'unbalanced'
# Based on Pozzolo, Caelen, & Bontempi (2015)
# https://cran.r-project.org/web/packages/unbalanced/unbalanced.pdf
# Modified for UMUC DATA 630 by Daanish Ahmed

# Implements the 'ubOver' function from the 'unbalanced' package so that
# we can oversample the minority class of our dependent variable.  Uses
# "suscribed.term.deposited" as the dependent variable and all other
# variables as the independent variables.
output <- bank$suscribed.term.deposited
input <- bank[ ,-15]
data <- ubOver(X=input, Y= output)
bank <- cbind(data$X, data$Y)

# End of Pozzolo, Caelen, & Bontempi code.

# We give the dependent variable its original name.
names(bank)[15] <- "suscribed.term.deposited"

# Generates a random seed to allow us to reproduce the results.
```

```
set.seed(1234)

# The following code splits the oversampled dataset into a training set
# consisting of 70% of the data and a test set containing 30% of the data.
ind <- sample(2, nrow(bank), replace = TRUE, prob = c(0.7, 0.3))
train.data <- bank[ind == 1, ]
test.data <- bank[ind == 2, ]

# The following code creates the decision tree classification model using the
# oversampled training data with suscribed.term.deposited as the dependent
# variable and all other variables as independent variables.
myFormula <- suscribed.term.deposited ~ .
model <- ctree(myFormula, data = train.data)

# Prints the decision tree model.
print(model)

# Displays the visualization of the decision tree (may take a few seconds).
plot(model)

# Plots a simplified version of the decision tree (may take a few seconds).
plot(model, type="simple")
```

```
# Displays the confusion matrix on the model for training data, allowing
# us to see the model's accuracy.
table(predict(model), train.data$suscribed.term.deposited)

# Displays the confusion matrix on the model for test data.
testPred <- predict(model, newdata = test.data)
table(testPred, test.data$suscribed.term.deposited)

# End of creating the oversampled decision tree model.



# This section of code covers the implementation of the random forest
# algorithm on the oversampled dataset.  This algorithm is used to reduce
# the variance of our decision tree model.

# Generates a random seed to allow us to reproduce the results.
set.seed(1234)

# Implements the random forest algorithm on the training data using
# "suscribed.term.deposited" as the dependent variable and all other
# variables as independent variables.  We will create 25 trees.
forest <- randomForest(suscribed.term.deposited ~ ., data = train.data,
                       ntree = 25, importance = TRUE)
```

```
# Displays the output of the random forest, including the type of forest,
# number of trees, number of variables used for each split, confusion
# matrix, and error rate.
forest

# Displays the confusion matrix on the model for test data.
forest.pred <- predict(forest, newdata = test.data)
table(forest.pred, test.data$suscribed.term.deposited)

# End of creating random forest.

# End of script.
```

Appendix B

Relevant R Output Images

```
> str(bank)
'data.frame':   4521 obs. of  17 variables:
 $ age                          : int  30 33 35 30 59 35 36 39 41 43 ...
 $ job                          : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
 $ marital.status               : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
 $ education                    : Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
 $ default.credit               : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ yearly.balance               : Factor w/ 2294 levels "-2","-20","-200",..: 736 1626 490 565 216 2023 1210 561 917 195 ...
 $ housing.loan                 : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
 $ personal.loan                : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
 $ last.contact.type            : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
 $ last.contact.day             : int  19 11 16 3 5 23 14 6 14 17 ...
 $ last.contact.month           : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
 $ last.contact.duration.seconds: int  79 220 185 199 226 141 341 151 57 313 ...
 $ current.campaign.contacts    : int  1 1 1 4 1 2 1 2 2 1 ...
 $ passed.days                  : Factor w/ 292 levels "?","1","100",..: 1 189 180 1 1 67 180 1 1 42 ...
 $ previous.campaigns           : int  0 4 1 0 0 3 2 0 0 2 ...
 $ outcome.previous.campaign    : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
 $ suscribed.term.deposited     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
>
```

Figure 1. Initial Data Structure of Bank Marketing Dataset.

```
> summary(bank)
      age                job       marital.status     education    default.credit yearly.balance housing.loan
 Min.   :19.00   management :969   divorced: 528   primary  : 678   no :4445       0      : 359   no :1962
 1st Qu.:33.00   blue-collar:946   married :2797   secondary:2306   yes:  76       2      :  26   yes:2559
 Median :39.00   technician :768   single  :1196   tertiary :1350                  1      :  22
 Mean   :41.17   admin.     :478                   unknown  : 187                  4      :  14
 3rd Qu.:49.00   services   :417                                                   5      :  14
 Max.   :87.00   retired    :230                                                   23     :  12
                 (Other)    :713                                                   (Other):4074
 personal.loan last.contact.type last.contact.day last.contact.month last.contact.duration.seconds
 no :3830      cellular :2896    Min.   : 1.00    may    :1398       Min.   :   4
 yes: 691      telephone: 301    1st Qu.: 9.00    jul    : 706       1st Qu.: 104
               unknown  :1324    Median :16.00    aug    : 633       Median : 185
                                 Mean   :15.92    jun    : 531       Mean   : 264
                                 3rd Qu.:21.00    nov    : 389       3rd Qu.: 329
                                 Max.   :31.00    apr    : 293       Max.   :3025
                                                  (Other): 571
 current.campaign.contacts  passed.days   previous.campaigns outcome.previous.campaign suscribed.term.deposited
 Min.   : 1.000             ?      :3705   Min.   : 0.0000    failure: 490              no :4000
 1st Qu.: 1.000             182    :  23   1st Qu.: 0.0000    other  : 197              yes: 521
 Median : 2.000             183    :  20   Median : 0.0000    success: 129
 Mean   : 2.794             363    :  12   Mean   : 0.5426    unknown:3705
 3rd Qu.: 3.000             92     :  12   3rd Qu.: 0.0000
 Max.   :50.000             91     :  11   Max.   :25.0000
                            (Other): 738
>
```

Figure 2. Descriptive Statistics of Initial Variables in Bank Marketing Dataset.

```
> print(model)

         Conditional inference tree with 15 terminal nodes

Response:  suscribed.term.deposited
Inputs:  age, job, marital.status, education, default.credit, housing.loan, personal.loan, last.contact.type, last.contact.day
, last.contact.month, last.contact.duration.seconds, current.campaign.contacts, previous.campaigns, outcome.previous.campaign
Number of observations:  3181

1) last.contact.duration.seconds <= 645; criterion = 1, statistic = 501.117
  2) outcome.previous.campaign == {success}; criterion = 1, statistic = 386.06
    3) last.contact.duration.seconds <= 159; criterion = 0.997, statistic = 13.804
      4)*  weights = 17
    3) last.contact.duration.seconds > 159
      5)*  weights = 66
  2) outcome.previous.campaign == {failure, other, unknown}
    6) last.contact.month == {mar, oct}; criterion = 1, statistic = 219.111
      7) last.contact.duration.seconds <= 140; criterion = 0.975, statistic = 11.206
        8) marital.status == {married}; criterion = 0.996, statistic = 16.422
          9)*  weights = 20
        8) marital.status == {divorced, single}
          10)*  weights = 10
      7) last.contact.duration.seconds > 140
        11)*  weights = 47
    6) last.contact.month == {apr, aug, dec, feb, jan, jul, jun, may, nov, sep}
      12) last.contact.duration.seconds <= 222; criterion = 1, statistic = 170.686
        13) last.contact.type == {telephone, unknown}; criterion = 0.993, statistic = 29.447
          14) job == {student}; criterion = 1, statistic = 65.7
            15)*  weights = 10
          14) job == {admin., blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, technician,
unemployed, unknown}
            16)*  weights = 657
        13) last.contact.type == {cellular}
          17)*  weights = 1128
      12) last.contact.duration.seconds > 222
        18) last.contact.type == {cellular, telephone}; criterion = 1, statistic = 51.969
          19) last.contact.month == {jun, sep}; criterion = 1, statistic = 57.706
            20)*  weights = 34
          19) last.contact.month == {apr, aug, dec, feb, jan, jul, may, nov}
            21) last.contact.month == {jan, jul, may, nov}; criterion = 0.978, statistic = 23.228
              22)*  weights = 383
            21) last.contact.month == {apr, aug, dec, feb}
              23)*  weights = 255
        18) last.contact.type == {unknown}
          24) last.contact.month == {aug, jul, nov}; criterion = 1, statistic = 27.784
            25)*  weights = 7
          24) last.contact.month == {jun, may}
            26) last.contact.duration.seconds <= 366; criterion = 0.999, statistic = 15.58
              27)*  weights = 183
            26) last.contact.duration.seconds > 366
              28)*  weights = 110
1) last.contact.duration.seconds > 645
  29)*  weights = 254
>
```

Figure 3. Textual Format of Decision Tree on Bank Marketing Dataset.

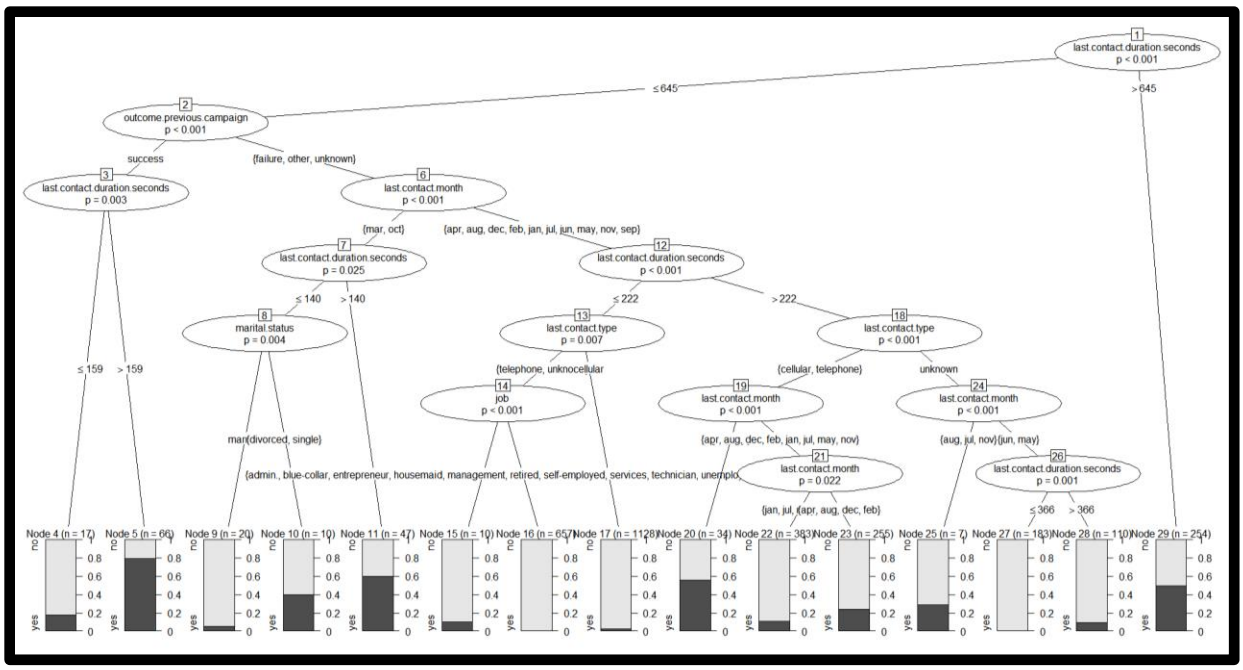

Figure 4. Graph of Decision Tree on Bank Marketing Dataset.

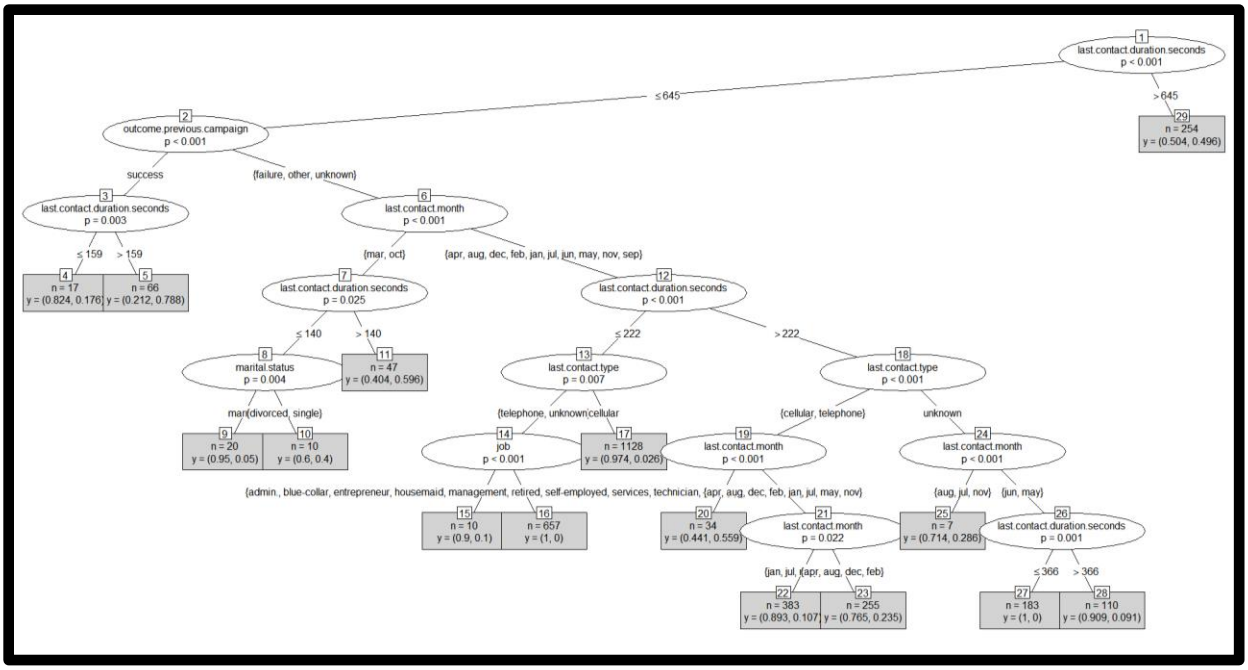

Figure 5. Simplified Graph of Decision Tree on Bank Marketing Dataset.

```
> table(predict(model), train.data$suscribed.term.deposited)

       no  yes
  no  2757  277
  yes   48   99
>
```

Figure 6. Confusion Matrix for Decision Tree on Training Set.

```
> testPred <- predict(model, newdata = test.data)
> table (testPred, test.data$suscribed.term.deposited)

testPred   no  yes
     no  1175  111
     yes   20   34
>
```

Figure 7. Confusion Matrix for Decision Tree on Test Set.

```
> print(model)

        Conditional inference tree with 93 terminal nodes

Response:  suscribed.term.deposited
Inputs:  age, job, marital.status, education, default.credit, housing.loan, personal.loan, last.contact.type, last.contact.day
, last.contact.month, last.contact.duration.seconds, current.campaign.contacts, previous.campaigns, outcome.previous.campaign
Number of observations:  5645

1) last.contact.duration.seconds <= 211; criterion = 1, statistic = 1281.165
  2) last.contact.month == {dec, feb, mar, oct, sep}; criterion = 1, statistic = 455.541
    3) last.contact.duration.seconds <= 79; criterion = 1, statistic = 35.412
      4)*  weights = 44
    3) last.contact.duration.seconds > 79
      5) last.contact.day <= 9; criterion = 1, statistic = 28.213
        6) outcome.previous.campaign == {success}; criterion = 1, statistic = 32.12
          7)*  weights = 17
        6) outcome.previous.campaign == {failure, other, unknown}
          8) last.contact.month == {dec, feb, oct, sep}; criterion = 1, statistic = 39.749
            9) education == {primary, secondary, unknown}; criterion = 0.97, statistic = 27.231
              10)*  weights = 37
            9) education == {tertiary}
              11)*  weights = 23
          8) last.contact.month == {mar}
            12)*  weights = 14
      5) last.contact.day > 9
        13) marital.status == {divorced}; criterion = 0.998, statistic = 17.967
          14)*  weights = 41
        13) marital.status == {married, single}
          15) last.contact.type == {unknown}; criterion = 0.981, statistic = 14.34
            16)*  weights = 14
          15) last.contact.type == {cellular, telephone}
            17) outcome.previous.campaign == {failure, other}; criterion = 0.975, statistic = 15.052
              18)*  weights = 18
            17) outcome.previous.campaign == {success, unknown}
              19)*  weights = 90
  2) last.contact.month == {apr, aug, jan, jul, jun, may, nov}
    20) outcome.previous.campaign == {failure, other, unknown}; criterion = 1, statistic = 442.253
```

Figure 8. Beginning of Textual Representation of Oversampled Decision Tree.

```
                        168) job == {admin., housemaid, retired, self-employed}
                          170)*  weights = 36
                151) personal.loan == {yes}
                  171) current.campaign.contacts <= 1; criterion = 0.96, statistic = 8.867
                    172)*  weights = 11
                  171) current.campaign.contacts > 1
                    173)*  weights = 18
              150) current.campaign.contacts > 6
                174)*  weights = 15
    73) last.contact.duration.seconds > 644
      175) last.contact.type == {telephone, unknown}; criterion = 0.996, statistic = 21.489
        176) last.contact.month == {apr, jan, may, nov}; criterion = 0.997, statistic = 31.127
          177) job == {blue-collar, entrepreneur, housemaid, management, self-employed, services, unemployed}; criterion = 0.997
, statistic = 31.835
            178)*  weights = 83
          177) job == {admin., retired, technician}
            179)*  weights = 102
        176) last.contact.month == {dec, feb, jul, jun, oct}
          180) job == {management, unemployed}; criterion = 0.963, statistic = 25.237
            181)*  weights = 8
          180) job == {admin., blue-collar, entrepreneur, housemaid, retired, self-employed, services, technician}
            182)*  weights = 153
      175) last.contact.type == {cellular}
        183) education == {unknown}; criterion = 0.982, statistic = 16.924
          184)*  weights = 15
        183) education == {primary, secondary, tertiary}
          185)*  weights = 760
> |
```

Figure 9. End of Textual Representation of Oversampled Decision Tree.

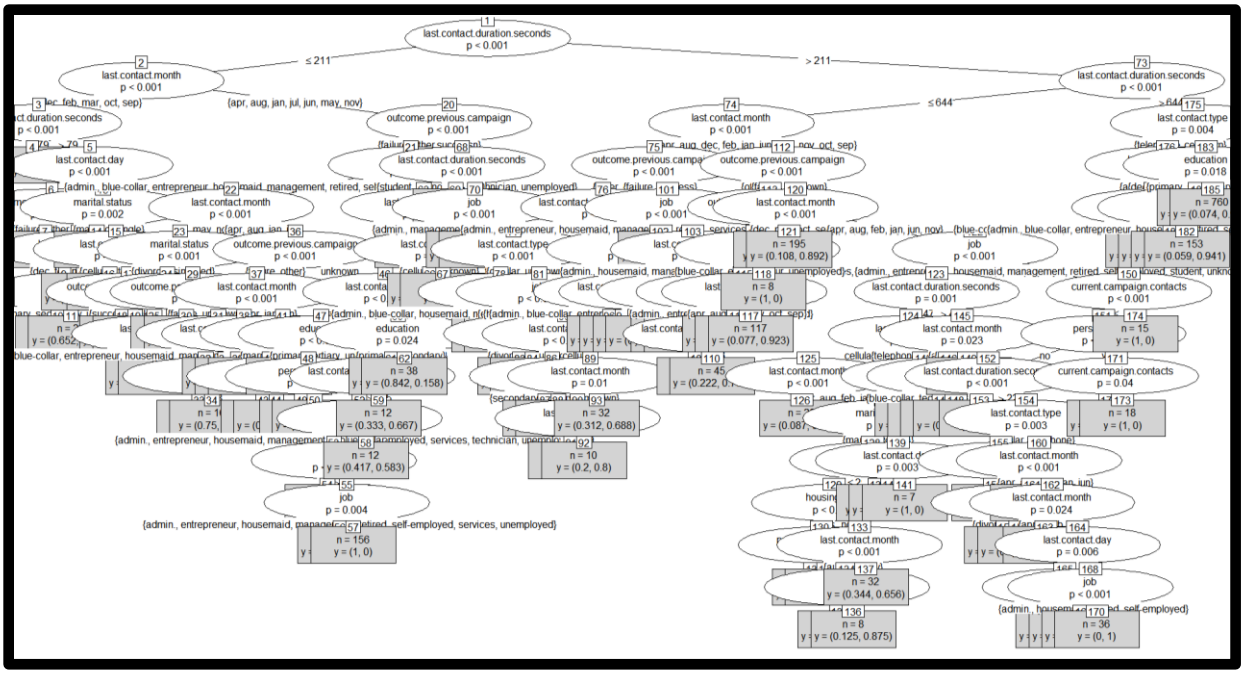

Figure 10. Simplified Graph of Decision Tree on Oversampled Dataset.

```
> table(predict(model), train.data$suscribed.term.deposited)

      0    1
  0 2386  150
  1  425 2684
>
```

Figure 11. Confusion Matrix for Oversampled Decision Tree on Training Set.

```
> testPred <- predict(model, newdata = test.data)
> table (testPred, test.data$suscribed.term.deposited)

testPred    0    1
       0  968   70
       1  221 1096
>
```

Figure 12. Confusion Matrix for Oversampled Decision Tree on Test Set.

```
> forest <- randomForest(suscribed.term.deposited ~ ., data = train.data,
+                        ntree = 25, importance = TRUE)
> forest

Call:
 randomForest(formula = suscribed.term.deposited ~ ., data = train.data,      ntree = 25, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 25
No. of variables tried at each split: 3

        OOB estimate of  error rate: 4.32%
Confusion matrix:
     0    1 class.error
0 2574  237 0.084311633
1    7 2827 0.002470007
>
```

Figure 13. Details of Random Forest and Confusion Matrix for Training Set.

```
> forest.pred <- predict(forest, newdata = test.data)
> table(forest.pred, test.data$suscribed.term.deposited)

forest.pred    0    1
          0 1088    0
          1  101 1166
>
```

Figure 14. Confusion Matrix for Random Forest on Test Set.