Assignment 1

Niall Trinder - R00088254

20 November 2018

Introduction

For this assignment, we're working on the premise that a manager from a local financial institution, Olivia, has contacted us to help her assess the credit worthiness of future potential customers.

She has promised to provide us with the following data:

- Data set 1; 793 past loan customers with 14 attributes, labelled based on Good/Bad Credit Standing.
- Data set 2; 10 potential loan customers with no labels.

Our goal is to explore and assess the past loan customer data set, chose an appropriate model to train using this data and then use that model to assess the potential customer data set to aid Olivia's financial institution minimize the risk of loaning money to a customer who may default on the loan and maximize the number of loans given out.

Exploratory Data Analysis (EDA)

Our goal of the EDA process is to learn about the data set which may help with the modelling process. This may include a strong correlation between a variable and the outcome label or finding missing or invalid entries in the data set. First we're going to look at the dimension of the data set, from the past customers set, we can see that there is 780 instances with 14 attributes. It's best if we look at the class variable, we can see from the chart below that the bar on the left in blue is the number of past customers who received a *"Bad"* credit standing label and the bar on the right corresponds to the number of past customers who received the *"Good"* label, they're 319(40.9%) and 461(59.1%) respectively.

Next we want to look at a histogram of how different values correlate to the class we want to predict. We won't look at all of these in detail however it must be noted that some of the values such as *Credit History*, *Checking Account*, *Savings Account* and *Employment* show to have a much clearer correlation to *Credit Standing* than say *Housing* or *Foreign National*. What we want to see is class purity, that the section of the chart is dominated by one class showing a clear correlation.

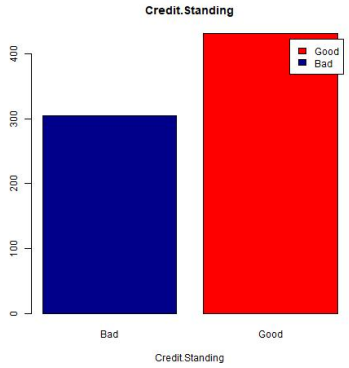
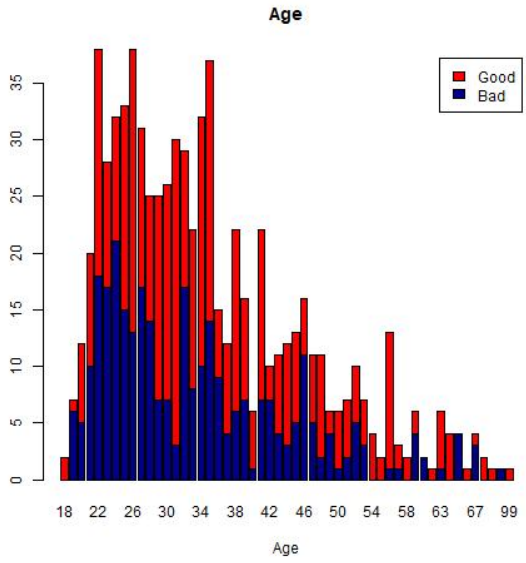
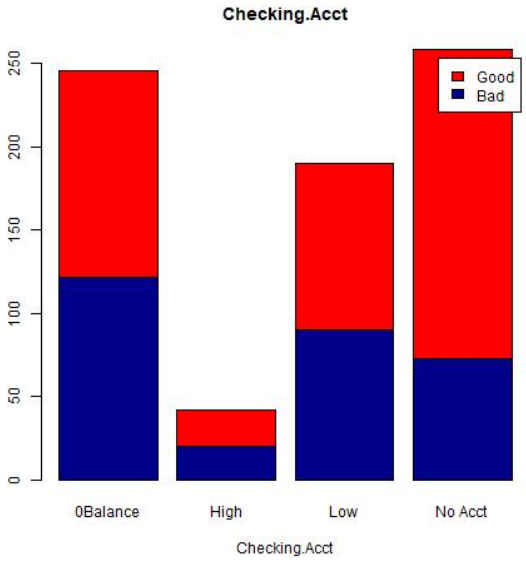
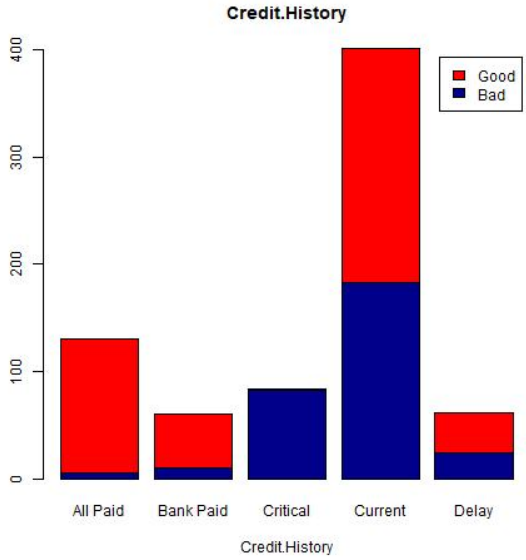


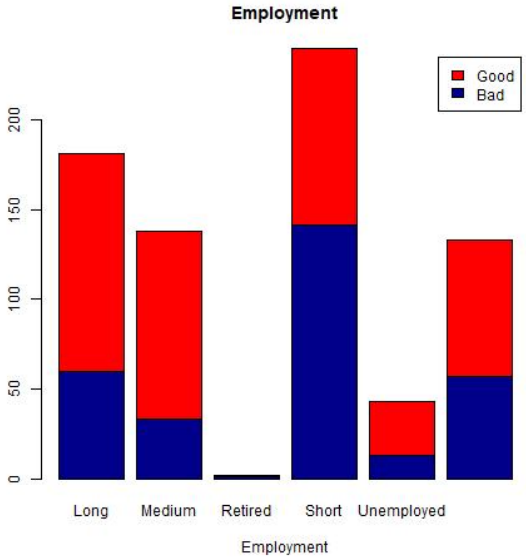
Figure 1:

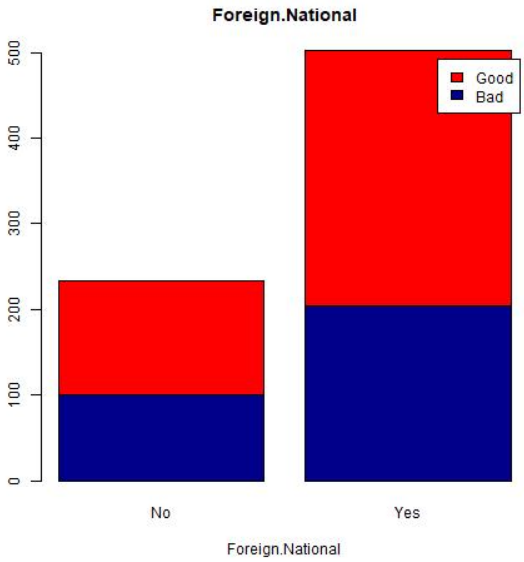
A Shapiro test was run on all the numerical variables and they all had a p-value of 2.2e-16 so we're going to accept that they follow normal distribution.

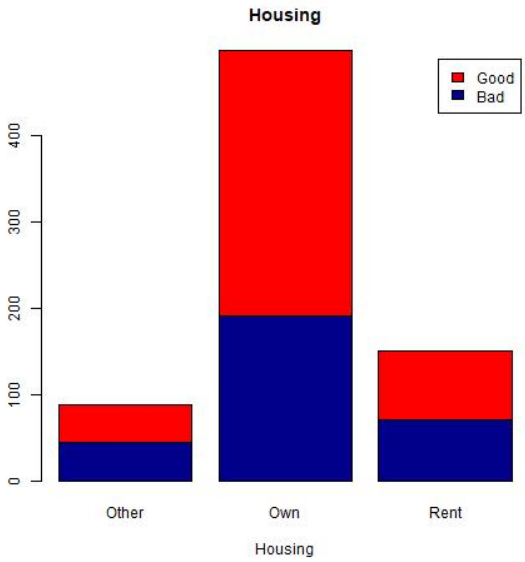


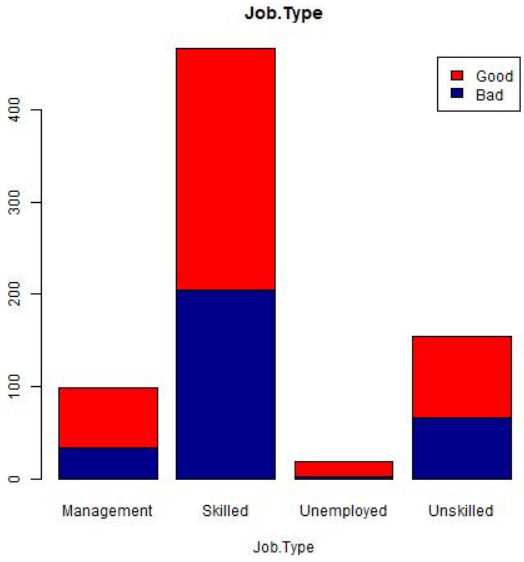


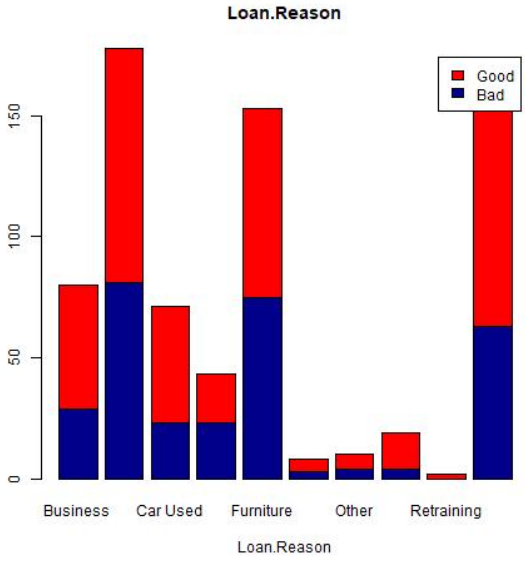


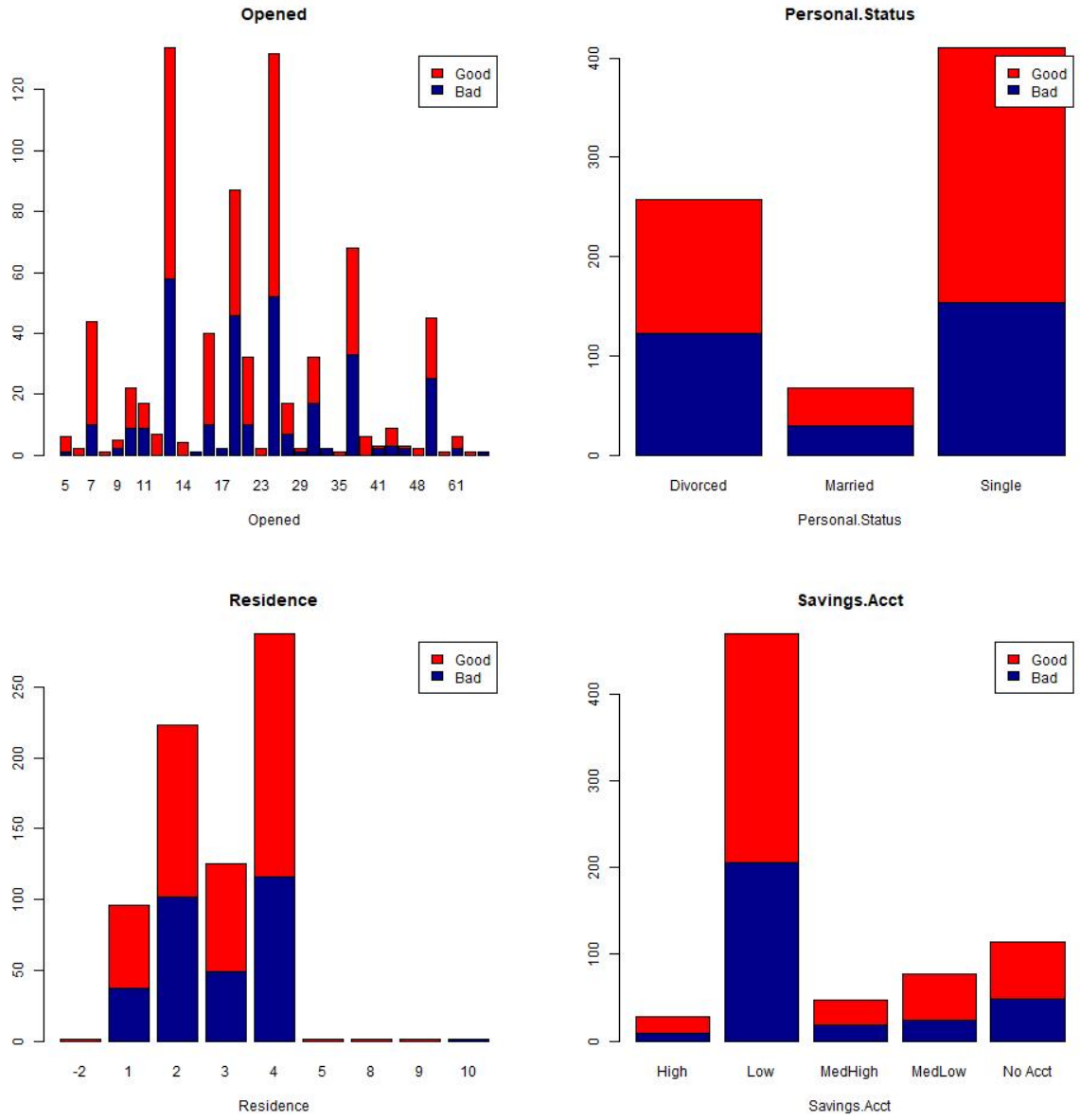



Foreign.National
Good
Bad
0
100
200
300
400
500
No
Yes
Foreign.National


Housing
Good
Bad
0
100
200
300
400
Other
Own
Rent
Housing


Job.Type
Good
Bad
0
100
200
300
400
Management
Skilled
Unemployed
Unskilled
Job.Type


Loan.Reason
Good
Bad
0
50
100
150
Business
Car Used
Furniture
Other
Retraining
Loan.Reason



Next we're going to look for NA or inconsistent data. First we're going to run the following lines of code to first remove all rows with NA and then check the dimensions of the data to see what we're left with:

```
past_customers <- na.omit(past_customers)
dim(past_customers)
```

The number of observations is reduced down to 737 meaning there were 43 rows with at least 1 NA value. This is necessary as the models used later won't be able to compile with NA values.

Running the summary function on past customers will let us explore the individual features closely and if there are inconsistencies hopefully we can find some here. As a result we can see that *Residence Time* has a min of -2, as time can't be negative this must be an error.

Now lets look at an example of trivariate analysis. Here we have Age on the x axis and Account Age or "Opened" on the y axis. These were chosen as they are both continous variables. The points are labeled using the associated Credit Standing values, blue for Good and red for Bad. It's difficult to see any possible relationships between the 3 variables, maybe the take home is that for this data set, there is no clear

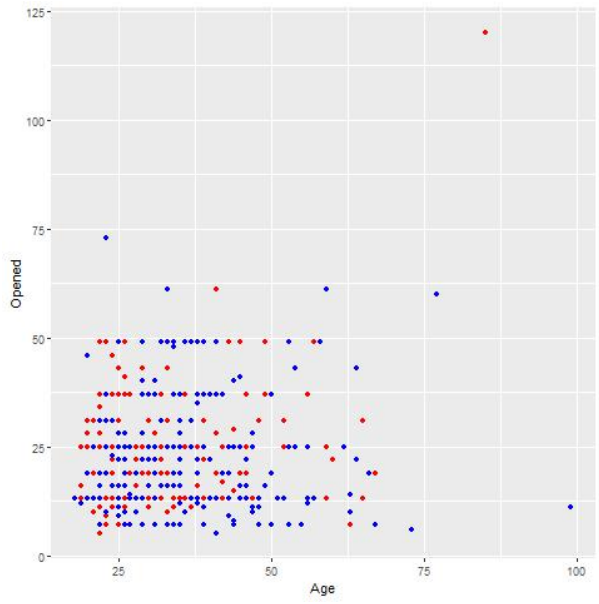


Figure 2: Age vs Account Age (Opened) vs Credit Standing

correlation between the 3.

Finally we're going to check numerical data for correlation. *Pearson's method* was used to find the correlation between *Age, Residence & Account Age. Residence* was shown to have a positive correlation with both *Age* and *Account Age* with p-values of **5.4e-15** and **6.9e-06** respectfully, there was no evidence of correlation between *Age* and *Account Age*. From this, it was chosen to drop *Residence* as a predictor variable from the future models. This is done with the aim of increasing accuracy by reducing the possibility of multi-colinearity.

ROC Curves

A ROC curve is a plot of the "true positive rate" on the y-axis and the "false positive rate" on the x-axis for every possible classification threshold.

The true positive rate (or sensitivity) is the number of true positive classifications divided by the total occurences of the true class. For instance; if you were to classify a patient has 'having the disease' or 'not having the disease' then the true positive rate would be the number of patients *correctly* classified as having the disease divided by the true total number of patients who have the disease.

The false positive rate (or specificity) is simply 1 - sensitivity, or, the number of false positive classifications divided by the total occurences of the false class.nThe ROC curve visualises all possible thresholds were as misclassification rate is the error rate for a single threshold. A diagonal line from (0, 0) to (1, 1) on the graph would represent a model that does no better than guessing. You can use the ROC curve to quantify the performance of the classifier by giving a higher rating to better performing models. To do this we evaluate the area under the curve and express it as a percentage of the total area.

Note that most problems in the real world don't have balanced classes and yet this does not affect the ROC curve. Also ROC curves are useful even if the predicted values are not properly calibrated.

Below is an example of a ROC with various results plotted. The red line would represent a model with good separation, the green line would represent fair separation and blue would be marginally better than guessing. The performance can be measured by the area under the curve with a value somewhere between 1 (perfect model) and 0.

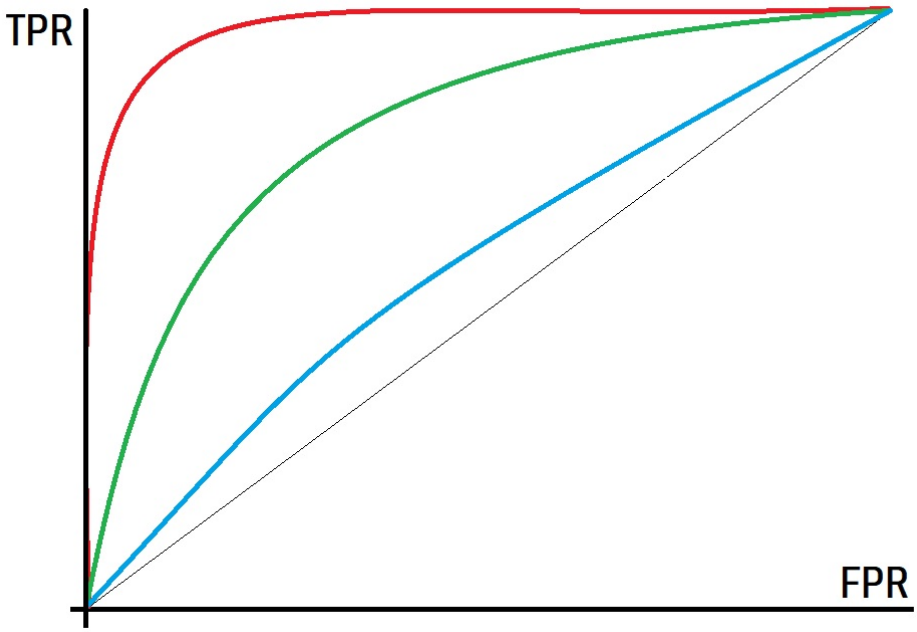


Figure 3: Example of ROC Curve

Decision Tree Model

Building the Model

This section will detail how the decision tree model was built and the choices behind it.

```
#using
library(tree)
# Build original tree model
tree_model <- tree(Credit.Standing~., past_train[,c(-1,-12)])
# Preform CV (10-fold) and plot result
tree_cv <- cv.tree(tree_model)
plot(tree_cv$size, tree_cv$dev, type='b')
# Make decision to prune tree to 4 nodes
tree_model <- prune.tree(tree_model, best=4)
# Make predictions of potential customers
tree_predictions <- predict(tree_model, potential_customers[,c(-1,-12)], type='class')
```

First the data from the past customers was split into test and train. A 70-30 split was found to yield the best outcome. The model was then built using the training data with the *"ID"* & and as mentioned in the EDA section, *"Residence"* columns removed. Following this, a 10-fold cross validation was performed in order to tune the number of nodes in the tree. The figure below generated by by the cv process shows how the number of nodes was chosen. Here we can see that the deviance is minimised at 4 nodes and as the graph takes on a U shape we can conclude that this is the optimum number.

From here, we can tune the tree model to 4 nodes, this will be the model we use to predict the credit standing of our potential customers.

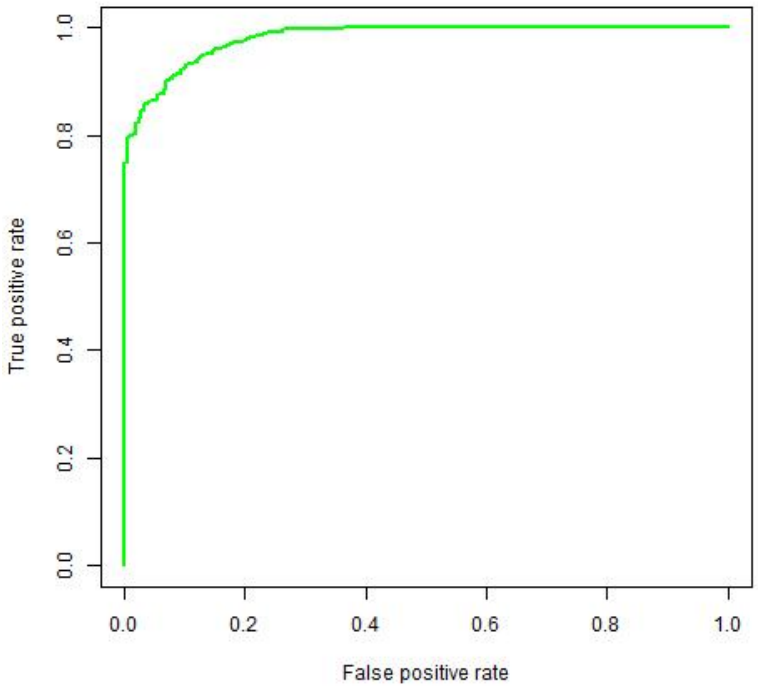


Figure 4: CV graph for Decision Tree

Interpreting the Model

In order to manually interpret the model, we can do use either one of two ways. First is to follow a the graphical representation of the decisions are model will take. After each decision the data will split and either be assigned or a new question will be asked, this will repeat until every customer has been assigned.

As you can see from the figure above, we start at the top with the question "Credit.History: Critical", this is called the root node and it is asking if the customer has a critical credit history, if the answer to that question is yes then you follow the line to the left, right if the answer is no. Following from that we can see that we have two outcomes, either we land a "Bad" node or we land at another question. If we landed at the next question, "Credit.History: Current, Delay" we repeat the same procedure as the first question, if this customer has a "Current" or "Delay" label for their credit history we follow the line to the left, otherwise we follow the line to the right. However, if we go back to where we ended up at the "Bad" node, we can see that there is further line of questioning, at this point we would label our customer as having a "Bad" Credit Standing.

As mentioned, this is only one way of interpreting the model, we can display all of this in a text output as well.

```
> tree_model
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 515 698.50 Good ( 0.41359 0.58641 )
   2) Credit.History: Critical 58   0.00 Bad ( 1.00000 0.00000 ) *
   3) Credit.History: All Paid,Bank Paid,Current,Delay 457 585.40 Good ( 0.33917 0.66083 )
     6) Credit.History: Current,Delay 328 449.30 Good ( 0.43598 0.56402 )
```

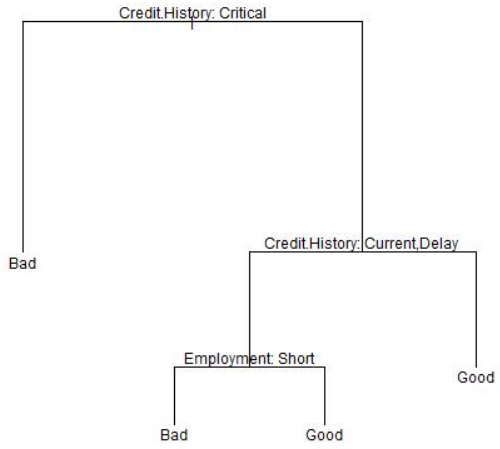


Figure 5: Tree Model Graphical Representation

```
      12) Employment: Short 109 142.20 Bad ( 0.64220 0.35780 ) *
      13) Employment: Long,Medium,Unemployed,Very Short 219 278.80 Good ( 0.33333 0.66667 ) *
     7) Credit.History: All Paid,Bank Paid 129  79.85 Good ( 0.09302 0.90698 ) *
```

One of the advantages of looking at the text output is that it displays the breakdowns and the probabilities at each step. As before, lets start on the root node. We can see that we start with 515 customers from the training data supplied, of these, 0.41% are labelled "Good" credit standing and 58.9% "Bad". Next we can see that we split the data based on credit history as before. When we broke the data into those with a Critical credit history and everyone else, we can see that everyone with the critical credit history had a bad credit standing. We calculate the probability at each node based on breakdown of the labels, here we have 58 customers with bad credit standing and 0 with good so we decide to give the customers who end at this node a 100% probablity of having a "Bad" credit standing. This logic is applied down through the tree.

Example of Model Outcome on Potential Customers

Lets look at an example of 3 potential customers and apply our tree model to predict their credit standing.

```
> potential_customers[1,]

     ID  Checking.Acct  Credit.History  Loan.Reason  Savings.Acct   Employment
    781        No Acct        All Paid      Car New       MedHigh        Short

    Personal.Status   Housing     Job.Type    Foreign.National
             Single      Rent    Unskilled                  No

Opened   Residence    Age   Credit.Standing
    11           2    39              <NA>
```

This is the data we have on potential customer ID 781, lets classify their credit status. Looking back up to the graphical representation, does this customer have critial credit history? No, lets go right. Does this customer have current or delay credit history? No, lets look at the text output, we can say that there is a 90.7% probabilty that this customer has "Good" credit standing. Lets double check that our model agrees.

```
> round(predict(tree_model, potential_customers[1,]),3)
      Bad  Good
781 0.093 0.907
```

And we can see that yes, our model agrees. Lets look at another.

```
> potential_customers[5,]
     ID Checking.Acct Credit.History     Loan.Reason Savings.Acct Employment Personal.Status Housing  Jo
785 785           Low        Current Small Appliance          Low     Medium        Divorced     Own Uns
    Foreign.National Opened Residence Age Credit.Standing
785              Yes      9         2  43            <NA>
```

This time we're looking at potential customer ID 785. Lets refer back to the graphical representation again. First question, does this customer have critical credit history? No, go right. Does this customer have current or delayed credit history? Yes, they have current credit history, go left. Does this customer have short employment status? No, go right. Back to text output, we can say that there is a 66.7% probabilty that this customer has "Good" credit standing. Let check the model.

```
> round(predict(tree_model, potential_customers[5,]),3)
      Bad  Good
785 0.333 0.667
```

Our model agrees, one more.

```
> potential_customers[13,]
     ID Checking.Acct Credit.History Loan.Reason Savings.Acct Employment Personal.Status Housing  Job.Ty
793 793           Low        Current     Car New          Low      Short          Single    Rent Unskil
    Opened Residence Age Credit.Standing
793     19         3  27            <NA>
```

Customer ID 793. Back to graphical representation, critical credit history? No, current, right. Current or delayed credit history? Yes, current, left. Employment short? Yes, back to text output and we can say there is a 64.2% probability that this customer has "Bad" credit standing.

```
> round(predict(tree_model, potential_customers[13,]), 3)
      Bad  Good
793 0.642 0.358
```

Se we can see how this approach is a clear and reliable way to classify our potential customers.

Other Approaches

Lets go back to the beginning and build 2 other models with different approaches to the same problem.

Random Forest

Our decision tree isn't going to be 100% accurate at predicting our customers due to varience. The way random forest addresses this is to effectively repeat the decision tree process a number of times with a random subset of the data set for each decision tree. In the end, seen as this is a discrete binary problem, the random forest is going to let each tree vote on the outcome and the highest vote will determine our prediction. Lets look how this was built.

```
# using
library(randomForest)
# build initial model
RF_model <- randomForest(Credit.Standing~.-ID-Residence, data = past_train, mtry = 6, importance = T, n
```

Here we use the randomForest library to build the mode. As you can see we are using the past_train data (minus the ID and Residence variables) to predict Credit.Standing. Our tuning parameters are mtry and and

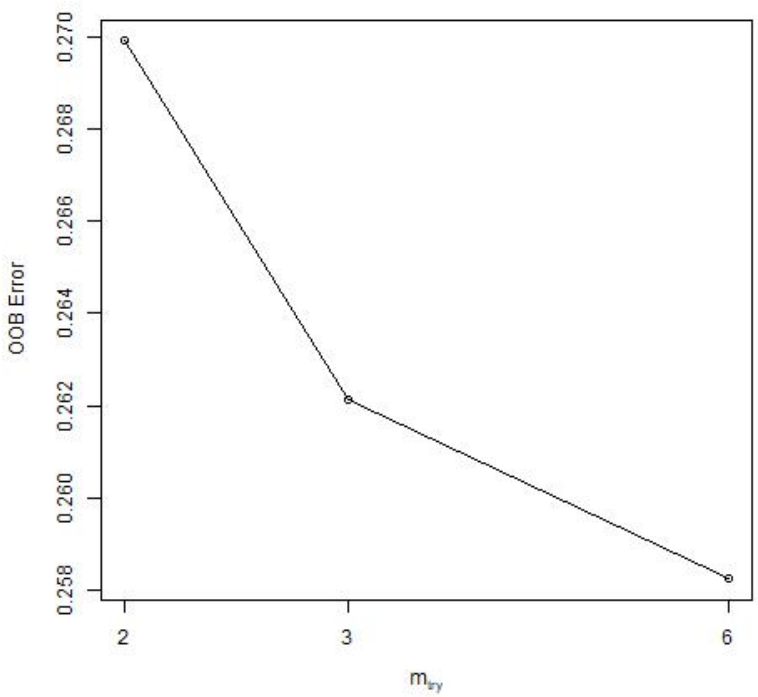


Figure 6: Tuning Mtry

ntree, ntree of 500 was found to deliver best results through trial and error while mtry was found using the following:

```
tuneRF(past_train[,-14], past_train[,14], stepfactor=1)
```

Now we can simply rebuild the model with our new tuning parameters and look at the confusion matrix.

```
> RF_model$confusion
     Bad Good class.error
Bad  140   73   0.3427230
Good  69  233   0.2284768
```

It must be noted that this confusion matrix is generated on the training set and as such we would expect a higher error on the testing data. Later we will be looking at how we can implement a method to reduce false negatives which will impact costs associated to the finacial insituation involved.

Gradient Boosted Model

A gradient boosted model or GBM is an an example of an ensemble technique. Unlike the random forest before which grew trees in parallel, the GBM algorithm grows trees in series which means we can learn from the previous trees. Lets look at how the GBM model was built.

```
library(gbm)
past_train$Credit.Standing <- ifelse(past_train$Credit.Standing=="Bad",0,1)
gbm_model <- gbm(Credit.Standing~.-ID-Residence, data = past_train, distribution='bernoulli',
                 shrinkage = 0.005, n.trees = 2000, interaction.depth = 1, cv.folds = 5)
```

First we're going to have to change our dependant variable to 0s and 1s, we are using bernouilli distribution

here to predict the Good or Bad credit standing. The following are all tuning parameters which we can change in order to increase the performance of the model, no fancy loops were used here, just simple trial and error to get a feel for the GBM process.

- Train/Test ratio
- Shrinage (rate which the model learns)
- n.trees
- Interaction Depth

Once that's complete we can take a look a some predictions. The following is generates the predictions for the test set, so all new data the model hasn't seen before. Note also that we have to define the number of trees and the type as response.

```
> gbm_predict <- predict(gbm_model, past_test[,-14], n.trees=2000, type='response')
> gbm_predict <- round(gbm_predict)
> gbm_predict <- ifelse(gbm_predict==0, "Bad", "Good")
> table(gbm_predict, past_test$Credit.Standing)

gbm_predict Bad Good
       Bad   52   17
       Good  40  113
```

Overall GBM is clearly quite powerful, there are plenty of parameters to tune. Going forward it would be recommended to keep the models in seperate R scripts, that way it would be more straight forward when trying to tuning such things as the training/test ratio. It's also clear how running simulations would be extremely taxing computationally.

Cost of Misclassification

```
> RF_sModel <- randomForest(Credit.Standing~.-ID-Residence, data = past_train, mtry = 6,
+                           importance = T, ntree=500, sampsize = c(50,250), strata=past_train$Credit.S
> tuneIT <- tuneRF(past_train[,-14], past_train[,14], stepfactor=1)

> RF_sModel$confusion
     Bad Good class.error
Bad   84  129  0.60563380
Good   7  295  0.02317881
```

```
> RF_sPredict <- predict(RF_sModel, past_test, type='class')
>
> table_sPredict <- table(RF_sPredict, past_test$Credit.Standing) %>%
+                   prop.table %>%
+                   round(3) %>%
+                   addmargins

> getAccuracy(table_sPredict)
[1] 0.6670644 # bad
[1] 0.8333333 # good
[1] 0.694     # overall
```

```
> RF_sModel$confusion
     Bad Good class.error
Bad   84  129  0.60563380
Good   7  295  0.02317881
```

Dealing with Inaccurate Classification

The credit standing classification process has been a mixture of a grading system along with human input. Here we are going to look into the claim that there was a peroid where this process was yielding inaccurate results. To do this, we're going to use the random forest model we've built to classify our past customers data set and compare the results to the label which was assigned to each from the past classification process. Once that is done, we can search for periods where both the random forest model and the past classification process consistantly do not agree with each other. After classifying the past customer data, we exported the ID and the labels for each customer then compared it to the *Credit Standing* label given to us from our original data set. Here simply using excel we can see that from roughly ID 305 to 325, the labels consistantly give contrasting labels. We're going to propose that this section of observations (from ID 305 to 325) have been either labelled incorrectly or the data gathering/input process was flawed and did not work as intended. If we found that the labels were wrongly assigned we could simply flip the labels, however seen as we cannot rule out the possibility that the data gathering/inupt was at fault the following is recommended:

First, discuss with Olivia and her associates to try discover where the error occured, was the data correctly inputted. Based on that discussion, either rebuild the model with the labels flipped or with the section of data removed. Analyse the new model, has it shown any improvement.

Here we're just going to assume that the data has been gathered/inputted incorrectly and as such we'll just remove it. Now lets rebuild the random forest model and look at the accuracy of the test set with & without the inconsistant data to see if there is an impact on performance.

List of Libraries Used

```
library(tree)
library(randomForest)
library(gbm)
library(ROCR)
library(dplyr)
library(magrittr)
library(ggplot2)
```

References

- http://gsp.humboldt.edu/OLM/R/05_04_CART.html
- https://www.r-bloggers.com/using-a-gbm-for-classification-in-r
- https://datascienceplus.com/gradient-boosting-in-r
- http://uc-r.github.io/gbm_regression
- https://rmarkdown.rstudio.com/authoring_basics.html
- https://stat.ethz.ch/R-manual/R-devel/library