Higher Diploma in Science in Data Analytics

Predicting personal loan charge offs using machine learning algorithms

Final Report

Noel Linnane

10389479@mydbs.ie

Student Number: 10389479

Supervisor: Dr. Shazia A. Afzal

Date: 13/12/2019

Word Count: 3,738

Abstract

The United States Federal Reserve states that 2.20% of personal loans charged off in Q3 2019. With total personal lending in the United States currently being estimated at $138 billion this represents a total charged off value of $3 billion.

The problem faced by all lenders is that some customers will be unable to repay their loans and as a result they will be charged off. When a loan is charged off it is most likely sent to a collection agency where the average recovery is only 20% of the loan value.

Therefore, it is important for lenders to be able to predict which customers are likely to charge off as early as possible.

The aim of this project is to assess the ability of several machine learning algorithms to predict charge offs using customer demographic and loan account data. It will assess the accuracy of these algorithms with the aim of concluding which if any is best suited to the task.

We will be using a dataset sourced from Kaggle which has been provided by Lending Club a peer to peer personal lender based in the United States.

Acknowledgements

I would like to take this opportunity to thank everyone who has supported me throughout this course in DBS. I am leaving this course with many new skills thanks to fantastic lecturers and mentors.

Contents

Figures

Introduction

A consumer loan is charged off if a customer has missed a specific number of payments, typically 4 consecutive payments. When a loan is charged off the lender considers the debt as unrecoverable, they may use a debt collection agency to recover the debt but on average only 20% of the value is recovered.

This is a problem for lenders as this debt may never be recovered. Therefore, it is important for lenders to be able to predict which customers will charge off. This is typically done at the application stage where lenders will rely on credit scores to assess credit worthiness.

The aim of this project is to assess the ability and accuracy of several machine learning algorithms to predict charge offs.

This project will use data sourced from Kaggle which has been provided by Lending Club. Lending Club is a peer to peer personal lender based in the United States.

The dataset contains 2,260,668 rows of data spanning from 2007 up to 2018. It contains both customer demographics and loan account data. In Figure 1 below we can see the number of loans issued by Lending Club.

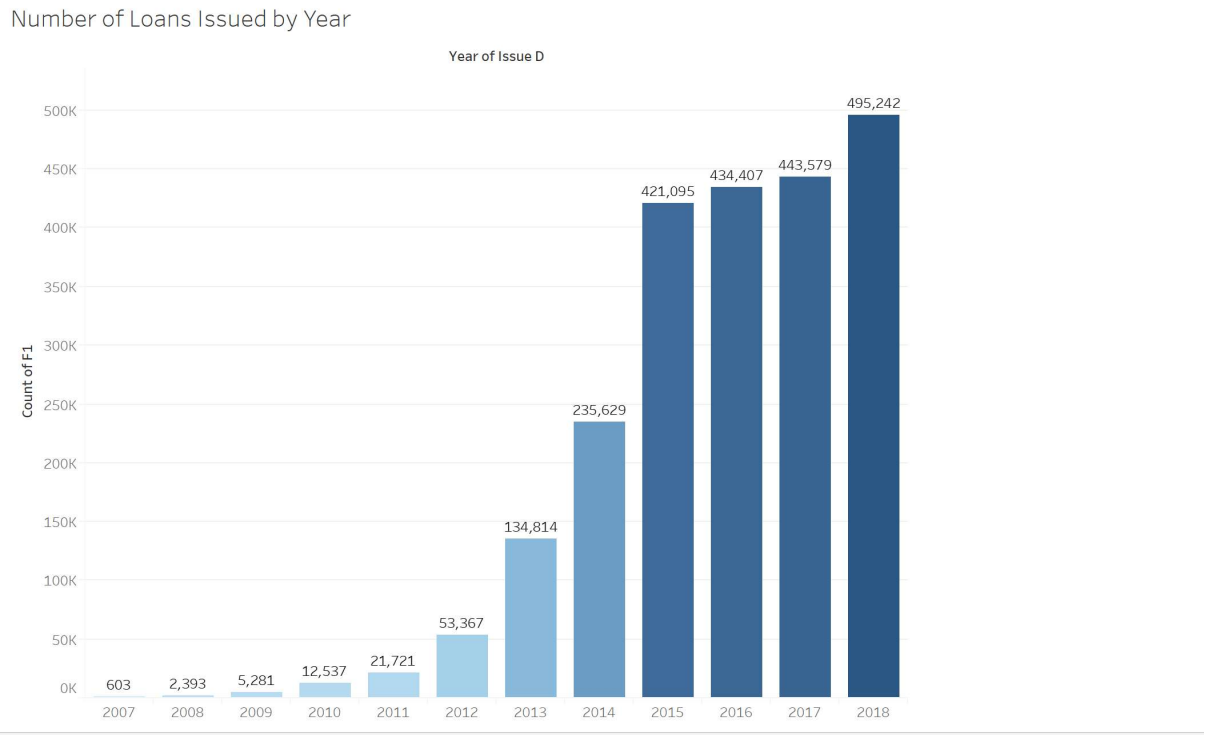

Figure 1 - Loans issued by Lending Club

In Figure 2 below we can see that Lending Club's charge off rate is significantly higher than the average quoted by the US Federal Reserve. In 2015 Lending Club's charge off rate was 17.88% which may speak to their motivation for publishing this dataset on Kaggle in the first place.

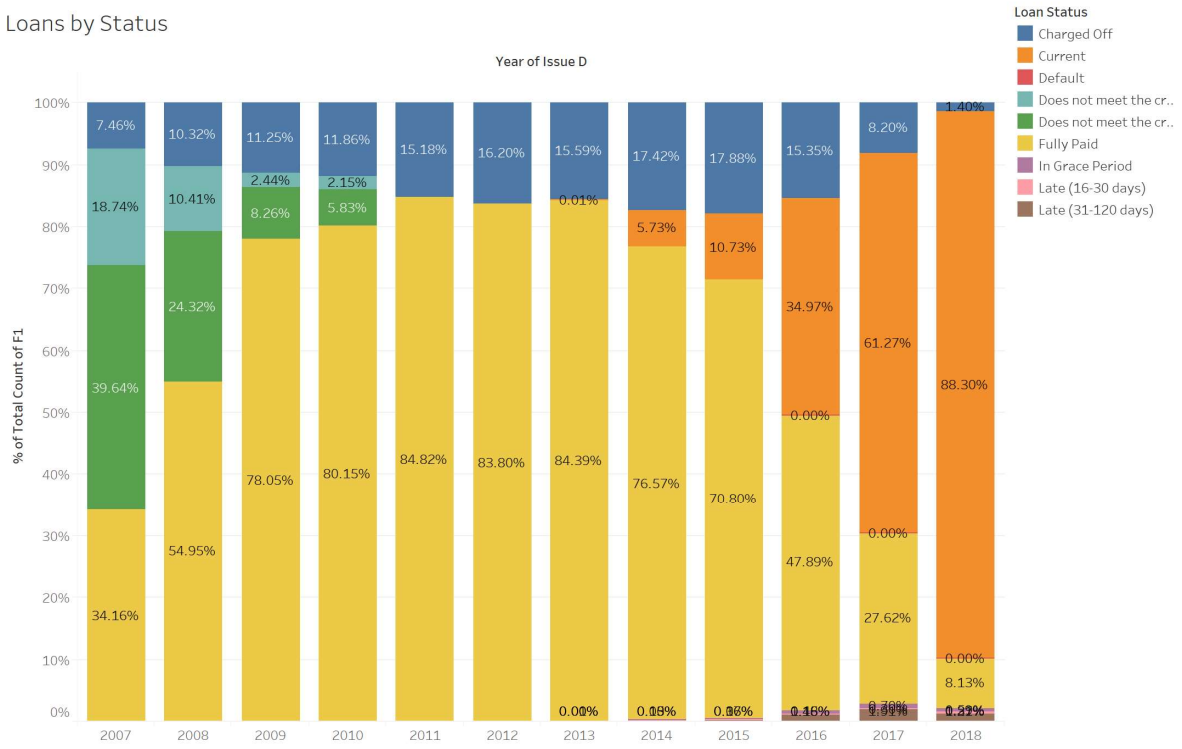

Figure 2 - Loans by Status

The question we are trying to answer in this paper is, is it possible to predict charge offs using machine learning and if so, how accurate are these models?

Background

Exploratory Data Analysis

The dataset is quite large, containing over 2 million rows of data with 145 features. Luckily there is a detailed data dictionary included with the dataset which gives detailed descriptions of the data features. The dataset was loaded into Tableau to perform some initial exploratory analysis of the data.

Address data is included in the dataset which allows us to map the number of loans issued by state. We can see that most loans are issued in California, Texas, Florida and New York (Figure 3).

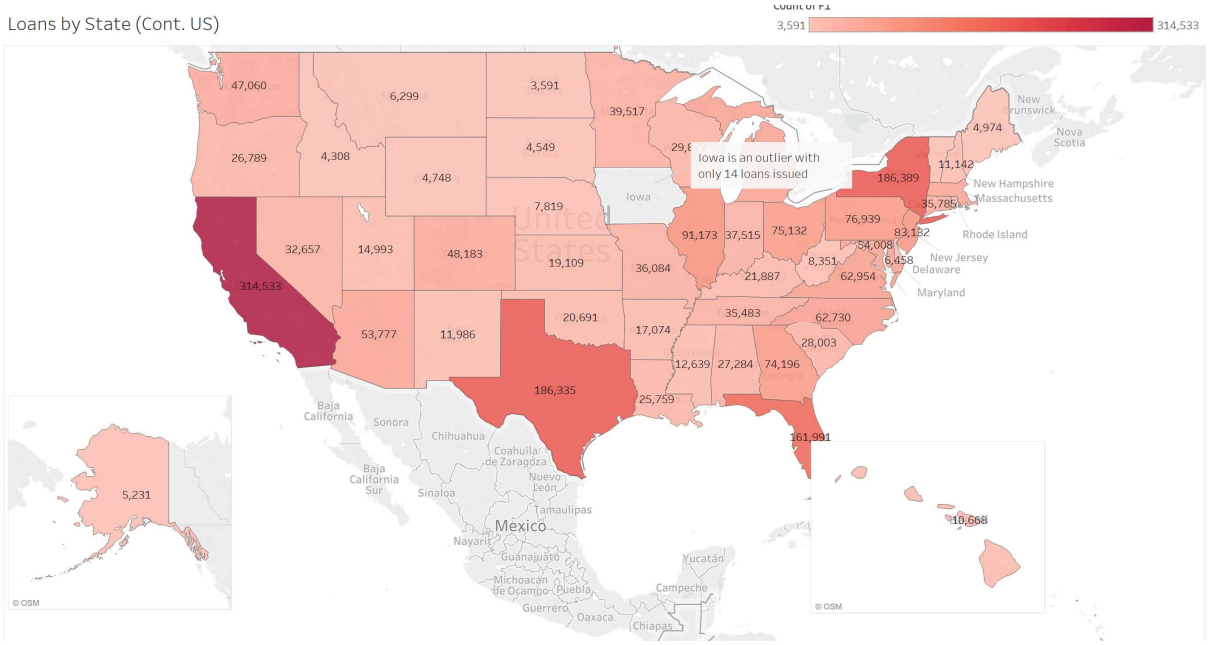

Figure 3 - Loans issued by state

Tableau allows the creation of new features by grouping, for example we can group the various loans statuses into just two statuses; “In Order” and “Default”; this allows us to see which states have the highest percentage of defaulted loans. We can see that the highest rate of default is in the following states (Figure 4):

- Alabama – 15.87%
- Mississippi – 15.62%
- Arkansas – 15.53%
- Louisiana – 15.47%
- Oklahoma – 15.29%

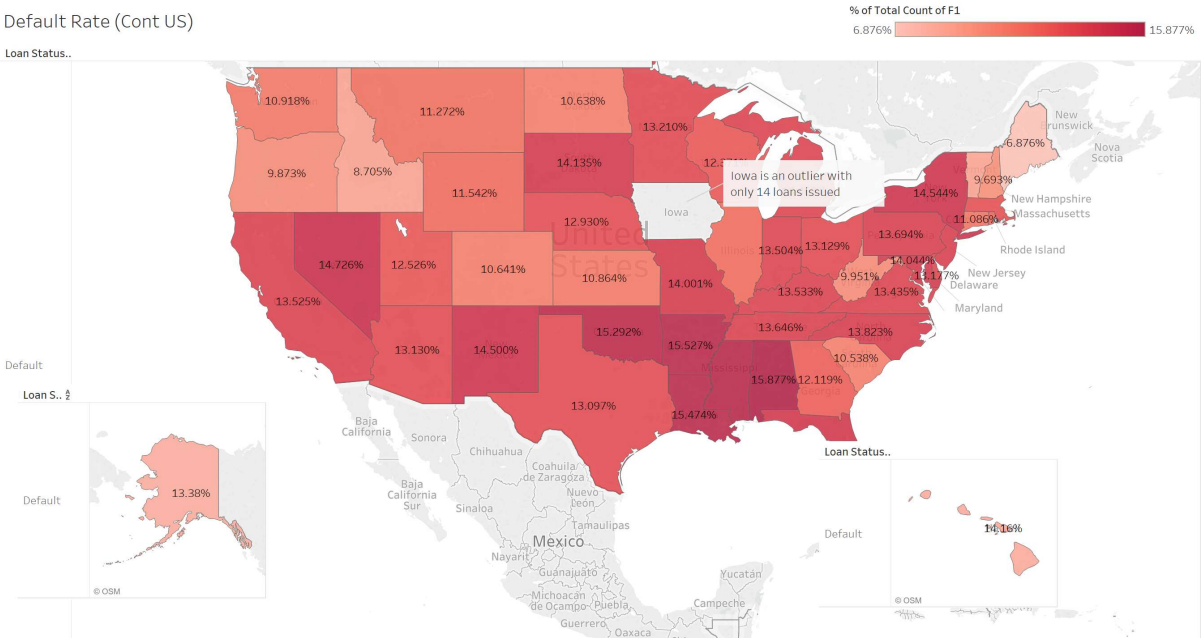

Figure 4 - Rate of default by state

The data also contains details about the purpose of the loan, this allows us to see if any loan type has a higher rate of default than others. In Figure 5 we can see that educational loans (student debt) have the highest rate of default.

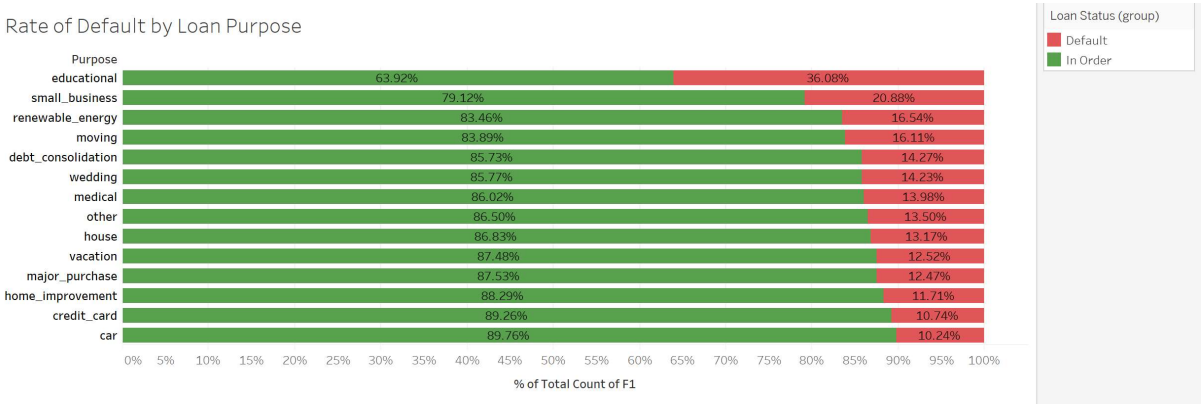

Figure 5 - Default rate by loan type

One might think then that educational loans should be treated as a higher risk and priced accordingly; however, we can see from Figure 6 that educational loans have the second lowest average interest rates.

Average Interest Rate by Loan Type

Purpose	
small_business	15.264
moving	14.739
renewable_energy	14.728
house	14.396
other	14.235
wedding	14.155
medical	13.632
debt_consolidation	13.523
vacation	13.451
major_purchase	12.764
home_improvement	12.621
car	12.182
educational	12.142
credit_card	11.696

Figure 6 - Average Interest Rates by Loan Purpose

Finally, we can see in Figure 7 that there is a greater instance of default with customers who rent. Customers who have put their home status as "None" and "Other" also have a significantly higher rate of default, but there are very few of these customers.

Defaults by Home Status

	Home Ownership					
Loan Status..	OTHER	NONE	RENT	OWN	MORTGAGE	ANY
Default	35.71%	22.22%	15.33%	13.23%	11.54%	7.53%
In Order	64.29%	77.78%	84.67%	86.77%	88.46%	92.47%

	Home Ownership					
Loan Status..	OTHER	NONE	RENT	OWN	MORTGAGE	ANY
Default	65	12	137,159	33,488	128,222	75
In Order	117	42	757,770	219,569	983,228	921

Figure 7 - Defaults by home status

This initial exploratory analysis of data might provide some valuable insights to Lending Club, for example default rates vary significantly by state, loan purpose and home ownership. It will be interesting to see if these features are significant in predicting charge offs later.

Requirements Specification and Design

For this project we will treat this as a classification problem.

Classification is a method of predicting a target variable given certain input variables. In this case the target variable will be “loan_status” and the input variables are the other 144 variables contained in the dataset, or rather a subset of these variables.

Classification is a very common form of supervised machine learning in which an algorithm learns from a “training” dataset containing all variables and then applies what it has learned to a new “test” dataset which withholds the target variable. We can then measure the effectiveness of the algorithm based on the accuracy of its predictions.

There are multiple classification algorithms available to use, this project will use the following.

Decision Tree

Decision tree is a simple classification method in which data is continuously split at “nodes” and branches out depending on the outcome of each split, it ends in a “terminal node” which contains our prediction.

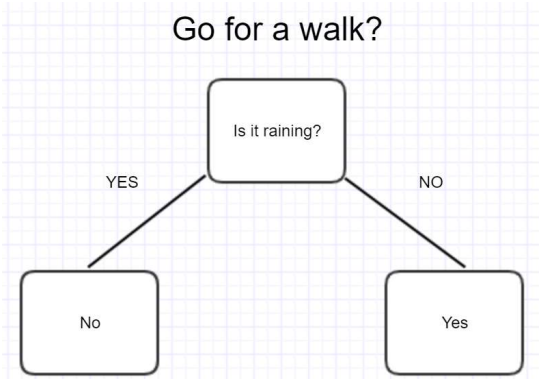

Figure 8 - Example decision tree

A very simple example can be seen in Figure 8 above. This is a decision tree asking whether we should go for a walk. A node will ask a question based on the input data like, “Is it raining?”, or, “Is the loan amount > $1,000?” and will branch out based on the answer to that question. A branch can have any number of nodes/questions before terminating in a decision.

Decision trees are easy to interpret, fast with large amounts of data and clearly highlight the important data features someone like Lending Club need to focus on. However, there are disadvantages as sometimes decision trees can get too big and too complicated, they can be bias towards more heavily weights variables and results can vary depending on how data is split between training and testing sets (Chakure, 2019).

Random Forest

Random Forest is what is known as an “ensemble algorithm”, this means that it combines multiple algorithms into one. It is regarded as one of the most powerful and accurate algorithms in machine learning (Brownlee, 2016).

Random Forest essentially runs multiple decision trees based on different subsets of the training dataset and aggregates the outcome from each subtree to decide the final output. This has the advantage of balancing out any potential bias that might be seen with Decision Tree. However, unlike Decision Tree we cannot see the tree structure, but we are able to extract which variables Random Forest has determined as most important.

GLM (Generalised linear model)

GLM is a statistical method that uses logistic regression to predict a binary outcome, i.e. 0 = Fully Paid, 1 = Charged Off, and will provide an outcome between 0 and 1. It is also an ensemble algorithm but unlike Random Forest it is easily interpreted. Where Decision Tree may sometimes fail to produce a tree, in cases where the variables are not very good indicators of the target variable, GLM will provide a result.

In order to accomplish this task, we will use the following tools:

- Tableau – which will be used for visualisation of the initial data exploration.
- RStudio – which will be used for data interrogation, manipulation and analysis. This is where we will build our model using several libraries such as, CARET, RPART, Random Forest, and ggplot2.
- Microsoft Excel 365

The program is outlined in Figure 9 below and will take the following steps:

- Importing data into RStudio using “read.csv”.
- Cleaning the data, primarily by removing NAs and reformatting data such as dates and numbers.
- Establish correlation between the variables and display them on a correlation heatmap.
- Feature selection, which is necessary to select important features only.
- Fitting the 3 algorithms.

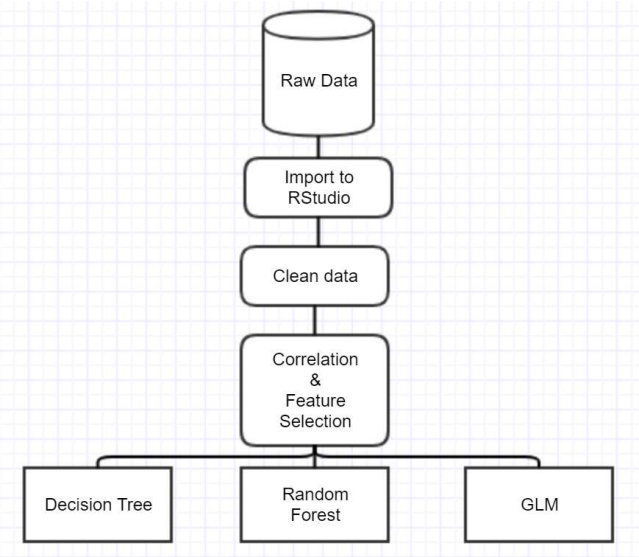

Figure 9 - Specification Design

Implementation

Import and cleaning

The first step is to load the data into RStudio, this is a straight-forward task using “read.csv” but as this is a large dataset it is going to take some time to read in the data. Once imported we can begin examining the data.

```
> str(data)
'data.frame':   2260668 obs. of  145 variables:
 $ id                                 : logi  NA NA NA NA NA NA ...
 $ member_id                          : logi  NA NA NA NA NA NA ...
 $ loan_amnt                          : int  2500 30000 5000 4000 30000 5550 2000 6000 5000 6000 ...
 $ funded_amnt                        : int  2500 30000 5000 4000 30000 5550 2000 6000 5000 6000 ...
 $ funded_amnt_inv                    : num  2500 30000 5000 4000 30000 5550 2000 6000 5000 6000 ...
 $ term                               : Factor w/ 2 levels " 36 months"," 60 months": 1 2 1 1 2 1 1 1 1 1
```

Figure 10 - Data structure

We can now see that the dataset is a data frame containing 2,260,668 rows of data with 145 variables (Figure 10). We can see a mix of data formats such as integers, numbers and factors. The first two variables, “id” and “member_id” are null. We can see there are many variables which are either all null or contain a very large number of nulls (Figure 11).

```
> colSums(is.na(data))
                                 id                           member_id                           loan_amnt
                            2260668                             2260668                                   0
                        funded_amnt                       funded_amnt_inv                                term
                                  0                                   0                                   0
                           int_rate                         installment                               grade
                                  0                                   0                                   0
                          sub_grade                           emp_title                          emp_length
                                  0                                  10                                   0
                    home_ownership                          annual_inc                   verification_status
                                  0                                   4                                   0
                            issue_d                         loan_status                          pymnt_plan
                                  0                                   0                                   0
                                url                                desc                             purpose
                            2260668                                   0                                   0
                              title                            zip_code                          addr_state
                                  1                                   0                                   0
                                dti                          delinq_2yrs                    earliest_cr_line
                               1711                                  29                                   0
                    inq_last_6mths                mths_since_last_delinq                mths_since_last_record
                                 30                             1158502                             1901512
                           open_acc                            pub_rec                           revol_bal
                                 29                                  29                                   0
```

Figure 11- Data NAs

We can remove columns which are all null or contain too many nulls. For this exercise a column with more than 2000 nulls will be removed. We can see this leaves us with 67 variables (Figure 12).

```
> data2 <- data[,-which(colSums(is.na(data))>2000)]
> names(data2)
 [1] "loan_amnt"                 "funded_amnt"               "funded_amnt_inv"           "term"
 [5] "int_rate"                  "installment"               "grade"                     "sub_grade"
 [9] "emp_title"                 "emp_length"                "home_ownership"            "annual_inc"
[13] "verification_status"       "issue_d"                   "loan_status"               "pymnt_plan"
[17] "desc"                      "purpose"                   "title"                     "zip_code"
[21] "addr_state"                "dti"                       "delinq_2yrs"               "earliest_cr_line"
[25] "inq_last_6mths"            "open_acc"                  "pub_rec"                   "revol_bal"
[29] "revol_util"                "total_acc"                 "initial_list_status"       "out_prncp"
[33] "out_prncp_inv"             "total_pymnt"               "total_pymnt_inv"           "total_rec_prncp"
[37] "total_rec_int"             "total_rec_late_fee"        "recoveries"                "collection_recovery_fee"
[41] "last_pymnt_d"              "last_pymnt_amnt"           "next_pymnt_d"              "last_credit_pull_d"
[45] "collections_12_mths_ex_med" "policy_code"              "application_type"          "verification_status_joint"
[49] "acc_now_delinq"            "chargeoff_within_12_mths"  "delinq_amnt"              "pub_rec_bankruptcies"
[53] "tax_liens"                 "sec_app_earliest_cr_line"  "hardship_flag"             "hardship_type"
[57] "hardship_reason"           "hardship_status"           "hardship_start_date"       "hardship_end_date"
[61] "payment_plan_start_date"   "hardship_loan_status"      "disbursement_method"       "debt_settlement_flag"
[65] "debt_settlement_flag_date" "settlement_status"         "settlement_date"
```

Figure 12 - Data remove null columns

We now need to reformat some of the data into correct or more useful formats, for example the correlation function requires numbers and the GLM algorithm requires the target variable to be binary.

We can use the Lubridate library to reformat dates which are currently formatted as text, the Tidyverse library to create a new variable called "class" which we will use as our target variable and the inbuilt libraries to reformat "term", "grade" and other variables in to factors that can be used by the algorithms later (Figure 13).

```
library(lubridate)
library(tidyverse)

data2$loan_amnt <- as.numeric(data2$loan_amnt)
data2$term <- as.numeric(as.factor(data2$term))
data2$grade <- as.numeric(as.factor(data2$grade))
data2$issue_d <- parse_date_time(data2$issue_d, orders = "my")
data2 <- mutate(data2, class=as.integer(as.factor(ifelse(data2$loan_status == "Fully Paid",1,2))))
```

Figure 13 - Reformat data types

Now the data has been reformatted, we are able to subset the dataset based on loan issue date and loan status in order to make it a more manageable size. We subset based on loans issued between 01/01/2018 and 31/12/2018 with a loan status of "Fully Paid" or "Charged Off" only.

```
data2 <- subset(data2, loan_status == "Fully Paid" | loan_status == "Charged Off")
data2 <- subset(data2, issue_d >= "2018-01-01" & issue_d <= "2018-12-31" )
dim(data2)
```

Figure 14 - Subsetting data

This clean up exercises leaves us with a more manageable 47,007 rows of data and 68 variables.

Correlation

Correlation allows us to see how the variables are related to each other, and in particular, how they are related to the target variable. This analysis will help us to select which features we will use going forward.

The correlation heatmap (Figure 15) is created using the GGally library it shows strong positive correlations as dark red, strong negative correlations as dark blue and variables with no correlation as white.

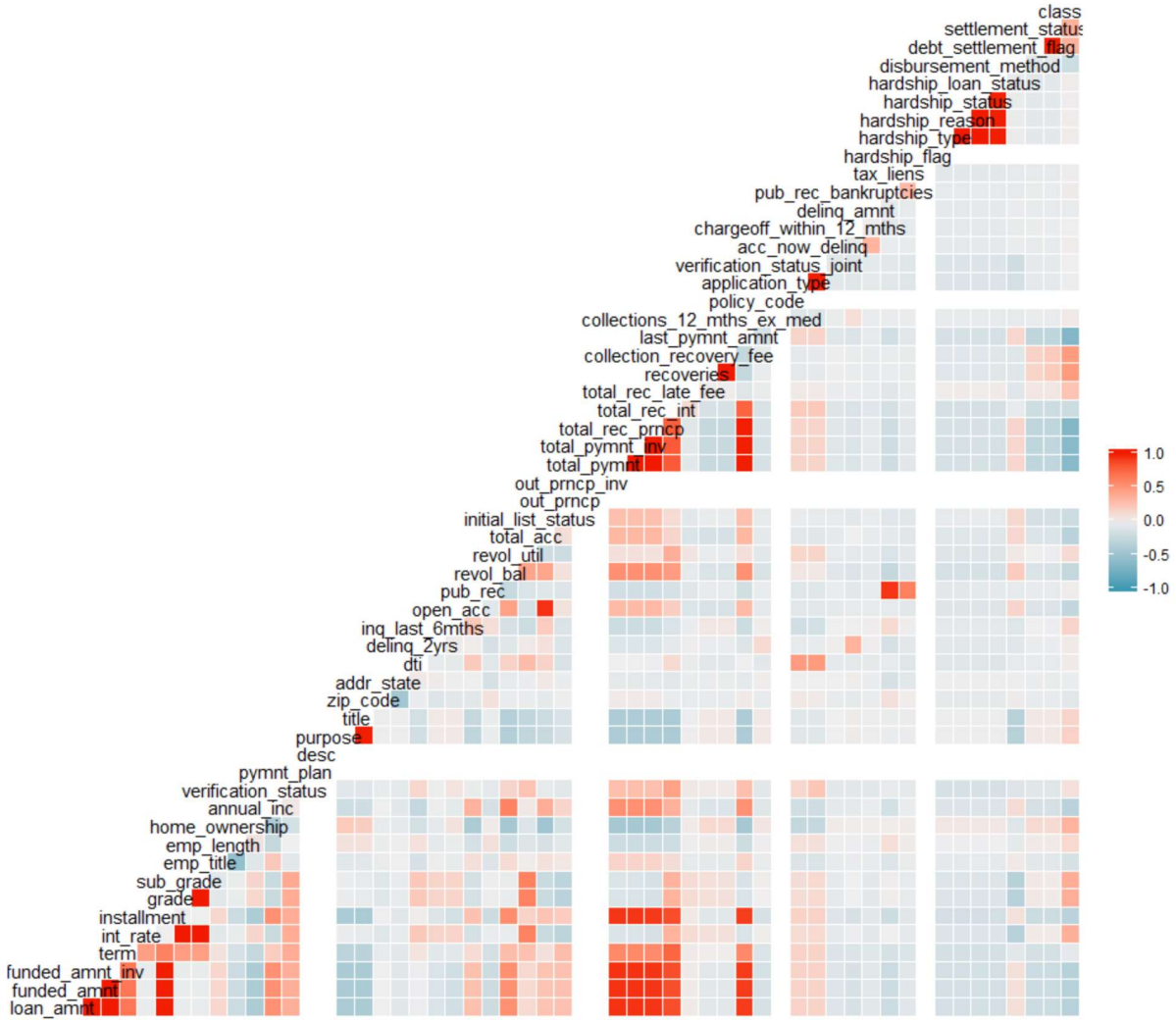

Figure 15 - Correlation Heat Map

We can see that “loan_amnt” is strongly correlated with “funded_amnt” and “funded_amnt_inv”. A quick look at the data dictionary supplied with the dataset would suggest these are the same values and therefore don’t add any additional data. We can remove “funded_amnt” and “funded_amnt_inv” from the dataset going forward.

We can see there are a number of variables showing no correlations at all, like “pymnt_plan” and “desc”. We can remove these variables also as they are not adding any information.

In relation to variables which are correlated with the target variable, “class”, we can see the following are positively correlated; “collection_recovery_fee”, “recoveries”, “grade” and “int_rate”. The following are negatively correlated, “last_pymnt_amnt”, “total_rec_prncp”, “total_pymnt”. However, none of these are strongly correlated.

Feature Selection

There are now 68 variables in our dataset which should be reviewed to see if any can be removed in order to increase the efficiency and accuracy of our models later. We have already seen in our correlation heatmap that some of these features add no value.

In addition to the correlation heatmap we can consult the data dictionary which has been provided by Lending Club. There is also an element of intuition involved in this process.

Having reviewed the correlation heat map and the data dictionary I have removed features which do not appear to add information and narrowed it down the following 29 features (Figure 16):

LoanStatNew	Description
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
addr_state	The state provided by the borrower in the loan application
annual_inc	The self-reported annual income provided by the borrower during registration.
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt
grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
initial_list_status	The initial listing status of the loan. Possible values are – W, F
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
issue_d	The month which the loan was funded
last_pymnt_amnt	Last total payment amount received
loan_amnt	credit department reduces the loan amount, then it will be reflected in this value.
loan_status	Current status of the loan
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records
pub_rec_bankruptcies	Number of public record bankruptcies
purpose	A category provided by the borrower for the loan request.
pymnt_plan	Indicates if a payment plan has been put in place for the loan
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
sub_grade	LC assigned loan subgrade
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
total_acc	The total number of credit lines currently in the borrower's credit file
total_pymnt	Payments received to date for total amount funded
total_rec_prncp	Principal received to date
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
hardship_flag	Flags whether or not the borrower is on a hardship plan

Figure 16 – List of features selected for modelling

Modelling – Decision Tree

The dataset is first split into a training set and test set using an 80/20 split.

We then use the rpart library to fit the decision tree model (Figure 17) and the rpart.plot library to plot the resulting decision tree which can be seen in Figure 18.

```
151  # Decision Tree
152
153  dt <- rpart(class ~ ., data = trainset, method = "class")
154  predictedvalues = predict(dt, testset, type = 'class')
155  tab_dt = table(predictedvalues, actualvalues=testset[,29]) #confusion matrix
156  accuracy_dt=sum(tab_dt[row(tab_dt)==col(tab_dt)])/sum(tab_dt)
157  accuracy_dt
158  rpart.plot(dt, cex=0.75) # Visualise the Decison Tree
159  tab_dt #Confusion matrix
160  confusionMatrix(predictedvalues, as.factor(testset$class))
161  roc.curve(data2$class, predict(dt, data2, type = "prob")[,1],plot=TRUE)
```

Figure 17 - Decision Tree Model

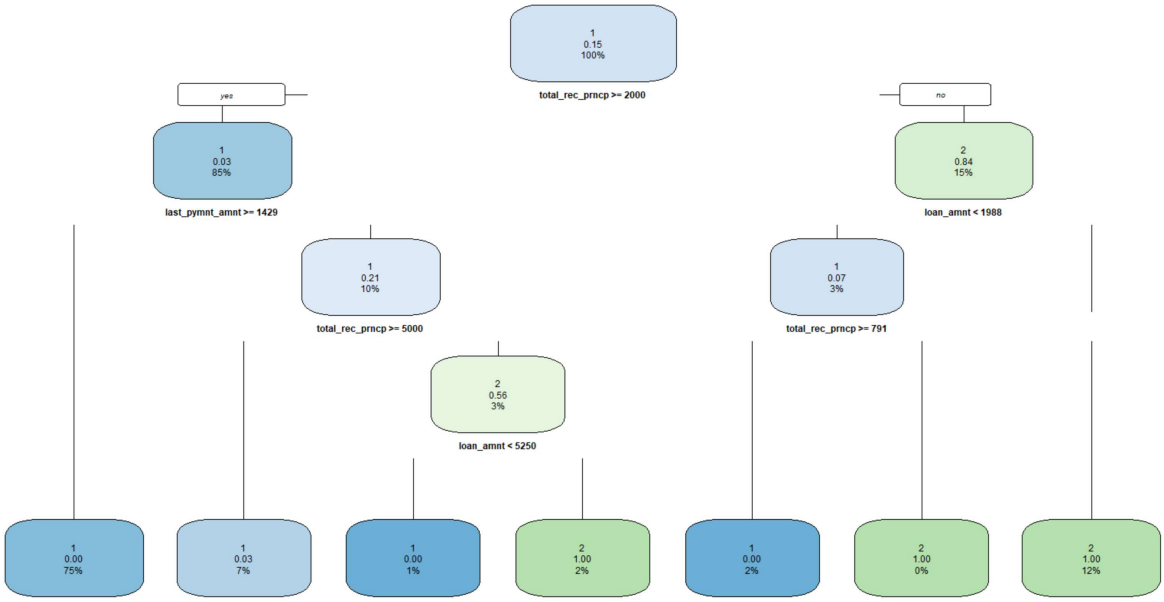

Figure 18 - Decision Tree Plot

We can see from the decision tree that the model has chosen "total_rec_prncp", "loan_amnt" and "last_pmnt_amt" as the most important variables in classifying the loan status as charged off or fully paid. (Remember we set Fully Paid = 1 and Charged off = 2 earlier).

The root node asks if the total recovered principle is greater than or equal to $2,000, it splits out based on a "yes" or "no" answer. If the answer is "yes" it then asks if the last payment amount was greater than or equal to $1,429 if the answer is again "yes" the node terminates and provides its prediction; that the account is paid in full.

The accuracy measure for this model is 99.8%, which is very high. We will examine this and other ways to measure the performance of the model in the next chapter.

Modelling – Random Forest

We then move on to the Random Forest model using the RandomForest library (Figure 19).

```
# Random Forest

library(randomForest)
library(ggplot2)

rf=randomForest(as.factor(class) ~ ., type='class', data = trainset, ntree=10) #as.factor is necessary for it to do classification
predictedvalues = predict(rf, testset, type='class')
tab_rf = table(predictedvalues, actualvalues=testset[,29]) #confusion matrix
accuracy_rf= sum(tab_rf[row(tab_rf)==col(tab_rf)])/sum(tab_rf)
accuracy_rf
tab_rf
roc.curve(data2$class, predict(rf, data2, type = "prob")[,1],plot=TRUE)

# Visualise Variable Importance
# Ggplot requires a dataframe, so convery VarImp in to a dataframe

vi <- data.frame(Variable = c(names(data2[,-29])), #removed the class variable as thats not in VarImp
                 Overall = c(varImp(rf)[1]))

v <- ggplot(data=vi,aes(x = reorder(Variable, Overall),y=Overall))+
  geom_bar(stat="identity", fill="steelblue") +
  theme_minimal() +
  #coord_cartesian(ylim = c(90,100)) +
  labs(x = "Variable")
v + coord_flip()
```

Figure 19 – Random Forest Model

Unfortunately, we don't see the trees output by the Random Forest model, but we are able to plot what the model considers to be the most important variables (Figure 20).

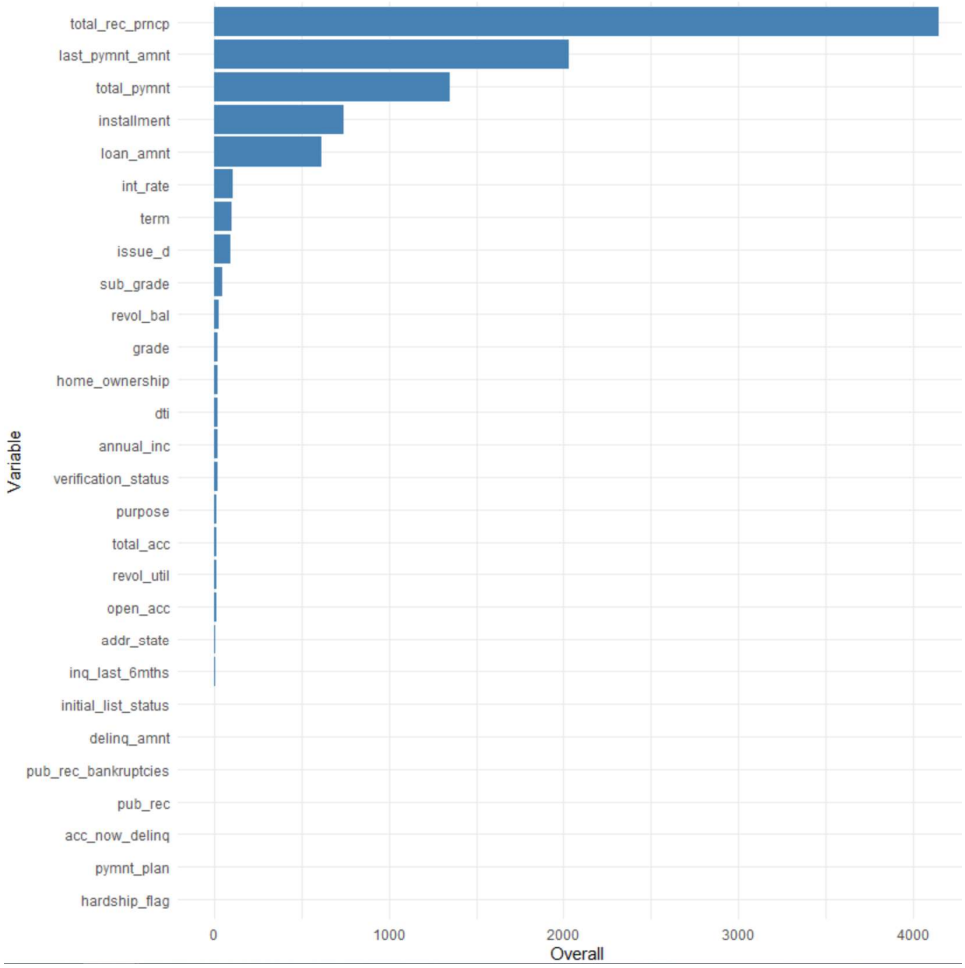

Figure 20 - Variable Importance (RF)

As we can see the model has determined that “total_rec_prncp”, “last_pymnt_amnt”, “total_pymnt”, “installment” and “loan_amnt” are the most important variables for classifying the loan status.

The accuracy for this model is 99.9%.

Modelling – GLM

Finally, we implement the GLM model in RStudio (Figure 21).

```
# GLM

trainset.glm <- glm(class-1 ~.,trainset, family="binomial") # class-1 makes the variable values 0 and 1
phati=predict(trainset.glm,testset, type="response")  # p hat i
predictedvalues=rep(0,length(phati))
predictedvalues[phati>0.5]=1   # probability of Gender being 1, if p<0.5 then gender=0
actual=testset[,29]-1
tab_glm=table( predictedvalues, actualvalues=testset[,29]-1) #
accuracy_gl=mean(predictedvalues == testset[,29]-1) ## was able to add headings so should do that for the others too...
accuracy_gl #higher better
tab_glm #Confusion matrix

```

Figure 21 - GLM Model

Looking at the output for the model (Figure 22), we can see that the following variables are significant; “loan_amnt”, “total_rec_prncp”, “acc_now_delinq”.

```
Coefficients: (2 not defined because of singularities)
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -2.290e+02  2.138e+02  -1.071   0.2842
loan_amnt            2.155e+02  4.823e+01   4.467 7.92e-06 ***
term                -3.727e+00  1.035e+01  -0.360   0.7186
int_rate            -7.268e-01  1.228e+00  -0.592   0.5538
installment         -2.766e-03  3.149e-02  -0.088   0.9300
grade                4.503e+00  3.502e+00   1.286   0.1985
sub_grade           -3.881e-01  1.034e+00  -0.375   0.7073
home_ownership       5.227e-01  4.936e-01   1.059   0.2896
annual_inc          -1.593e-05  2.446e-05  -0.651   0.5148
verification_status -1.631e+00  1.324e+00  -1.232   0.2181
issue_d              1.362e-07  1.108e-07   1.229   0.2192
pymnt_plan                  NA         NA      NA       NA
purpose              4.927e-01  2.274e-01   2.166   0.0303 *
addr_state          -2.731e-02  5.250e-02  -0.520   0.6030
dti                 -3.414e-02  8.954e-02  -0.381   0.7030
inq_last_6mths      -1.291e+00  1.392e+00  -0.928   0.3536
open_acc             3.631e-01  2.732e-01   1.329   0.1838
pub_rec             -9.390e+00  1.880e+03  -0.005   0.9960
revol_bal            2.191e-05  9.885e-06   2.217   0.0266 *
revol_util          -2.906e-02  4.467e-02  -0.650   0.5154
total_acc           -2.575e-01  1.977e-01  -1.302   0.1928
initial_list_status  8.887e+00  6.629e+01   0.134   0.8933
total_pymnt          9.278e-04  3.491e-03   0.266   0.7904
total_rec_prncp     -2.155e+02  4.823e+01  -4.467 7.92e-06 ***
last_pymnt_amnt      1.359e-03  1.583e-03   0.858   0.3906
acc_now_delinq      -1.672e+06  3.743e+05  -4.467 7.93e-06 ***
delinq_amnt         -1.449e-02  6.226e-01  -0.023   0.9814
pub_rec_bankruptcies 1.540e+00  1.881e+03   0.001   0.9993
hardship_flag               NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 22 - GLM Output

The accuracy measure for the GLM model is 100%. We will now look more closely at measuring the performance of these models.

Testing and Evaluation

At the beginning we asked the question, “is it possible to predict charge offs using machine learning and if so, how accurate are these models?”. I think we can now conclude that it is possible to predict charge offs using machine learning and it would also appear that these models are very accurate.

See Figure 23 below which plots the measured accuracy for each model.

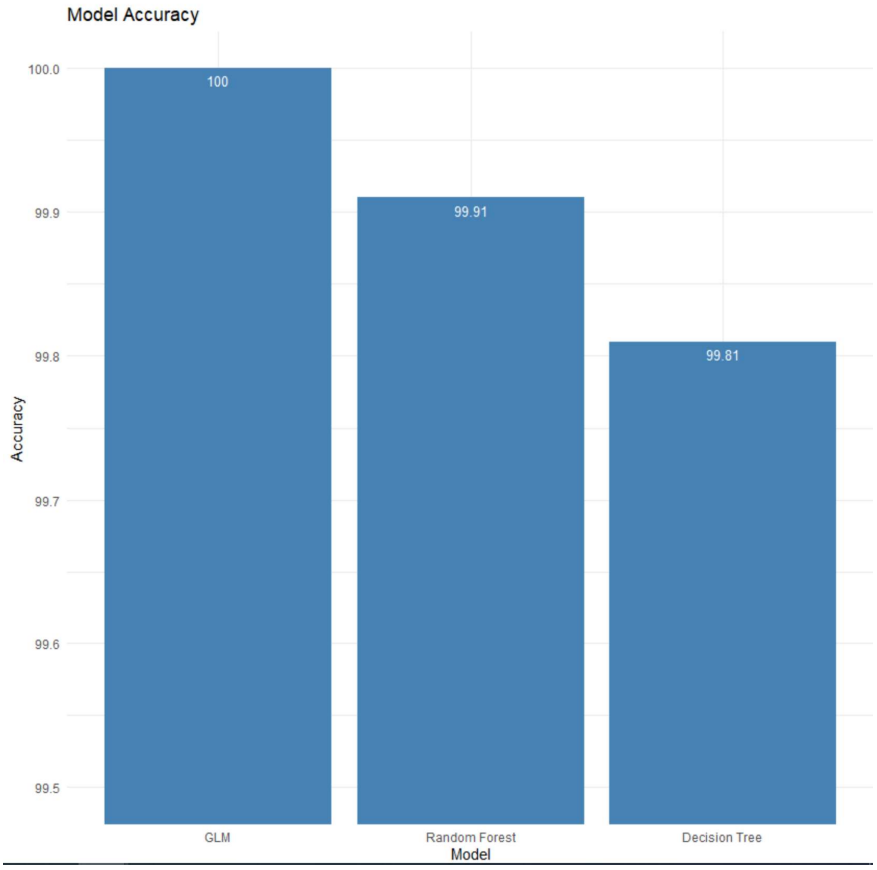

Figure 23 - Model Accuracy

We can see from Figure 23 that all 3 models are more than 99.8% accurate at predicting loan status. Let’s have a closer look at the models’ performance starting with their confusion matrixes from which the accuracy rate is derived.

A confusion matrix is a table we create for each model containing the predicted values of the target variable and the actual values of the target variable. The confusion matrix for each model can be seen in Figure 24 below.

Confusion Matrix Outputs

Decision Tree

	Actual Values	
Predicted Values	Fully Paid	Charged Off
Fully Paid	8033	18
Charged Off	0	1351

Total Obersvations	9402
Total Actual Chargeoffs	1369
Total Predicted Chargeoffs	1351
Of Which True Positives	1351
Of Which False Positives	0
Accuracy Rate	99.81%
Error Rate	0.19%
True Positive Rate	98.69%
True Negative Rate	100.00%
False Positive Rate	1.31%

Random Forest

	Actual Values	
Predicted Values	Fully Paid	Charged Off
Fully Paid	8033	8
Charged Off	0	1361

Total Obersvations	9402
Total Actual Chargeoffs	1369
Total Predicted Chargeoff	1361
Of Which True Positives	1361
Of Which False Positives	0
Accuracy Rate	99.91%
Error Rate	0.09%
True Positive Rate	99.42%
True Negative Rate	100.00%
False Positive Rate	0.58%

GLM

	Actual Values	
Predicted Values	Fully Paid	Charged Off
Fully Paid	8033	0
Charged Off	0	1369

Total Obersvations	9402
Total Actual Chargeoffs	1369
Total Predicted Chargeoff	1369
Of Which True Positives	1369
Of Which False Positives	0
Accuracy Rate	100.00%
Error Rate	0.00%
True Positive Rate	100.00%
True Negative Rate	100.00%
False Positive Rate	0.00%

False Negative	
True Negative	
True Positive	
False Positive	

Figure 24 - Confusion Matrixes

The important terms to understand in relation to confusion matrixes in this case are:

- True Positive: cases where the loan was fully paid and correctly predicted as fully paid.
- True Negative: cases where the loan was charged off and correctly predicted as charged off.
- False Positive: cases where the loan was charged off but incorrectly predicted as fully paid.
- False Negative: cases where the loan was fully paid but incorrectly predicted as charged off.

We can see from the confusion matrix that the Decision Tree model has the highest number of false positives, incorrectly predicting 18 loans as fully paid when they were charged off. Random Forest does a better job only incorrectly classifying 8 loans. GLM on the other hand has correctly classified all loans.

Accuracy is not always a good measure of performance. Take for example a dataset containing 1 million rows. If the model incorrectly predicts 10,000 as charged off when they are fully paid the model would still achieve an accuracy rate of 99%.

An alternative measure of performance is Area Under the Curve (AUC). AUC (Area Under the Curve) is one of the most important metrics for measuring a model's performance (Narkhede, 2018). An ROC (Receiver Operating Characteristics) curve is created plotting the true positive rate against the false positive rate and the area under the curve is measured. The higher the AUC value the more accurate the model is.

We can calculate the AUC using the roc.curve function in the ROSE library. See the ROC curve for Decision Tree and Random Forest in Figures 25 and 26 below. Unfortunately, I was unable to get this to work for the GLM model, but I think we can intuit that the AUC for GLM is 100.

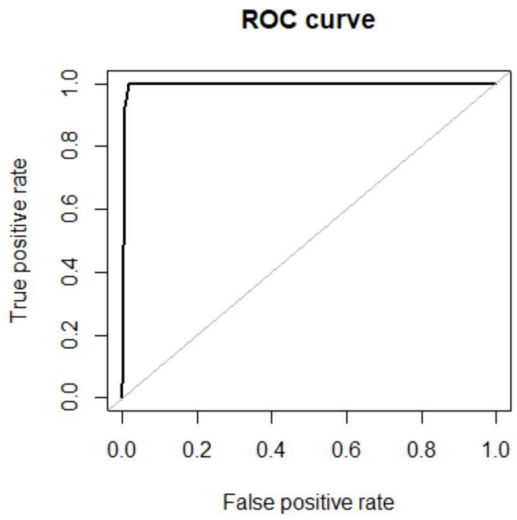

Figure 25 - ROC Curve for Decision Tree

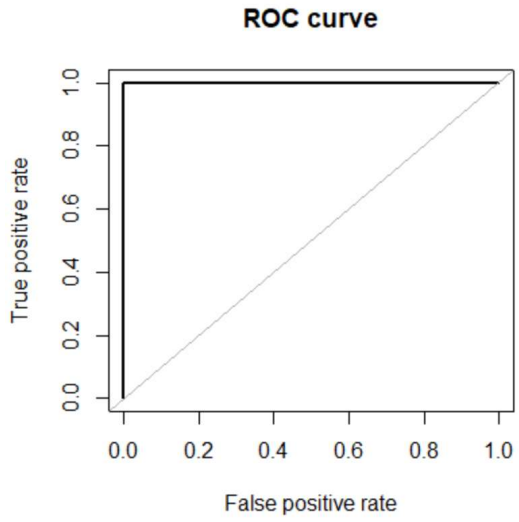

Figure 26 - ROC Curve for Random Forest

The AUC for decision tree is 99.7% and the AUC for random forest is 99.99%.

Imbalanced Datasets

One last thing worth considering in terms of model performance is whether the dataset is imbalanced. For example, datasets containing credit card fraud can be extremely unbalanced where only a few of the transactions are fraudulent and more than 99% are not. As we mentioned earlier decision trees are prone to bias and in cases like this accuracy measures are not reliable.

I was anticipating that this dataset would be imbalanced given the US Federal Reserve quoted a personal loan charge off rate of 2.20% but as we have seen Lending Club's charge off rate is considerably higher.

Nevertheless, there is still some imbalance in the data as we can see from Figure 27 which shows the distribution of values for the target variable in our training set. Figure 27 shows that for every 100 rows of data 85 are "Fully Paid" and 15 are "Charged Off". For a dataset to be balanced this would need to be closer to 50:50.

```
> prop.table(table(trainset$class)) #distribution of values

        1         2
0.8525462 0.1474538
```

Figure 27 - Imbalance in Training Set

There are several ways to address imbalance which aim to even out the number of "Fully Paid" and "Charged Off" examples in the training set:

- **Over sampling**: This method increases the minority class by replicating rows of data were loan_status = "Charged Off".
- **Under sampling**: This method reduces the majority class by removing rows of data where loan_status = "Fully Paid".
- **Over sampling and under sampling**: This is a combination of over sampling and under sampling.
- **Synthetic data generation**: This method generates artificial data by creating its own rows of data where loan_status = "Charged Off".

These methods can be achieved using the ROSE library and can be seen below in Figures 28 – 31.

```
train_over <- ovun.sample(class ~ ., data = trainset, p=0.5, method = "over")$data
table(train_over$class)

dt_over <- rpart(class ~ ., data = train_over, method = "class")
pred_dt_over = predict(dt_over,testset,type='class')
tab_dt_over = table(pred_dt_over,actual = testset$class)  # confusion matrix
accuracy_dt_over = sum(tab_dt_over[row(tab_dt_over)==col(tab_dt_over)])/sum(tab_dt_over)
accuracy_dt_over
tab_dt_over
confusionMatrix(as.factor(pred_dt_over), as.factor(testset$class))
roc.curve(data2$class, predict(dt_over, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_over)
```

Figure 28 - Over Sampling

```
train_under <- ovun.sample(class~., data = trainset, p=0.5, method = "under")$data
table(train_under$class)

dt_under <- rpart(class~., data=train_under, method="class")
pred_dt_under = predict(dt_under, testset, type = 'class')
tab_dt_under = table(pred_dt_under, actual = testset$class) #confusion matrix
accuracy_dt_under = sum(tab_dt_under[row(tab_dt_under)==col(tab_dt_under)])/sum(tab_dt_under)
accuracy_dt_under
tab_dt_under
confusionMatrix(as.factor(pred_dt_under), as.factor(testset$class))
roc.curve(data2$class, predict(dt_under, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_under)
```

Figure 29 - Under Sampling

```
train_both <- ovun.sample(class~., data = trainset, N=nrow(trainset), p=0.5, method="both")$data
table(train_both$class)

dt_both <- rpart(class~., data=train_both, method="class")
pred_dt_both = predict(dt_both, testset, type = 'class')
tab_dt_both = table(pred_dt_both, actual = testset$class) #confusion matrix
accuracy_dt_both = sum(tab_dt_both[row(tab_dt_both)==col(tab_dt_both)])/sum(tab_dt_both)
accuracy_dt_both
tab_dt_both
confusionMatrix(as.factor(pred_dt_both), as.factor(testset$class))
roc.curve(data2$class, predict(dt_both, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_both)
```

Figure 30 – Over sampling and under sampling

```
train_rose <- ROSE(class~., data=trainset[,-10])$data #removed issue date as causing errors algorithm only a
table(train_rose$class)
dt_rose <- rpart(class~., data=train_rose, method="class")
pred_dt_rose = predict(dt_rose, testset, type = 'class')
tab_dt_rose = table(pred_dt_rose, actual = testset$class) #confusion matrix
accuracy_dt_rose = sum(tab_dt_rose[row(tab_dt_rose)==col(tab_dt_rose)])/sum(tab_dt_rose)
accuracy_dt_rose
tab_dt_rose
confusionMatrix(as.factor(pred_dt_rose), as.factor(testset$class))
roc.curve(data2$class, predict(dt_rose, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_rose)
```

Figure 31 - Synthetic Data Generation

Figures 32 and 33 below show the Accuracy and AUC for each model, now including the ROSE models. Unfortunately, this has not helped to improve the accuracy or AUC of the decision tree model. This could be interpreted as meaning that the dataset was not too badly imbalanced after all.

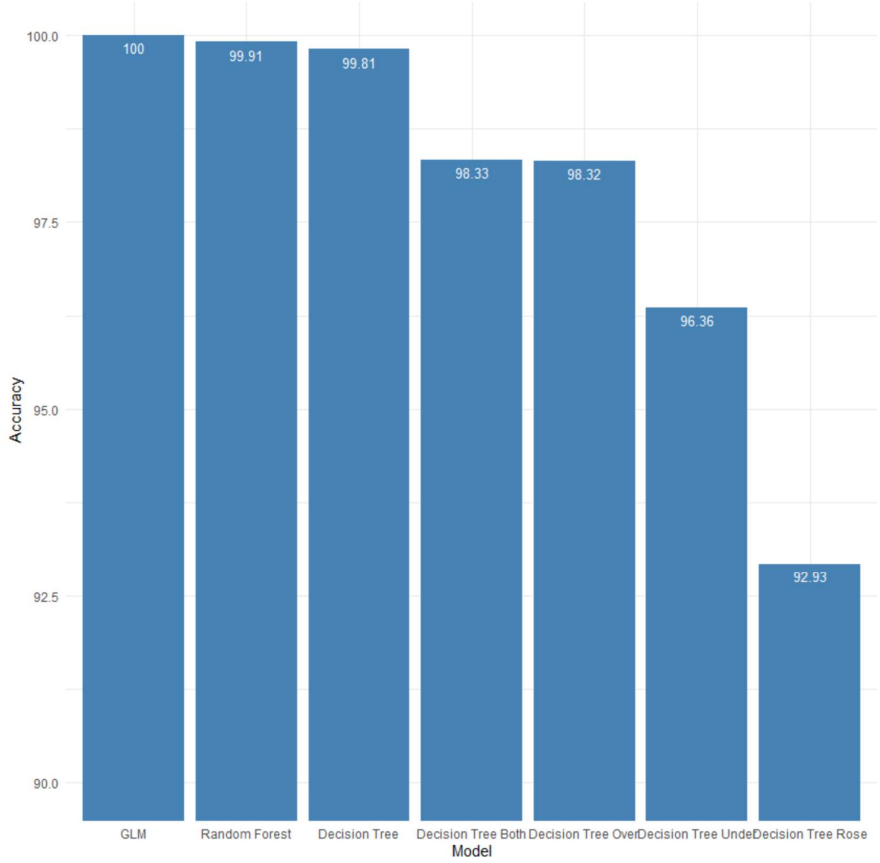

Figure 32 - Accuracy (Including balanced datasets)

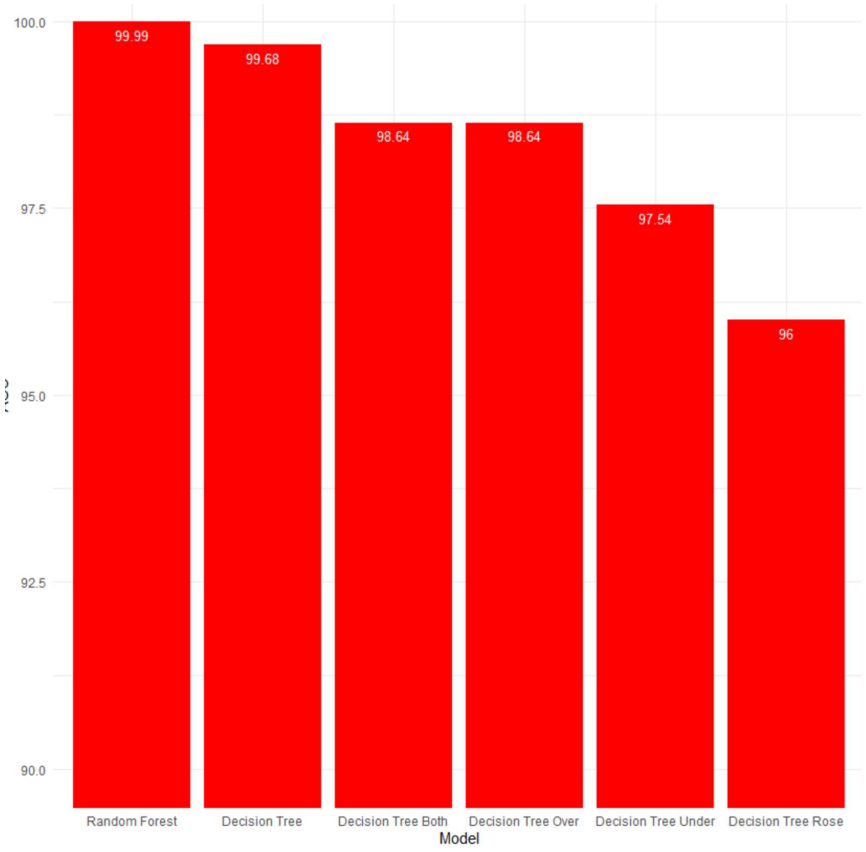

Figure 33 - AUC (Including balanced datasets)

Critical Evaluation/Challenges

There were several challenges with this project which primarily involved receiving errors while running the decision tree, random forest, GLM and correlation algorithms. I found that despite the popularity of RStudio it is quite difficult to find solutions to these issues online. Thankfully I was able to resolve these issues by comparing this dataset with another that I know has worked with these algorithms previously and discovered that many of the variables required reformatting to work.

The algorithms picked mostly the same variables as being important in classifying the loan status these were; the total amount repaid, the last payment amount and account now delinquent flag. I think it is unfortunate that the algorithms identified variables where it may be too late to do anything about the account and help the customer. I have run the algorithms excluding these variables, but it appears that the remaining variables are not good enough indicators to produce a decision tree.

Conclusions

We began by asking if it is possible to predict personal loan charge offs using machine learning algorithms and if so, how accurate are they.

In this paper I have treated this as a classification problem and shown that it can be solved using 3 classification algorithms; Decision Tree, Random Forest and GLM. I have shown that these algorithms can predict charge offs and that they can do so with 100% accuracy in the case of GLM, and over 99% accuracy in the case of decision tree and random forest.

We have tested and evaluated their accuracy using confusion matrixes, ROC curves and additional models which over sampled, under sampled and used synthetically generated data and still achieved accuracy over 96%.

Future work

Future work could be improved with more comprehensive data as the Lending Club dataset offers no insight into the customer's current situation. Home status, income, employment and other demographics are all static as they were collected when the loan was applied for and have not been updated throughout the life of the loan.

A more comprehensive and dynamic dataset would pick up on customer life events, for example a bank would be able to detect a job loss as they would no longer see a salary being paid into the customer's current account. Unfortunately, such a dataset is not available publicly and finding good quality publicly available datasets is a considerable challenge.

References

Asiri, S. (2019). *Machine Learning Classifiers*. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

Brownlee, J. (2019). *Bagging and Random Forest Ensemble Algorithms for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/

Chakure, A. (2019). *Decision Tree Classification*. [online] Medium. Available at: https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac

Chakure, A. (2019). *Random Forest Classification and its implementation*. [online] Medium. Available at: https://towardsdatascience.com/random-forest-classification-and-its-implementation-d5d840dbead0

Narkhede, S. (2019). *Understanding AUC - ROC Curve*. [online] Medium. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Narkhede, S. (2019). *Understanding Confusion Matrix*. [online] Medium. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Phy, V. (2019). *Accuracy is NOT enough for Classification Task*. [online] Medium. Available at: https://towardsdatascience.com/accuracy-is-not-enough-for-classification-task-47fca7d6a8ec

Reserve, F. (2019). *FRB: Charge-Off and Delinquency Rates on Loans and Leases at Commercial Banks*. [online] Federalreserve.gov. Available at: https://www.federalreserve.gov/releases/chargeoff/chgallnsa.htm

Shaikh, R. (2019). *Feature Selection Techniques in Machine Learning with Python*. [online] Medium. Available at: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

Sidana, M. (2019). *Types of classification algorithms in Machine Learning*. [online] Medium. Available at: https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14

Vidhya, A. (2019). *Practical Guide to deal with Imbalanced Classification Problems in R*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

Appendices

RStudio Code

```
#
Read
the
datas
et

          data = read.csv("C:\\Users\\Noel\\Documents\\College\\Project\\Lending
          Club\\Datasets\\loan.csv")

          # See the structure of the dataset

          str(data)


          ################## CLEAN THE DATA #########################

          # How many nulls in each column

          colSums(is.na(data))

          # Remove columns with too many nulls

          data2 <- data[,-which(colSums(is.na(data))>2000)] # remove columns with >2000 NAs in them

          names(data2)

          # Remove rows with Nulls

          data2 <- na.omit(data2)

          # Subset data to "Fully Paid" and "Charged Off" only

          data2 <- subset(data2, loan_status == "Fully Paid" | loan_status == "Charged Off")

          dim(data2)

          # Convert into correct data formats

          library(lubridate)
          library(tidyverse)

          data2$loan_amnt <- as.numeric(data2$loan_amnt)
```

```
data2$funded_amnt <- as.numeric(data2$funded_amnt)
data2$funded_amnt_inv <- as.numeric(data2$funded_amnt_inv)
data2$term <- as.numeric(as.factor(data2$term))
data2$int_rate <- as.numeric(data2$int_rate)
data2$installment <- as.numeric(data2$installment)
data2$grade <- as.numeric(as.factor(data2$grade))
data2$sub_grade <- as.numeric(as.factor(data2$sub_grade))
data2$emp_title <- as.numeric(as.factor(data2$emp_title))
data2$emp_length <- as.numeric(as.factor(data2$emp_length))
data2$home_ownership <- as.numeric(as.factor(data2$home_ownership))
data2$annual_inc <- as.numeric(data2$annual_inc)
data2$verification_status <- as.numeric(as.factor(data2$verification_status))
data2$issue_d <- parse_date_time(data2$issue_d, orders = "my")
data2 <- mutate(data2, class=as.integer(as.factor(ifelse(data2$loan_status == "Fully
Paid",1,2))))
data2$pymnt_plan <- as.numeric(as.factor(data2$pymnt_plan))
data2$desc <- as.numeric(as.factor(data2$desc))
data2$purpose <- as.numeric(as.factor(data2$purpose))
data2$title <- as.numeric(as.factor(data2$title))
data2$zip_code <- as.numeric(as.factor(data2$zip_code))
data2$addr_state <- as.numeric(as.factor(data2$addr_state))
data2$dti <- as.numeric(data2$dti)
data2$delinq_2yrs <- as.numeric(data2$delinq_2yrs)
data2$earliest_cr_line <- parse_date_time(data2$earliest_cr_line, orders = "my")
data2$inq_last_6mths <- as.numeric(data2$inq_last_6mths)
data2$open_acc <- as.numeric(data2$open_acc)
data2$pub_rec <- as.numeric(data2$pub_rec)
data2$revol_bal <- as.numeric(data2$revol_bal)
data2$revol_util <- as.numeric(data2$revol_util)
data2$total_acc <- as.numeric(data2$total_acc)
data2$initial_list_status <- as.numeric(as.factor(data2$initial_list_status))
data2$out_prncp <- as.numeric(data2$out_prncp)
data2$out_prncp_inv <- as.numeric(data2$out_prncp_inv)
data2$total_pymnt <- as.numeric(data2$total_pymnt)
data2$total_pymnt_inv <- as.numeric(data2$total_pymnt_inv)
data2$total_rec_prncp <- as.numeric(data2$total_rec_prncp)
data2$total_rec_int <- as.numeric(data2$total_rec_int)
data2$total_rec_late_fee <- as.numeric(data2$total_rec_late_fee)
data2$recoveries <- as.numeric(data2$recoveries)
data2$collection_recovery_fee <- as.numeric(data2$collection_recovery_fee)
data2$last_pymnt_d <- parse_date_time(data2$last_pymnt_d, orders = "my")
data2$last_pymnt_amnt <- as.numeric(data2$last_pymnt_amnt)
data2$next_pymnt_d <- parse_date_time(data2$next_pymnt_d, orders = "my")
data2$last_credit_pull_d <- parse_date_time(data2$last_credit_pull_d, orders = "my")
data2$collections_12_mths_ex_med <- as.numeric(data2$collections_12_mths_ex_med)
data2$policy_code <- as.numeric(data2$policy_code)
```

```
data2$application_type <- as.numeric(as.factor(data2$application_type))
data2$verification_status_joint <- as.numeric(as.factor(data2$verification_status_joint))
data2$acc_now_delinq <- as.numeric(data2$acc_now_delinq)
data2$chargeoff_within_12_mths <- as.numeric(data2$chargeoff_within_12_mths)
data2$delinq_amnt <- as.numeric(data2$delinq_amnt)
data2$pub_rec_bankruptcies <- as.numeric(data2$pub_rec_bankruptcies)
data2$tax_liens <- as.numeric(data2$tax_liens)
data2$sec_app_earliest_cr_line <- parse_date_time(data2$sec_app_earliest_cr_line, orders =
"my")
data2$hardship_flag <- as.numeric(as.factor(data2$hardship_flag))
data2$hardship_type <- as.numeric(as.factor(data2$hardship_type))
data2$hardship_reason <- as.numeric(as.factor(data2$hardship_reason))
data2$hardship_status <- as.numeric(as.factor(data2$hardship_status))
data2$hardship_start_date <- parse_date_time(data2$hardship_start_date, orders = "my")
data2$hardship_end_date <- parse_date_time(data2$hardship_end_date, orders = "my")
data2$payment_plan_start_date <- parse_date_time(data2$payment_plan_start_date, orders =
"my")
data2$hardship_loan_status <- as.numeric(as.factor(data2$hardship_loan_status))
data2$disbursement_method <- as.numeric(as.factor(data2$disbursement_method))
data2$debt_settlement_flag <- as.numeric(as.factor(data2$debt_settlement_flag))
data2$debt_settlement_flag_date <- parse_date_time(data2$debt_settlement_flag_date, orders
= "my")
data2$settlement_status <- as.numeric(as.factor(data2$settlement_status))
data2$settlement_date <- parse_date_time(data2$settlement_date, orders = "my")

# Subset data to loans issues in 2018 only (now that dates have been formatted)

data2 <- subset(data2, issue_d >= "2018-01-01" & issue_d <= "2018-12-31" )

str(data2)



##################### DO CORRELATION ############################

library(corrplot)
library(GGally)

c <- cor(data2[,unlist(lapply(data2, is.numeric))], use = "complete.obs")

ggcorr(c, label = FALSE, hjust= 0.75, layout.exp = 0) #Plot the correlations




################# Feature selection ########################
```

```
# Choose features to keep going forward

cols <-
c("loan_amnt","term","int_rate","installment","grade","sub_grade","home_ownership","annual_
inc"

,"verification_status","issue_d","class","pymnt_plan","purpose","addr_state","dti","inq_las
t_6mths",

"open_acc","pub_rec","revol_bal","revol_util","total_acc","initial_list_status","total_pymn
t","total_rec_prncp",

"last_pymnt_amnt","acc_now_delinq","delinq_amnt","pub_rec_bankruptcies","hardship_flag")
#removed emp_length for now

data2 <- data2[names(data2) %in% cols]




################## MODELLING ################################

library(rpart)
library(rpart.plot)
library(ROCR)
library(ROSE)

n=nrow(data2)
indexes = sample(n,n*(80/100))
trainset = data2[indexes,]
testset = data2[-indexes,]

dim(data2)
dim(testset)
dim(trainset)

# Decision Tree

dt <- rpart(class ~ ., data = trainset, method = "class")
predictedvalues = predict(dt, testset, type = 'class')
tab_dt = table(predictedvalues, actualvalues=testset[,29]) #confusion matrix
accuracy_dt=sum(tab_dt[row(tab_dt)==col(tab_dt)])/sum(tab_dt)
accuracy_dt
rpart.plot(dt, cex=0.75) # Visualise the Decison Tree
```

```
tab_dt #Confusion matrix
confusionMatrix(predictedvalues, as.factor(testset$class))
roc.curve(data2$class, predict(dt, data2, type = "prob")[,1],plot=TRUE)

# Random Forest

library(randomForest)
library(ggplot2)

rf=randomForest(as.factor(class) ~ ., type='class', data = trainset, ntree=10) #as.factor
makes it run much faster
predictedvalues = predict(rf, testset, type='class')
tab_rf = table(predictedvalues, actualvalues=testset[,29]) #confusion matrix
accuracy_rf= sum(tab_rf[row(tab_rf)==col(tab_rf)])/sum(tab_rf)
accuracy_rf
tab_rf
roc.curve(data2$class, predict(rf, data2, type = "prob")[,1],plot=TRUE)

# Visualise Variable Importance
# Ggplot requires a dataframe, so convery VarImp in to a dataframe

vi <- data.frame(Variable = c(names(data2[,-29])), #removed the class variable as thats not
in VarImp
                 Overall = c(varImp(rf)[1]))

v <- ggplot(data=vi,aes(x = reorder(Variable, Overall),y=Overall))+
  geom_bar(stat="identity", fill="steelblue") +
  theme_minimal() +
  #coord_cartesian(ylim = c(90,100)) +
  labs(x = "Variable")
v + coord_flip()

# GLM

trainset.glm <- glm(class-1 ~.,trainset, family="binomial") # class-1 makes the variable
values 0 and 1
phati=predict(trainset.glm,testset, type="response")  # p hat i
predictedvalues=rep(0,length(phati))
predictedvalues[phati>0.5]=1   # probability of x being 1, if p<0.5 then x=0
actual=testset[,29]-1
tab_glm=table( predictedvalues, actualvalues=testset[,29]-1)
accuracy_gl=mean(predictedvalues == testset[,29]-1)
accuracy_gl #higher better
tab_glm #Confusion matrix
summary(trainset.glm)
```

```
############## TESTING AND EVALUATION ################

# Plot the accuracy of each model

df <- data.frame(Model = c("Decision Tree",
                           "Random Forest",
                           "GLM"
),
Accuracy = c(round(accuracy_dt,4)*100,
             round(accuracy_rf,4)*100,
             round(accuracy_gl,4)*100
))
head(df)

p<-ggplot(data=df,aes(x = reorder(Model, -Accuracy),y=Accuracy))+
  geom_bar(stat="identity", fill="steelblue") +
  geom_text(aes(label=Accuracy), vjust=1.6, color="white",size=3.5) +
  theme_minimal() +
  coord_cartesian(ylim = c(99.5,100)) +
  labs(x = "Model")
p + ggtitle("Model Accuracy")

#

# Confusion matrixes (matrices)

tab_dt
tab_rf
tab_glm


# AUC (Area Under the curve)

# Decision Tree

rc_dt <- roc.curve(data2$class, predict(dt, data2, type = "prob")[,1])
rc_dt <- as.character(rc_dt)
rc_dt <- as.numeric(rc_dt[2])
rc_dt

# Random Forest
```

```
rc_rf <- roc.curve(data2$class, predict(rf, data2, type = "prob")[,1])
rc_rf <- as.character(rc_rf)
rc_rf <- as.numeric(rc_rf[2])
rc_rf

plot(rc_dt, rc_rf)



######################### Deal with imbalanced data ###################################

# Balancing the dataset (if necessary)

# DECISION TREE
# OVERSAMPLING THE MINORITY CLASS

dim(trainset)
table(trainset$class)
prop.table(table(trainset$class)) #distribution of values

train_over <- ovun.sample(class ~ ., data = trainset, p=0.5, method = "over")$data
table(train_over$class)

dt_over <- rpart(class ~ ., data = train_over, method = "class")
pred_dt_over = predict(dt_over,testset,type='class')
tab_dt_over = table(pred_dt_over,actual = testset$class)  # confusion matrix
accuracy_dt_over = sum(tab_dt_over[row(tab_dt_over)==col(tab_dt_over)])/sum(tab_dt_over)
accuracy_dt_over
tab_dt_over
confusionMatrix(as.factor(pred_dt_over), as.factor(testset$class))
roc.curve(data2$class, predict(dt_over, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_over)


# UNDERSAMPLING THE MAJORITY CLASS

train_under <- ovun.sample(class~., data = trainset, p=0.5, method = "under")$data
table(train_under$class)

dt_under <- rpart(class~., data=train_under, method="class")
pred_dt_under = predict(dt_under, testset, type = 'class')
tab_dt_under = table(pred_dt_under, actual = testset$class) #confusion matrix
accuracy_dt_under =
sum(tab_dt_under[row(tab_dt_under)==col(tab_dt_under)])/sum(tab_dt_under)
accuracy_dt_under
tab_dt_under
```

```
confusionMatrix(as.factor(pred_dt_under), as.factor(testset$class))
roc.curve(data2$class, predict(dt_under, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_under)


# BOTH OVERSAMPLING AND UNDERSAMPLING

train_both <- ovun.sample(class~., data = trainset, N=nrow(trainset), p=0.5,
method="both")$data
table(train_both$class)

dt_both <- rpart(class~., data=train_both, method="class")
pred_dt_both = predict(dt_both, testset, type = 'class')
tab_dt_both = table(pred_dt_both, actual = testset$class) #confusion matrix
accuracy_dt_both = sum(tab_dt_both[row(tab_dt_both)==col(tab_dt_both)])/sum(tab_dt_both)
accuracy_dt_both
tab_dt_both
confusionMatrix(as.factor(pred_dt_both), as.factor(testset$class))
roc.curve(data2$class, predict(dt_both, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_both)


# SYNTHETIC DATA GENERATION

#need to remove no-numeric columns

names(trainset)

train_rose <- ROSE(class~., data=trainset[,-10])$data #removed issue date as causing errors
algorithm only acceots numerical variables
table(train_rose$class)
dt_rose <- rpart(class~., data=train_rose, method="class")
pred_dt_rose = predict(dt_rose, testset, type = 'class')
tab_dt_rose = table(pred_dt_rose, actual = testset$class) #confusion matrix
accuracy_dt_rose = sum(tab_dt_rose[row(tab_dt_rose)==col(tab_dt_rose)])/sum(tab_dt_rose)
accuracy_dt_rose
tab_dt_rose
confusionMatrix(as.factor(pred_dt_rose), as.factor(testset$class))
roc.curve(data2$class, predict(dt_rose, data2, type = "prob")[,1],plot=TRUE)
rpart.plot(dt_rose)

## So which model was most "accurate"?

# Plot accuracy for each model

df <- data.frame(Model = c("Decision Tree",
```

```
                            "Random Forest",
                            "GLM",
                            "Decision Tree Over",
                            "Decision Tree Under",
                            "Decision Tree Both",
                            "Decision Tree Rose"
),
Accuracy = c(round(accuracy_dt,4)*100,
             round(accuracy_rf,4)*100,
             round(accuracy_gl,4)*100,
             round(accuracy_dt_over,4)*100,
             round(accuracy_dt_under,4)*100,
             round(accuracy_dt_both,4)*100,
             round(accuracy_dt_rose,4)*100
))
head(df)

p<-ggplot(data=df,aes(x = reorder(Model, -Accuracy),y=Accuracy))+
  geom_bar(stat="identity", fill="steelblue") +
  geom_text(aes(label=Accuracy), vjust=1.6, color="white",size=3.5) +
  theme_minimal() +
  coord_cartesian(ylim = c(90,100)) +
  labs(x = "Model")
p


# Plot AUC (Area UNder the Curve) for each model

#Decision Tree:
rc_dt <- roc.curve(data2$class, predict(dt, data2, type = "prob")[,1])
rc_dt <- as.character(rc_dt)
rc_dt <- as.numeric(rc_dt[2])
rc_dt

# Random Forest:
rc_rf <- roc.curve(data2$class, predict(rf, data2, type = "prob")[,1])
rc_rf <- as.character(rc_rf)
rc_rf <- as.numeric(rc_rf[2])
rc_rf

# Decision Tree Over Sampled:

rc_dt_over <- roc.curve(data2$class, predict(dt_over, data2, type = "prob")[,1])
rc_dt_over <- as.character(rc_dt_over)
rc_dt_over <- as.numeric(rc_dt_over[2])
rc_dt_over
```

```
# Decision Tree Under Sampled:

rc_dt_under <- roc.curve(data2$class, predict(dt_under, data2, type = "prob")[,1])
rc_dt_under <- as.character(rc_dt_under)
rc_dt_under <- as.numeric(rc_dt_under[2])
rc_dt_under

# Decision Over and Under Sampled:

rc_dt_both <- roc.curve(data2$class, predict(dt_both, data2, type = "prob")[,1])
rc_dt_both <- as.character(rc_dt_both)
rc_dt_both <- as.numeric(rc_dt_both[2])
rc_dt_both

# Decision TRee with Synthetic Data

rc_dt_rose <- roc.curve(data2$class, predict(dt_rose, data2, type = "prob")[,1])
rc_dt_rose <- as.character(rc_dt_rose)
rc_dt_rose <- as.numeric(rc_dt_rose[2])
rc_dt_rose

df2 <- data.frame(Model = c("Decision Tree",
                            "Random Forest",
                            "Decision Tree Over",
                            "Decision Tree Under",
                            "Decision Tree Both",
                            "Decision Tree Rose"
),
AUC = c(round(rc_dt,4)*100,
        round(rc_rf,4)*100,
        round(rc_dt_over,4)*100,
        round(rc_dt_under,4)*100,
        round(rc_dt_both,4)*100,
        round(rc_dt_rose,4)*100
))

head(df2)

p2 <- ggplot(data=df2,aes(x = reorder(Model, -AUC),y=AUC))+
  geom_bar(stat="identity", fill="red") +
  geom_text(aes(label=AUC), vjust=1.6, color="white",size=3.5) +
  theme_minimal() +
  coord_cartesian(ylim = c(90,100)) +
  labs(x = "Model")
p2
```