edx Harvardx CYO Project

Kaggle - Give me Some Credit

Suresh Thyagarajan

5/25/2020

Contents

Dedication

I dedicate this project to my family and friends who have been a pillar of support.

Acknowledgement

I take this opportunity to place on record my sincere gratitude and appreciation to Prof. Rafael Irizarry for his most wonderful insights and explanations during the course.

Introduction

Banks are the fulcrum for growth in any economy. Banks act as catalysts by providing credit or loan to companies and individuals. This spurs industrial growth as well as increasing consumer spending. A loan or a credit is deemed as an asset for banking industry. Unfortunately, in the real world, not all assets perform. There are instances when loan/credit is not serviced by debtors and in some instances leads to defaults or delinquencies. The risk arising on account

of lending is characterized as Credit Risk. This, is the biggest challenge that any commercial bank would face.

Wikipedia defines *Credit Risk* as the risk of default on a debt that may arise from a borrower failing to make required payments.

Banks would have to mitigate this challenge if they have to survive and sustain in the long run. Qualitative factors such as the industry, past track record,reputation of the borrower, management philosophy etc are analysed before sanctioning a loan. However, this is not emotion agnostic and does not give clarity in decision making. Hence, the need for Credit Scoring algorithms. These algorithms predict the probability of default thus enabling banks to decide on whether to grant a loan or not.

This project aims to leverage on machine learning techniques such as logistics regression, decision tree and ROC Curve give predictions on probable defaulters in 2 years.

At the end of it, we would also make a competition submission in the prescribed format. This competition was conducted in 2011 and has since been closed. The purpose of choosing this project is to demonstrate understanding of course material. This competition though closed, provides the bandwidth to display learning from the course.

Kaggle - Give Me Some Credit https://www.kaggle.com/c/GiveMeSomeCredit/overview was originally a competition to build a most efficient model to predict delinquencies in 2 years.

We use the training data-set as provided in Kagle - Give Me Some Credit https://www.kaggle.com/c/GiveMeSomeCredit/data?select=cs-training.csv.
The test data-set for competition submission is https://github.com/SureshThyagara1/CYO—Kaggle-Give-Me-Some-Credit/blob/master/Data/cs-test.csv.zip

The training data-set has been used to build models.

The data-sets are also available in the following github repository: https://github.com/SureshThyagara1/CYO—Kaggle-Give-Me-Some-Credit/tree/master/Data

Following PC was used for this project

```
print('Operating System:')
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
```

```
## os              mingw32
## system          x86_64, mingw32
## status
## major           4
## minor           0.0
## year            2020
## month           04
## day             24
## svn rev         78286
## language        R
## version.string R version 4.0.0 (2020-04-24)
## nickname        Arbor Day
```

Methods and Analysis

Data Import

The data-sets needs to be downloaded into the working directory and unzipped for extraction and reading. Please do note, only setting up of the working directory has an absolute path. Further to that, all paths would be relative.

```
# First, we set the working directory
setwd("C:\\Data Science\\CYO\\Credit Score")
# Check the the working directory
getwd()
```

```
## [1] "C:/Data Science/CYO/Credit Score"
```

```
# Download the compressed folder with the dataset into the working directory
# https://www.kaggle.com/c/GiveMeSomeCredit/data?select=cs-training.csv
# Following codes are used to
# 1) unzip a compressed folder and
# 2) Download trainig data which is in csv format
# Create an object called credit_data with the dowloaded dataset
credit_data <- read.csv(unz("cs-training.csv.zip","cs-training.csv"))
# Following codes are used to
# 1) unzip a compressed folder and
# 2) download test data for competition which is in csv format
credit_data_comp <- read.csv(unz("cs-test.csv.zip","cs-test.csv"))
```

Let's view the structure of downloaded training data-set:

```
# View structure of downloaded dataset
str(credit_data)
```

```
## 'data.frame':    150000 obs. of  12 variables:
##  $ X                                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ SeriousDlqin2yrs                    : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ RevolvingUtilizationOfUnsecuredLines: num  0.766 0.957 0.658 0.234 0.907 ...
##  $ age                                 : int  45 40 38 30 49 74 57 39 27 57 ...
##  $ NumberOfTime30.59DaysPastDueNotWorse: int  2 0 1 0 1 0 0 0 0 0 ...
##  $ DebtRatio                           : num  0.803 0.1219 0.0851 0.036 0.0249 ...
##  $ MonthlyIncome                       : int  9120 2600 3042 3300 63588 3500 NA 3500 NA 2
##  $ NumberOfOpenCreditLinesAndLoans     : int  13 4 2 5 7 3 8 8 2 9 ...
##  $ NumberOfTimes90DaysLate             : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ NumberRealEstateLoansOrLines        : int  6 0 0 0 1 1 3 0 0 4 ...
##  $ NumberOfTime60.89DaysPastDueNotWorse: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NumberOfDependents                  : int  2 1 0 0 0 1 0 0 NA 2 ...
```

Data Processing

We now explore and visualize data. Here, we identify missing data as well as outliers. The same are treated to fit into the data-set.

Let's begin by having a quick overview of the downloaded data

```
# An overview of downloaded data
head(credit_data[1:3,])
```

```
##   X SeriousDlqin2yrs RevolvingUtilizationOfUnsecuredLines age
## 1 1                1                            0.7661266  45
## 2 2                0                            0.9571510  40
## 3 3                0                            0.6581801  38
##   NumberOfTime30.59DaysPastDueNotWorse  DebtRatio MonthlyIncome
## 1                                    2 0.80298213          9120
## 2                                    0 0.12187620          2600
## 3                                    1 0.08511338          3042
##   NumberOfOpenCreditLinesAndLoans NumberOfTimes90DaysLate
## 1                              13                       0
## 2                               4                       0
## 3                               2                       1
##   NumberRealEstateLoansOrLines NumberOfTime60.89DaysPastDueNotWorse
```

```
## 1                            6                                       0
## 2                            0                                       0
## 3                            0                                       0
##   NumberOfDependents
## 1                  2
## 2                  1
## 3                  0
```

```
# Names of the variables in the data
names(credit_data)
```

```
##  [1] "X"
##  [2] "SeriousDlqin2yrs"
##  [3] "RevolvingUtilizationOfUnsecuredLines"
##  [4] "age"
##  [5] "NumberOfTime30.59DaysPastDueNotWorse"
##  [6] "DebtRatio"
##  [7] "MonthlyIncome"
##  [8] "NumberOfOpenCreditLinesAndLoans"
##  [9] "NumberOfTimes90DaysLate"
## [10] "NumberRealEstateLoansOrLines"
## [11] "NumberOfTime60.89DaysPastDueNotWorse"
## [12] "NumberOfDependents"
```

Data Exploration and Visualization

`credit_data` data set has 12 variables in total. The first variable is a serial number which does not contribute to analysis.

```
# Removing "X" which is for serial number
# "X" does not qualify to be a explaratory or a predictor variable
credit_data$X = NULL
# Check if the "X" variable has been removed from dataset
names(credit_data)
```

```
##  [1] "SeriousDlqin2yrs"
##  [2] "RevolvingUtilizationOfUnsecuredLines"
##  [3] "age"
##  [4] "NumberOfTime30.59DaysPastDueNotWorse"
##  [5] "DebtRatio"
##  [6] "MonthlyIncome"
##  [7] "NumberOfOpenCreditLinesAndLoans"
```

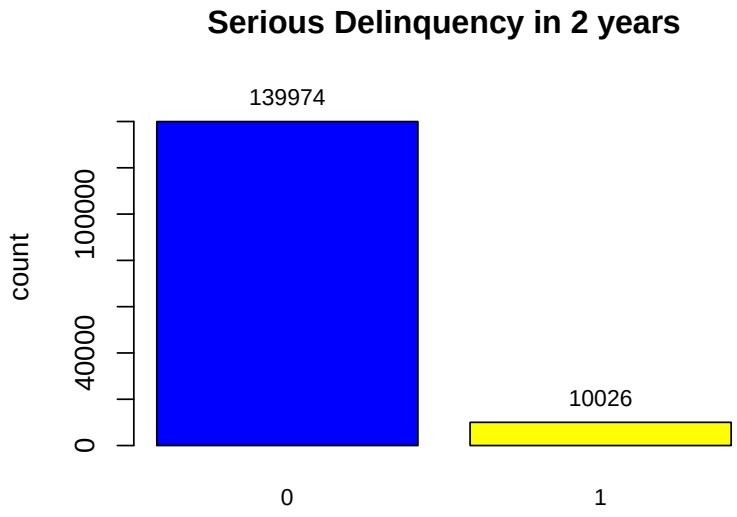
```
##  [8] "NumberOfTimes90DaysLate"
##  [9] "NumberRealEstateLoansOrLines"
## [10] "NumberOfTime60.89DaysPastDueNotWorse"
## [11] "NumberOfDependents"
```

Data Summary

Let's explore the data-set now.

1) **SeriousDlqin2yrs** - Person experienced 90 days past due delinquency or worse Values in this variable are 0 for no and 1 for yes on serious delinquency in the last 2 years. This is the outcome variable of the data-set.

```
library(epiDisplay)
tab1(credit_data$SeriousDlqin2yrs,sort.group = "decreasing",
     cum.percent = FALSE, bar.values = "frequency", cex = 0.8, cex.names = 0.8,
     main = "Serious Delinquency in 2 years", xlab = "Delinquency",
     ylab = "count", col = c("blue", "yellow"))
```



```
## credit_data$SeriousDlqin2yrs :
##         Frequency Percent
## 0          139974    93.3
## 1           10026     6.7
##   Total    150000   100.0
```

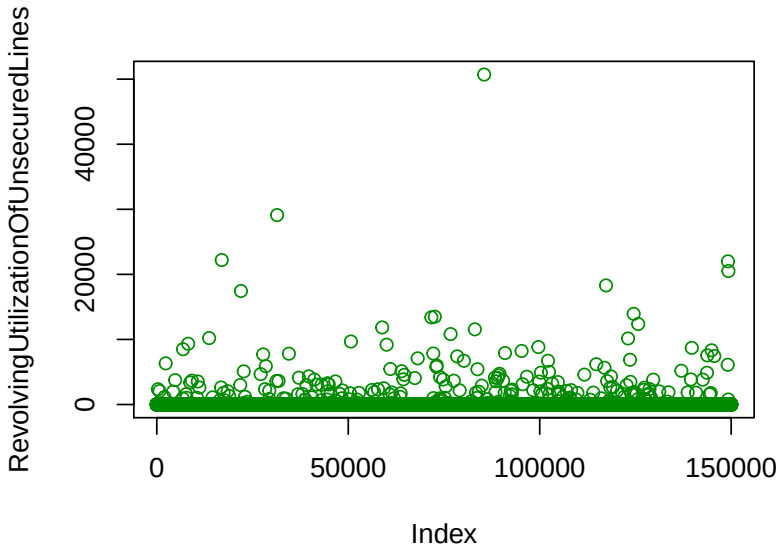
This shows 6.7 % had serious delinquency in last 2 years

2) **RevolvingUtilizationofUnsecuredLines** - Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits. This is a predictor variable.

```
summary(credit_data$RevolvingUtilizationOfUnsecuredLines)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     0.03     0.15     6.05     0.56 50708.00
```

```
plot(credit_data$RevolvingUtilizationOfUnsecuredLines,
     col = "green4",
     ylab = "RevolvingUtilizationOfUnsecuredLines")
```



```r
sum(credit_data$RevolvingUtilizationOfUnsecuredLines > 1)
```

```
## [1] 3321
```

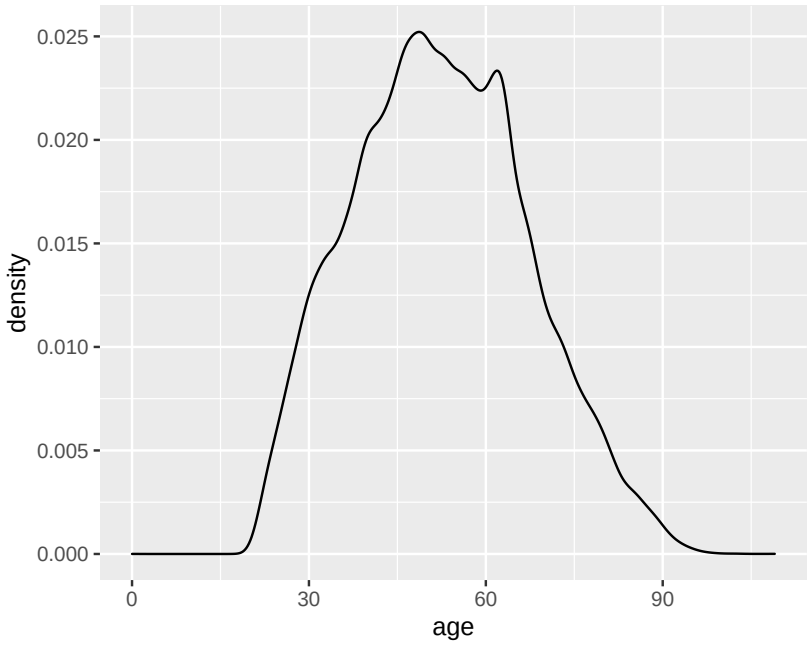
This data should be around 1 being a proportion. However, we find that a number of data (3321) have significantly high values. This outlier treatment would be dealt while handing outliers in one of the following sections.

3) **age** - Age of borrower in years. This is a predictor variable.

```r
summary(credit_data$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    41.0    52.0    52.3    63.0   109.0
```

```r
library(dplyr)
library(ggplot2)
credit_data %>% ggplot(aes(age)) + geom_density()
```



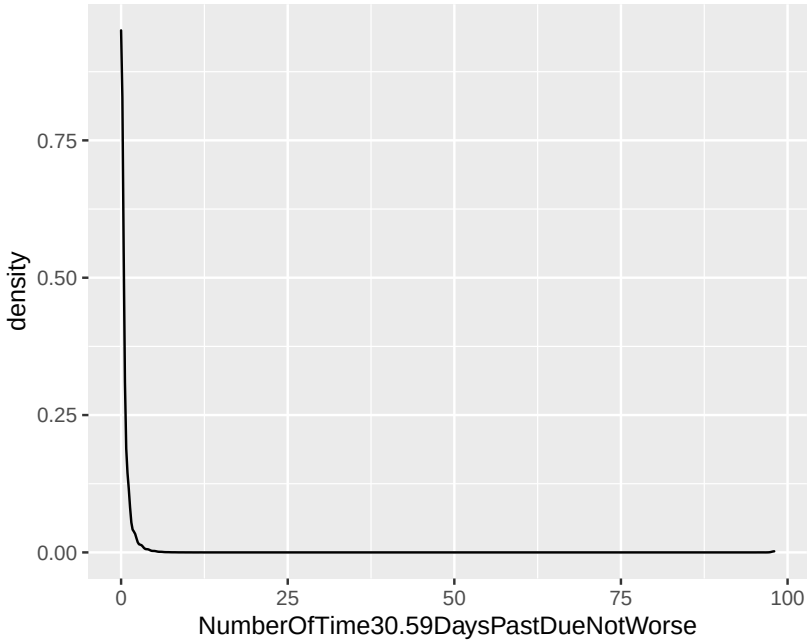
This data gives the age profile of the borrowers with the maximum number being in the 30 to 60 years range

4) **NumberOfTime30-59DaysPastDueNotWorse** - Number of times borrower has been 30-59 days past due but no worse in the last 2 years. This is a predictor variable.

```
summary(credit_data$NumberOfTime30.59DaysPastDueNotWorse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.421   0.000  98.000
```

```
credit_data %>% ggplot(aes(NumberOfTime30.59DaysPastDueNotWorse)) + geom_density()
```




```
table(credit_data$NumberOfTime30.59DaysPastDueNotWorse)
```

```
##
##      0      1      2      3      4      5      6      7      8      9     10
## 126018  16033   4598   1754    747    342    140     54     25     12      4
##     11     12     13     96     98
##      1      2      1      5    264
```

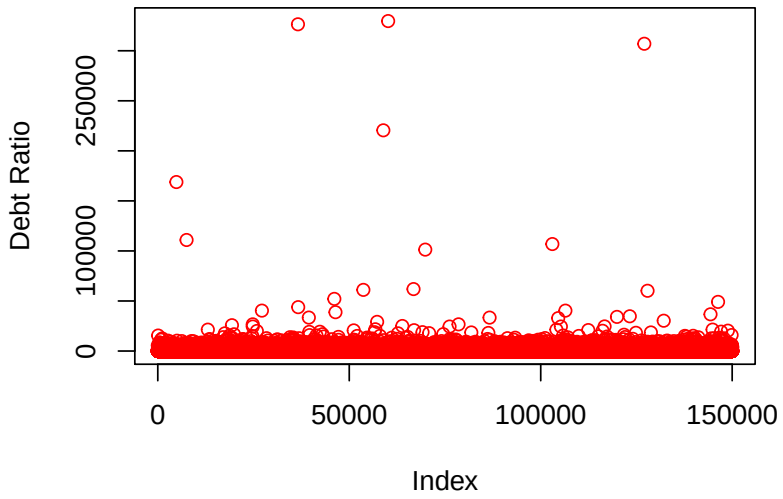
Vast majority of the borrowers did not have past dues for 30 to 59 days.

5) **DebtRatio** - Monthly debt payments, alimony,living costs divided by monthly gross income. This is a predictor variable.

```
summary(credit_data$DebtRatio)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##       0.0       0.2       0.4    353.0       0.9 329664.0
```

```
plot(credit_data$DebtRatio, col = "red", ylab = "Debt Ratio")
```



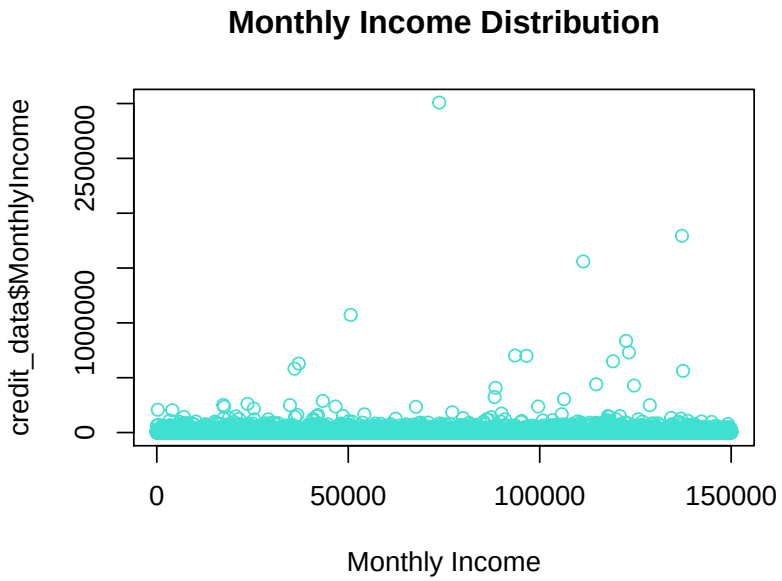

```
sum(credit_data$DebtRatio > 1)
```

```
## [1] 35137
```

Debt ratio is a proportion/percentage. There are a possible 35137 outliers in this predictor variable. This would be dealt with in outlier treatment section.

6) **MonthlyIncome** - Monthly income. This is a predictor variable.

```
plot(credit_data$MonthlyIncome,
main = "Monthly Income Distribution",
xlab = "Monthly Income", col = "turquoise")
```



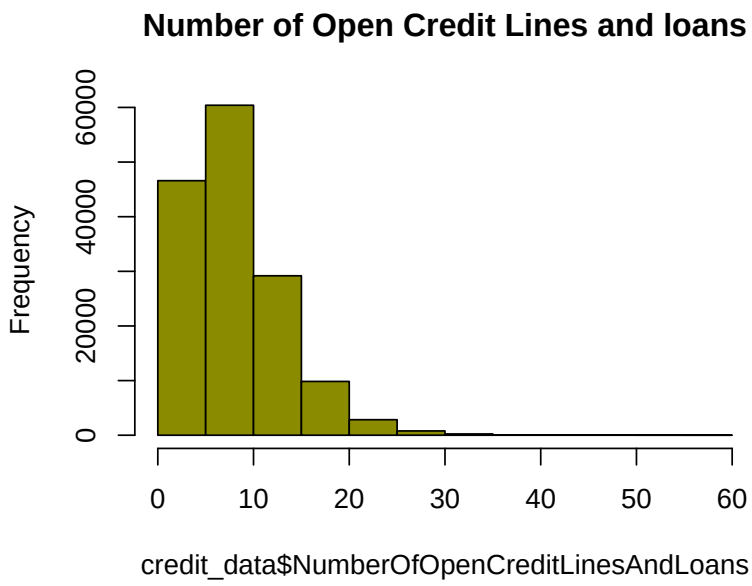
MonthlyIncome variable consists of missing values as well as outliers. They will be dealt with in a later section.

7) **NumberOfOpenCreditLinesAndLoans** - Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards). This is a predictor variable.

```
summary(credit_data$NumberOfOpenCreditLinesAndLoans)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.000   8.000   8.453  11.000  58.000
```

```r
hist(credit_data$NumberOfOpenCreditLinesAndLoans,
 main = "Number of Open Credit Lines and loans", col = "yellow4")
```




```r
table(credit_data$NumberOfOpenCreditLinesAndLoans)
```

```
##
##     0     1     2     3     4     5     6     7     8     9    10    11    12
##  1888  4438  6666  9058 11609 12931 13614 13245 12562 11355  9624  8321  7005
##    13    14    15    16    17    18    19    20    21    22    23    24    25
##  5667  4546  3645  3000  2370  1874  1433  1169   864   685   533   422   337
##    26    27    28    29    30    31    32    33    34    35    36    37    38
##   239   194   150   114    88    74    52    47    35    27    18     7    13
##    39    40    41    42    43    44    45    46    47    48    49    50    51
##     9    10     4     8     8     2     8     3     2     6     4     2     2
##    52    53    54    56    57    58
##     3     1     4     2     2     1
```

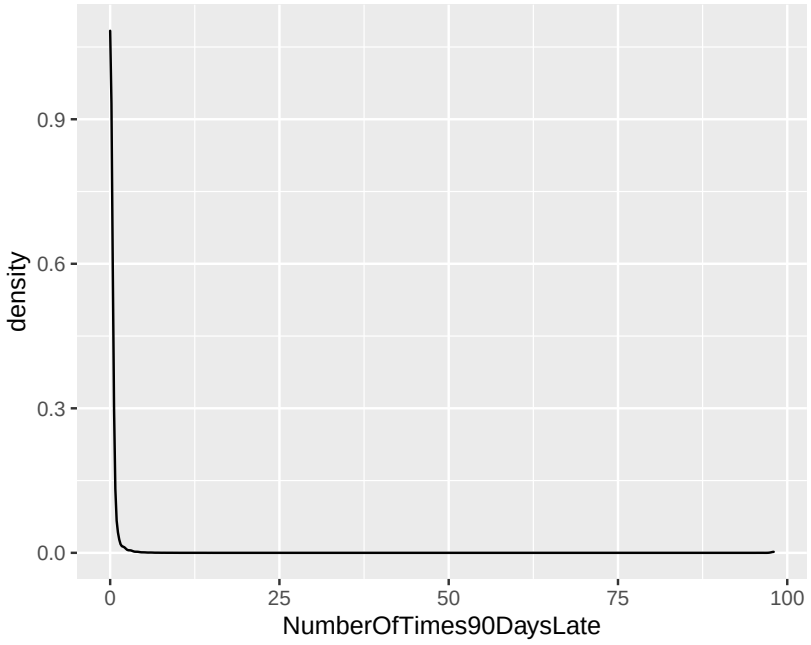
The number of open credit lines and loans average 8.453 and has a maximum value of 58. There are outliers in this as well with very high number of NumberOfOpenCreditLinesAndLoans.

8) **NumberOfTimes90DaysLate** - Number of times borrower has been 90 days or more past due. This is a predictor variable.

```
summary(credit_data$NumberOfTimes90DaysLate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.266   0.000  98.000
```

```
credit_data %>% ggplot(aes(NumberOfTimes90DaysLate)) + geom_density()
```




```
table(credit_data$NumberOfTimes90DaysLate)
```

```
##
##      0      1      2      3      4      5      6      7      8      9     10
## 141662   5243   1555    667    291    131     80     38     21     19      8
##     11     12     13     14     15     17     96     98
##      5      2      4      2      2      1      5    264
```

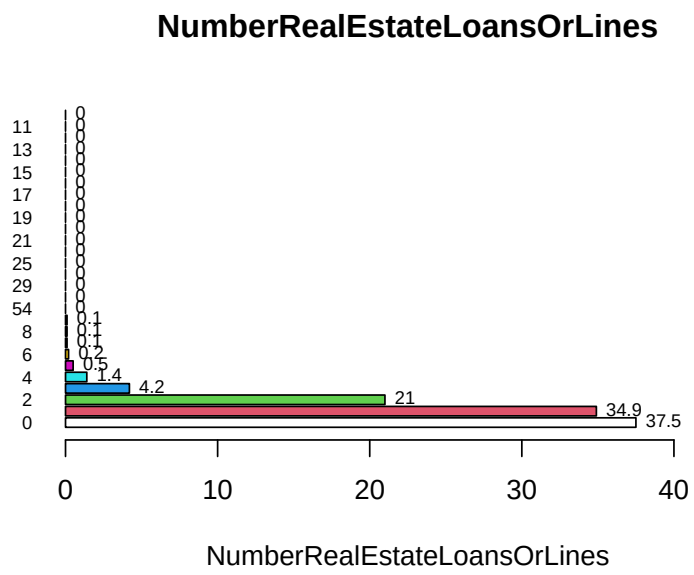
Again, a vast majority of the borrowers did not have even a single occasion when they have been past due for 90 days or more

9) **NumberRealEstateLoansOrLines** - Number of mortgage and real estate loans including home equity lines of credit. This is a predictor variable.

```
summary(credit_data$NumberRealEstateLoansOrLines)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   1.018   2.000  54.000
```

```
library(epiDisplay)
tab1(credit_data$NumberRealEstateLoansOrLines ,
     sort.group = "decreasing", cum.percent = FALSE,
     bar.values = "percent",
     main = "NumberRealEstateLoansOrLines",
     xlab = "NumberRealEstateLoansOrLines",
     ylab = "count",horiz = TRUE, cex = .7,cex.names = 0.7)
```




```
## credit_data$NumberRealEstateLoansOrLines :
##         Frequency Percent
## 0           56188    37.5
## 1           52338    34.9
## 2           31522    21.0
```

```
## 3             6300     4.2
## 4             2170     1.4
## 5              689     0.5
## 6              320     0.2
## 7              171     0.1
## 8               93     0.1
## 9               78     0.1
## 10              37     0.0
## 11              23     0.0
## 12              18     0.0
## 13              15     0.0
## 15               7     0.0
## 14               7     0.0
## 17               4     0.0
## 16               4     0.0
## 25               3     0.0
## 23               2     0.0
## 20               2     0.0
## 19               2     0.0
## 18               2     0.0
## 54               1     0.0
## 32               1     0.0
## 29               1     0.0
## 26               1     0.0
## 21               1     0.0
##   Total     150000   100.0
```

```
table(credit_data$NumberRealEstateLoansOrLines)
```

```
##
##     0     1     2     3     4     5     6     7     8     9    10    11    12
## 56188 52338 31522  6300  2170   689   320   171    93    78    37    23    18
##    13    14    15    16    17    18    19    20    21    23    25    26    29
##    15     7     7     4     4     2     2     2     1     2     3     1     1
##    32    54
##     1     1
```

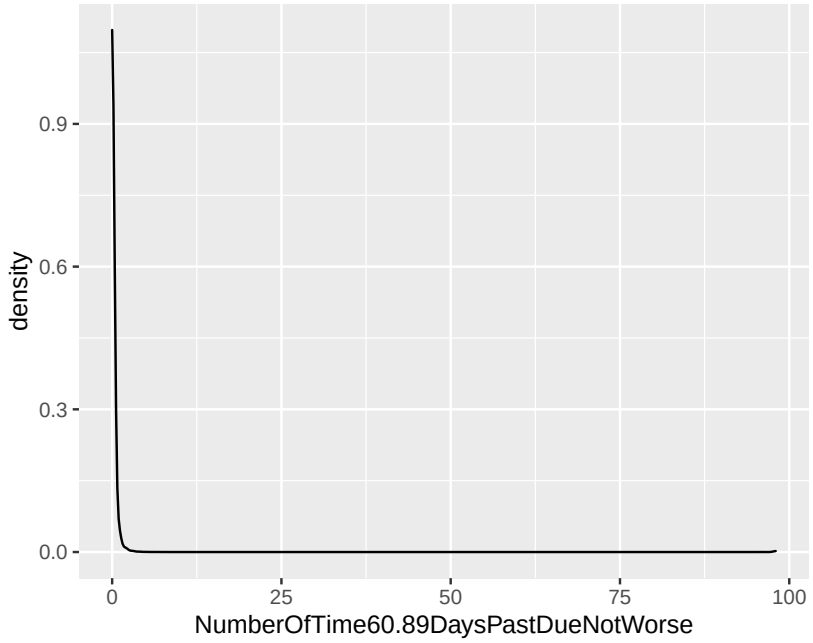
A significant majority of the borrowers had upto 2 mortgage loans. There are a few outliers with very high number of **NumberRealEstateLoansorLines**

10) **NumberOfTime60-89DaysPastDueNotWorse** - Number of times borrower has been 60-89 days past due but no worse in the last 2 years. This is a predictor variable.

```
summary(credit_data$NumberOfTime60.89DaysPastDueNotWorse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.2404  0.0000 98.0000
```

```
credit_data %>% ggplot(aes(NumberOfTime60.89DaysPastDueNotWorse)) + geom_density()
```




```
table(credit_data$NumberOfTime60.89DaysPastDueNotWorse)
```

```
##
##      0      1      2      3      4      5      6      7      8      9     11
## 142396   5731   1118    318    105     34     16      9      2      1      1
##     96     98
##      5    264
```

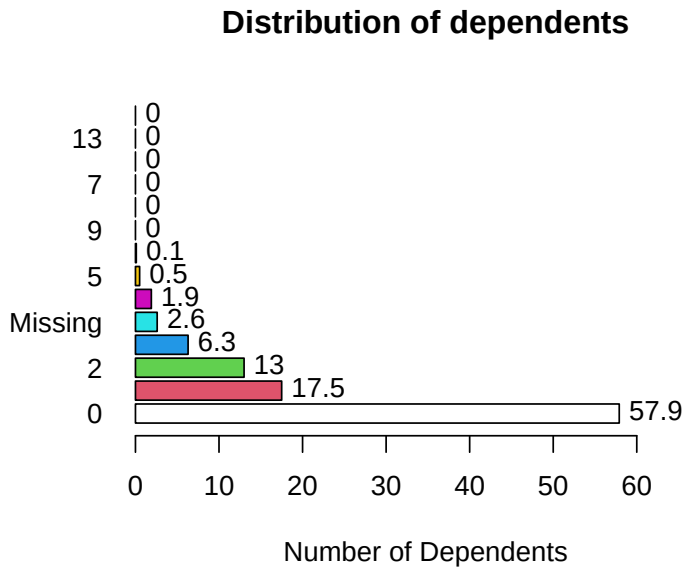
Again,a vast majority do not have any record of past due in 60 to 89 days

11) **NumberOfDependents** - Number of dependents in family excluding themselves (spouse, children etc.). This is a predictor variable.

```r
summary(credit_data$NumberOfDependents)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.000   0.000   0.757   1.000  20.000    3924
```

```r
library(epiDisplay)
tab1(credit_data$NumberOfDependents, sort.group = "decreasing", cum.percent = FALSE,
     bar.values = "percent", main = "Distribution of dependents",
     xlab = "Number of Dependents",
     ylab = "count",horiz = TRUE)
```




```
## credit_data$NumberOfDependents :
##         Frequency   %(NA+)   %(NA-)
## 0           86902     57.9     59.5
## 1           26316     17.5     18.0
## 2           19522     13.0     13.4
## 3            9483      6.3      6.5
## <NA>         3924      2.6      0.0
## 4            2862      1.9      2.0
## 5             746      0.5      0.5
## 6             158      0.1      0.1
## 7              51      0.0      0.0
```

```
## 8              24      0.0      0.0
## 9               5      0.0      0.0
## 10              5      0.0      0.0
## 13              1      0.0      0.0
## 20              1      0.0      0.0
##   Total    150000    100.0    100.0
```

A significant number i.e. 57.9% do not have any dependents. Also, important to note that about 2.6% values are missing in this predictor variable.

Data Validation

After having explored and visualized the data, the next step would be to check for missing values and data ranges to identify outliers. Then, they would have to be treated with either imputation or elimination to complete the data-set.

Missing Values Identify number of missing values in data-set

```
sum(is.na(credit_data))
```

```
## [1] 33655
```

There are 33655 missing values in the train data-set. Identify, how significant as a proportion of the data-set are the missing values.

```
sum(is.na(credit_data))/(nrow(credit_data)*ncol(credit_data))
```

```
## [1] 0.02039697
```

About 2% of the data are missing from the data-set

Identify number of missing values in data-set

```
sapply(credit_data, function(x) sum(is.na(x)))
```

```
##                     SeriousDlqin2yrs RevolvingUtilizationOfUnsecuredLines
##                                    0                                     0
##                                  age NumberOfTime30.59DaysPastDueNotWorse
```

```
##                                   0                                   0
##                           DebtRatio                       MonthlyIncome
##                                   0                               29731
##     NumberOfOpenCreditLinesAndLoans             NumberOfTimes90DaysLate
##                                   0                                   0
##        NumberRealEstateLoansOrLines NumberOfTime60.89DaysPastDueNotWorse
##                                   0                                   0
##                  NumberOfDependents
##                                3924
```

Only **MonthlyIncome** and **NumberOfDependents** have missing values.

Treatment of Missing Values Proportion of missing values for "MonthlyIncome" explanatory variable.

```
29731/150000
```

```
## [1] 0.1982067
```

About 20% of the "MonthlyIncome" variable has missing values.

```
summary(credit_data$MonthlyIncome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0    3400    5400    6670    8249 3008750   29731
```

Following are critical observations for determining imputation method for missing values of **MonthlyIncome** variable

- 75% of the values are 8249 or below
- Max value is very high - an oulier
- Imputing with mean value distorts the MonthlyIncome of the data set

Hence, the preferred approach for **MonthlyIncome** variable is Median value imputations for missing values.

Replace missing values with median values in **MonthlyIncome** explanatory variable

```
credit_data$MonthlyIncome[is.na
(credit_data$MonthlyIncome)]<-
median(credit_data$MonthlyIncome,na.rm = TRUE)
```

Check if the missing values have been imputed

```
sum(is.na(credit_data$MonthlyIncome))
```

```
## [1] 0
```

Missing values in **MonthlyIncome** explanatory variable have been imputed successfully.

Now, imputation of missing values of **NumberOfDependents** explanatory variable.

```
summary(credit_data$NumberOfDependents)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.000   0.000   0.757   1.000  20.000    3924
```

- There are 3924 missing values in **NumberofDependents** predictor variable.
- Mean value cannot be used for imputation. Needs to be a integer.

Identify unique values in **NumberOfDependents** explanatory variable

```
unique(credit_data$NumberOfDependents)
```

```
##  [1]  2  1  0 NA  3  4  5  6  8  7 20 10  9 13
```

Values range from 0 to 20.

Establish the frequency of each of these values

```r
table(credit_data$NumberOfDependents)
```

```
##
##     0     1     2     3     4     5     6     7     8     9    10    13    20
## 86902 26316 19522  9483  2862   746   158    51    24     5     5     1     1
```

Vast majority of values are 0.

```r
sum(is.na(credit_data$NumberOfDependents))/nrow(credit_data)
```

```
## [1] 0.02616
```

Only 2% of the values are missing in **NumberOfDependents** variable

% of **NumberOfDependents** variable values are 0 is

```r
86902/150000
```

```
## [1] 0.5793467
```

About 58% of value in this variable are 0.

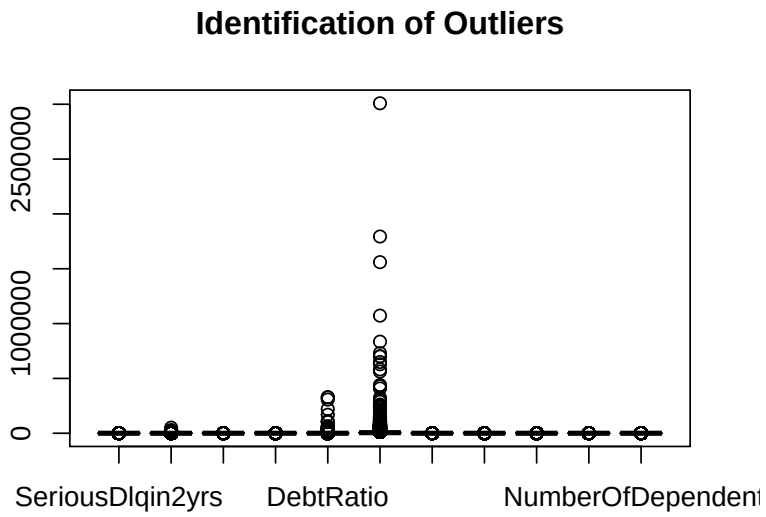
Impute missing values with 0

```r
credit_data$NumberOfDependents[is.na(credit_data$NumberOfDependents)] <- 0
sum(is.na(credit_data$NumberOfDependents))
```

```
## [1] 0
```

Outlier Analysis Outliers in the data-set can be viewed with *boxplot* function

```r
boxplot(credit_data, main = "Identification of Outliers")
```



MonthlyIncome & **DebtRatio** clearly have outliers. During data visualization other outliers like **RevolvingUtilizationOfUnsecuredLines, NumberOfOpenCreditLinesAndLoans** and **NumberRealEstateLoansOrLines** were also identified. Outliers can be identified and replaced using percentile method as well.

Outlier replacement for **RevolvingUtilizationOfUnsecuredLines**

```
# Outlier replacement for **RevolvingUtilizationOfUnsecuredLines**
quantile(credit_data$RevolvingUtilizationOfUnsecuredLines, c(.1,.5,.8,.9,.98,.99,1))
```

```
##          10%          50%          80%          90%          98%          99%
## 2.968979e-03 1.541807e-01 6.988571e-01 9.812777e-01 1.006199e+00 1.092956e+00
##         100%
## 5.070800e+04
```

```
# Replace all values above 99th percentile with the value of 99th percentile.
credit_data$RevolvingUtilizationOfUnsecuredLines[
  credit_data$RevolvingUtilizationOfUnsecuredLines >
+ quantile(credit_data$RevolvingUtilizationOfUnsecuredLines,c(.99))] <-
+ quantile(credit_data$RevolvingUtilizationOfUnsecuredLines,c(.99))
summary(credit_data$RevolvingUtilizationOfUnsecuredLines)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.02987 0.15418 0.32050 0.55905 1.09296
```

Outlier replacement for **DebtRatio**

```
quantile(credit_data$DebtRatio, c(.1,.5,.8,.9,.98,.99,1))
```

```
##          10%          50%          80%          90%          98%          99%
## 3.087398e-02 3.665078e-01 4.000000e+00 1.267000e+03 3.839000e+03 4.979040e+03
##         100%
## 3.296640e+05
```

```
# Replace all values above 80th percentile with the value of 80th percentile.
credit_data$DebtRatio[credit_data$DebtRatio >
  + quantile(credit_data$DebtRatio,c(.8))] <-
  + quantile(credit_data$DebtRatio,c(.8))
summary(credit_data$DebtRatio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1751  0.3665  1.0881  0.8683  4.0000
```

Outlier replacement for **MonthlyIncome**

```
quantile(credit_data$MonthlyIncome, c(.1,.5,.8,.9,.98,.99,.999,1))
```

```
##        10%        50%        80%        90%        98%        99%      99.9%
##    2325.00    5400.00    8250.00   10750.00   18000.00   23000.00   72759.16
##       100%
## 3008750.00
```

```
# Replace all values above 99th percentile with the value of 99th percentile.
credit_data$MonthlyIncome[credit_data$MonthlyIncome >
  + quantile(credit_data$MonthlyIncome,c(.99))] <-
  + quantile(credit_data$MonthlyIncome,c(.99))
summary(credit_data$MonthlyIncome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    3903    5400    6142    7400   23000
```

Outlier replacement for **NumberOfOpenCreditLinesAndLoans**

```
quantile(credit_data$NumberOfOpenCreditLinesAndLoans, c(.1,.5,.8,.9,.98,.99,1))
```

```
##  10%  50%  80%  90%  98%  99% 100%
##    3    8   12   15   22   24   58
```

```
# Replace all values above 99th percentile with the value of 99th percentile
credit_data$NumberOfOpenCreditLinesAndLoans[credit_data$NumberOfOpenCreditLinesAndLoans >
 + quantile(credit_data$NumberOfOpenCreditLinesAndLoans, c(.99))] <-
 + quantile(credit_data$NumberOfOpenCreditLinesAndLoans, c(.99))
summary(credit_data$NumberOfOpenCreditLinesAndLoans)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.000   8.000   8.404  11.000  24.000
```

Outlier replacement for **NumberRealEstateLoansOrLines**

```
quantile(credit_data$NumberRealEstateLoansOrLines,c(.1,.5,.8,.9,.98,.99,.999,1))
```

```
##   10%   50%   80%   90%   98%   99% 99.9%  100%
##     0     1     2     2     4     4     9    54
```

```
# Replace all values above 99th percentile with the value of 99th percentile
credit_data$NumberRealEstateLoansOrLines[credit_data$NumberRealEstateLoansOrLines >
  + quantile(credit_data$NumberRealEstateLoansOrLines, c(.99))] <-
  + quantile(credit_data$NumberRealEstateLoansOrLines, c(.99))
summary(credit_data$NumberRealEstateLoansOrLines)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.9926  2.0000  4.0000
```

Outcome variable into categorical The outcome variable is now converted into categorical variable

```
# Convert outcome variable to categorical
credit_data$SeriousDlqin2yrs <- as.factor(credit_data$SeriousDlqin2yrs)
class(credit_data$SeriousDlqin2yrs)
```

```
## [1] "factor"
```

Numerical features normalization Before starting modeling process, we need to apply on all numeric variables z-score normalization (from each feature value we subtract mean and the result is divided by standard deviation). We will use function **scale** for this purpose.

```
# z-score normalisation of numeric variables
credit_data %>% mutate_if(is.numeric, scale)
```

The transformed data is not printed in this report, given the size of the data-set. We use results='hide' in code chunk to hide the output while the code still runs

Building Models

Data-set will be partitioned into 70% training & 30% validation using **caret** package and **createDataPartition** function. The idea behind have 70/30 split between training and validation set is that, for size of this data-set, at 70% for training set it makes the classification model better and at 30% for test it makes the error estimates more accurate.

```
# Set the seed
set.seed(1, sample.kind = "Rounding")
# Partition the data & store it in index_test
library(caret)
index_test <- createDataPartition(y = credit_data$SeriousDlqin2yrs,
                                  times = 1, p = 0.7, list = FALSE)
```

Create training and validation data sets

```
# Create train set
train_models <- credit_data[index_test, ]
```

There are 105001 observations in the `train_models` data-set.

```r
# Create validation set
validate_models <- credit_data[-index_test,]
```

There are 44999 observations in the `validate_models` set.

Now, let's compare both the training and validation data sets. For this, we use the package **dataCompareR** and the function **rCompare**.

```r
# comparison of both training and validation datasets
library(dataCompareR)
comparison_train_val <- rCompare(train_models, validate_models)
comp_summ <- summary(comparison_train_val)
comp_summ[c("datasetSummary", "ncolInAOnly", "ncolInBOnly", "ncolCommon",
            "rowsInAOnly", "rowsInBOnly", "nrowCommon")]
```

```
## $datasetSummary
##      Dataset Name Number of Rows Number of Columns
## 1    train_models         105001                11
## 2 validate_models          44999                11
##
## $ncolInAOnly
## [1] 0
##
## $ncolInBOnly
## [1] 0
##
## $ncolCommon
## [1] 11
##
## $rowsInAOnly
##   indices_removed
## 1           77661
## 2           96771
## 3           83260
## 4           81958
## 5           64700
##
## $rowsInBOnly
## [1] indices_removed
## <0 rows> (or 0-length row.names)
##
## $nrowCommon
## [1] 44999
```

Logistics Regression Model

The first prediction model we would use is the Logistics Regression Model in the Generalized Linear Model. We use the **glm** function and the outcome variable is **SeriousDlqin2yrs**, a categorical variable.

```
# Model fitting
glm_model <- glm(SeriousDlqin2yrs ~ ., family = 'binomial', data = train_models)
# Get the Summary
summary(glm_model)
```

```
##
## Call:
## glm(formula = SeriousDlqin2yrs ~ ., family = "binomial", data = train_models)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4930  -0.3562  -0.2280  -0.1759   3.8844
##
## Coefficients:
##                                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          -3.406e+00  6.502e-02 -52.386  < 2e-16 ***
## RevolvingUtilizationOfUnsecuredLines  2.507e+00  3.935e-02  63.708  < 2e-16 ***
## age                                  -1.555e-02  1.062e-03 -14.645  < 2e-16 ***
## NumberOfTime30.59DaysPastDueNotWorse  3.199e-01  1.350e-02  23.700  < 2e-16 ***
## DebtRatio                            -6.343e-03  9.830e-03  -0.645  0.51877
## MonthlyIncome                        -4.734e-05  4.294e-06 -11.026  < 2e-16 ***
## NumberOfOpenCreditLinesAndLoans       4.077e-02  3.118e-03  13.073  < 2e-16 ***
## NumberOfTimes90DaysLate               3.115e-01  1.730e-02  18.011  < 2e-16 ***
## NumberRealEstateLoansOrLines          4.731e-02  1.599e-02   2.959  0.00309 **
## NumberOfTime60.89DaysPastDueNotWorse -6.078e-01  2.074e-02 -29.308  < 2e-16 ***
## NumberOfDependents                    8.837e-02  1.130e-02   7.817 5.39e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 51536  on 105000  degrees of freedom
## Residual deviance: 42633  on 104990  degrees of freedom
## AIC: 42655
##
## Number of Fisher Scoring iterations: 6
```

It is important to note that almost all features are statistically significant.

Now, let's get the prediction using **predict** function

```r
# Prediction
pred_logit <- predict(glm_model, newdata = validate_models, type = "response")
```

The probabilities in **pred_logit** need to be converted into predictions. For this we use cut-off of 0.5. Then, we evaluate the accuracy by using the **confusionMatrix** function in **caret** package.

```r
y_hat_glm <- factor(ifelse(pred_logit > 0.5, 1, 0))
# Confusion Matrix
library(caret)
confusionMatrix(y_hat_glm, reference = validate_models$SeriousDlqin2yrs,
                positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 41868  2833
##          1   124   174
##
##                Accuracy : 0.9343
##                  95% CI : (0.932, 0.9366)
##     No Information Rate : 0.9332
##     P-Value [Acc > NIR] : 0.1751
##
##                   Kappa : 0.0944
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.057865
##             Specificity : 0.997047
##          Pos Pred Value : 0.583893
##          Neg Pred Value : 0.936623
##              Prevalence : 0.066824
##          Detection Rate : 0.003867
##    Detection Prevalence : 0.006622
##       Balanced Accuracy : 0.527456
##
##        'Positive' Class : 1
##
```

Now, lets print only the overall accuracy of glm model

```r
# Accuracy of GLM model
confusionMatrix(y_hat_glm, reference = validate_models$SeriousDlqin2yrs,
                positive = "1")$overall["Accuracy"]
```

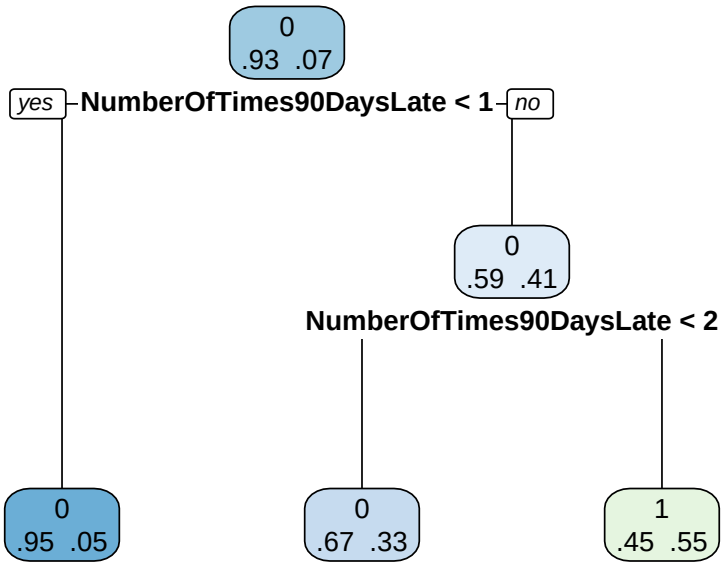
```
##  Accuracy
## 0.9342874
```

Accuracy of the model is a very good at 93%. Specificity is at 99%. However, the sensitivity is at only 5%. The NPV is 93% and PPV 58%. Also, the cut-off could be tuned to reflect the actual event rate instead of 0.5. As highlighted earlier, a vast majority of the features are statistically significant.

Classification Tree Model

We now build the classification tree model by using the **rpart** function from **rpart** library.

```r
# Decision tree
library(rpart)
# Build the model
dec_tree <- rpart(SeriousDlqin2yrs ~ . , method='class', data=train_models)
# Plot the decision tree
library(rpart.plot)
rpart.plot(dec_tree, uniform = TRUE, extra = 4)
```



Let's now make the prediction using **predict** function.

```
# Predictions
pred_tree <- predict(dec_tree,newdata = validate_models, type = 'class')
```

Now, let's evaluate the output of the Decision Tree model using **confusionMatrix** function using **caret** package.

```
# Confusion Matrix
confusionMatrix(pred_tree,
                reference=validate_models$SeriousDlqin2yrs,
                positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 41582  2488
##          1   410   519
##
##                Accuracy : 0.9356
##                  95% CI : (0.9333, 0.9378)
##     No Information Rate : 0.9332
```

```
##     P-Value [Acc > NIR] : 0.01984
##
##                   Kappa : 0.2397
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.17260
##             Specificity : 0.99024
##          Pos Pred Value : 0.55867
##          Neg Pred Value : 0.94354
##              Prevalence : 0.06682
##          Detection Rate : 0.01153
##    Detection Prevalence : 0.02064
##       Balanced Accuracy : 0.58142
##
##        'Positive' Class : 1
##
```

Let's view only the accuracy of the Decision Tree model

```
# Accuracy of Decision Tree Model
confusionMatrix(pred_tree, reference = validate_models$SeriousDlqin2yrs,
                positive = "1")$overall["Accuracy"]
```

```
##  Accuracy
## 0.9355986
```

Accuracy of the Decision Tree Model is a very good 93%. The specificity is a very good 99% and sensitivity is 17%. NPV is at 94% and PPV is at 55%

Receiver Operating Characteristic (ROC) curve model

First, we build a Logistics Regression Model and do the model performance evaluation for the same using **ROCR**.

```
library(nnet)
library(tidyverse)
# Model fitting using training dataset
lr_model <- multinom(SeriousDlqin2yrs~.,train_models)
```

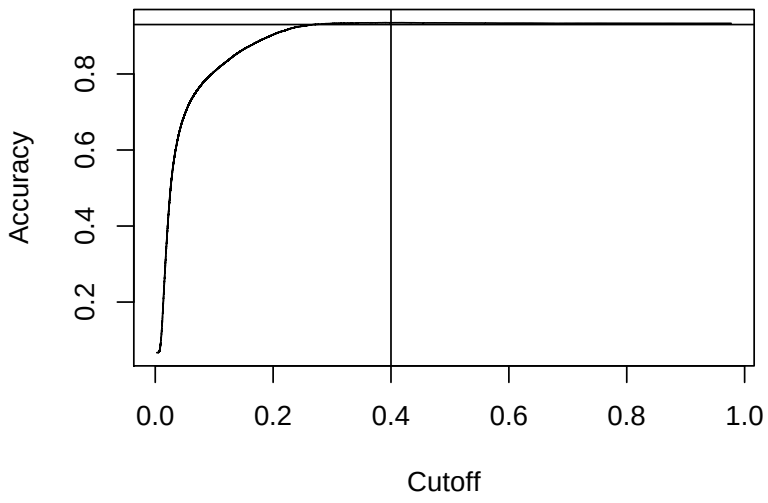
```
## # weights:  12 (11 variable)
## initial  value 72781.147106
## iter  10 value 23377.590041
```

```
## iter  20 value 21316.327247
## iter  20 value 21316.327194
## iter  20 value 21316.327194
## final  value 21316.327194
## converged
```

```
# Model prediction using validation data set
p <- predict(lr_model,validate_models)
```

The model performance is evaluated using **ROCR** package.

```
library(ROCR)
pred <- predict(lr_model,newdata=validate_models, type = 'prob')
pred <- prediction(pred,validate_models$SeriousDlqin2yrs)
eval <- performance(pred, "acc")
plot(eval)
# Identification of cut-off based on eye estimate
abline(h=0.93, v=0.40)
```




```
eval
```

```
## A performance instance
##   'Cutoff' vs. 'Accuracy' (alpha: 'none')
##   with 44698 data points
```

Above is an eye estimate of the cut-off.

An advantage with ROC curve is that it gives performance of the model at various cut-offs.

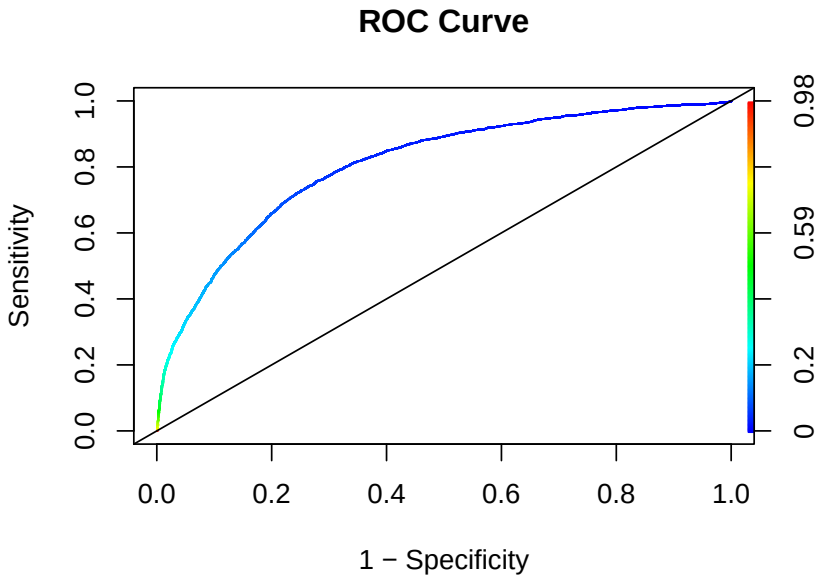
```
# Identifying best cut-off and accuracy
max <- which.max(slot(eval, "y.values")[[1]])
acc <- slot(eval, "y.values")[[1]][max]
cut <- slot(eval, "x.values")[[1]][max]
print(c(Accuracy = acc, Cutoff = cut))
```

```
##      Accuracy Cutoff.131289
##     0.9348600     0.3954938
```

Accuracy value is 0.93486 and cut off is at 0.3954938. This is not far from our earlier visual estimate.

Now, let us visualize the curve performance at different cut-off values.

```
# ROC curve gives performance at different cut off values
roc <- performance(pred, "tpr", "fpr")
plot(roc, colorize = T, main = "ROC Curve", ylab = "Sensitivity", xlab = "1 - Specificity")
abline(a=0, b=1)
```



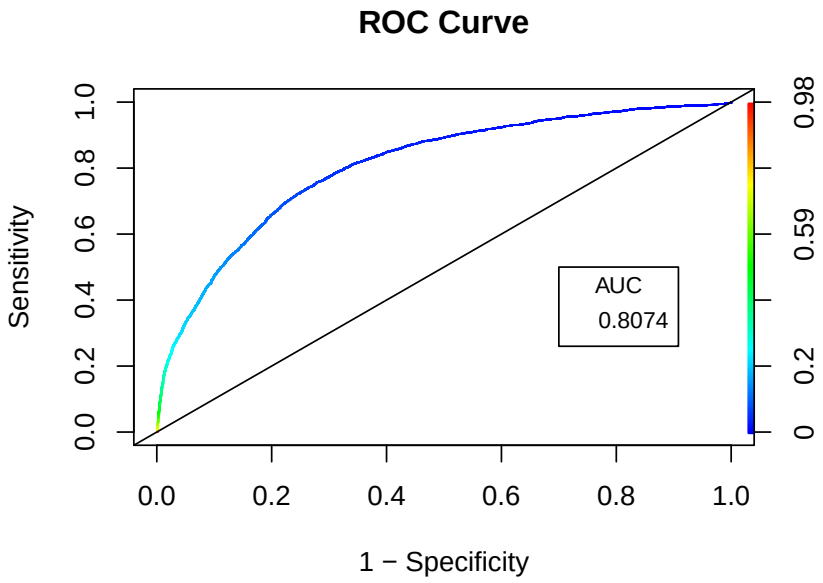
An important aspect to note here is that, optically it is clear that the performance is much better than at a cut-off of 0.5.

Now, let's estimate the Area Under the Curve(AUC) for the model.

```
# Area under the curve
auc <- performance(pred, "auc")
auc <- unlist(slot(auc, "y.values"))
auc <- round(auc, 4)
auc
```

```
## [1] 0.8074
```

```
plot(roc, colorize = T, main = "ROC Curve", ylab = "Sensitivity", xlab = "1 - Specificity")
abline(a=0, b=1)
legend(.7,.5, auc, title = "AUC", cex = 0.8)
```



Area under the curve is a very good 0.8074.

Competition Submission

Though this competition was closed in 2011, we generate the predicted values based on ROC Curve model.

```
# Make predictions based on the ROCR Curve model
pred_comp <- predict(lr_model, credit_data_comp,type = 'prob')
# We now, write the prediction into csv format as required for competition submission
write.csv(pred_comp, "submission.csv")
```

We are, however, unable to validate our predictions in competition submission, since, we are not privy to actual outcome variable.

Results Summary

- **Logistics Regression Model** - Accuracy of the model is a very good at 93%. Specificity is at 99%. However, the sensitivity is at only 5%. The NPV is 93% and PPV 58%.
- **Decision Tree Model** - Accuracy of the Decision Tree Model is a very good 93%. The specificity is a very good 99% and sensitivity is 17%. NPV is at 94% and PPV is at 55%
- **ROC Curve Model** - Accuracy value is 0.93486 and cut off is at 0.3954938. Area under the curve is also a very good 0.8074.

Conclusion

Accuracy levels have been high in all models. It is essential to predict delinquencies in 2 years more accurately than no delinquencies. Both Logistic Regression and Decision Tree approach despite high level of accuracy might fall short of predicting delinquency in 2 years in comparison to ROC Curve approach.

Some of the reasons attributed for that could be:

- Almost all features were statistically significant in Logistic Regression model. The process could be further enhanced by filtering variables, if required. For this we need to determine *goodness of fit* or *R-squared* of the model.
- Possible interactions of the predictor features with the outcome variable.
- Possible cross-interactions of predictor features.

- A different cut-off similar to that of the actual event rate rather than 0.5 could be used.

With an AUC of 0.8074 ROC Curve model does present a very good case for the prediction model.

The intent and purpose behind this project is to showcase the skill and knowledge gained during learning the course.

It has to be mentioned, however, that the ROC Curve model could be further improvised by increasing the features by the following but not limited to:

- Further transformations like log(features) or feature interactions(additive, multiplicative)
- Adding features like account history, years of employment,family status, residency status etc.

This algorithm based approach is important for banks to identify potential defaulters at an early stage. Iterative improvisations on prediction models, which cater to dynamic changes evolving in the economy/society would go a long way in making the banks more safe than sorry!

References

- Wikipedia, https://en.wikipedia.org/wiki/Credit_risk
- Introduction to Data Science,https://rafalab.github.io/dsbook/
- Writing R Scripts, http://environmentalcomputing.net/good-practice-for-writing-scripts/
- Understanding AUC-ROC Curve, https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
- Chunk Options, https://yihui.org/knitr/options/
- MiKTex, https://miktex.org/about
- RMarkdown, https://cran.r-project.org/web/packages/stationery/vignettes/Rmarkdown.pdf
- Significant variables & R Squared, https://statisticsbyjim.com/regression/low-r-squared-regression/
- Econometric Analysis, William H. Greene, 8th Edition