Capstone Project – Car Insurance Claims

Final Report

- By Samrat Mallik

Table of Contents:

1.Introduction

a) Defining Problem Statement:

We have a data from an insurance company (Name not disclosed) which is losing revenue in fraudulent claims. Based upon the given data we are required to build the below models:

- Predict if the DRV_CLAIM_STATUS will be 'Rejected' or 'Accepted'. Consider 'Closed' status given in data as 'Accepted' status.
- Identify clusters/associations in the claims data to classify them based on the severity of the claims. Identify fields that are classifying the claims as highly severe claims. These will be the claims that will need more focus and scrutiny from the fraud management team.

b) Need of the Study/Project:

Huge losses have prompted insurance companies to set-up a new anti-fraud department whose job is to identify the risk and loss to these frauds and to find out ways to reduce fraudulent claims. This prediction will help the Claims Management team to manage the claims effectively. Prediction model will give them the list of claims to be rejected in advance thus helping them to scrutinize these claims and ensure no leakage in for of frauds.

c) Understanding business/social opportunity:

India is a huge market for insurance but the industry is bleeding losses due to increasing fraud from inside and outside of the system. Insurance frauds leads to close to INR 40,000 crores loss which is close to 8.5% revenue of the insurance sector in India. Thus, this study aims to understand what precautions these compani4es might take in order to mitigate losses from fraudulent claims.

2.Data Report

a)Understanding how data was collected in terms of time, frequency and methodology:

Data contains records of claims from the insurance company's database starting from 1998-99 and ending at 2012-13. Each claim has an unique id.

b) Visual inspection of data

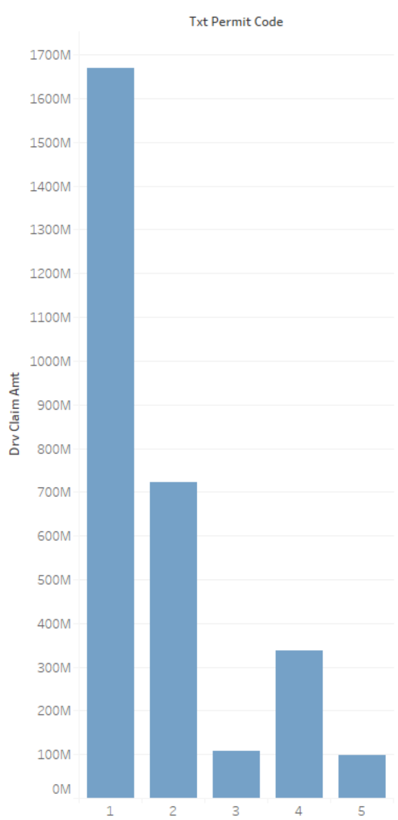
The dataset has 75200 rows and 32 columns. Out of 75200 observations 71324 are regular claims and 3876 observations are those of fraudulent claims.

c) Understanding of attributes

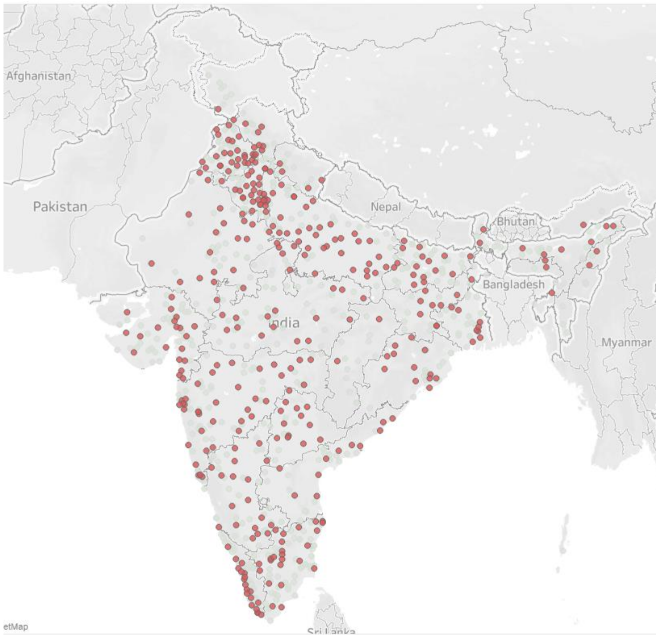
Out of the 32 variables, 4 are numeric, 3 are dates and the rest are factors. Our dependent variable "DRV_CLAIM_STATUS" is a factor having 2 levels namely "CLOSED" and "REJECTED". We may consider "DRV_CLAIM_AMT" as a numeric counterpart to the dependent variable. We also have geographical data in some of the variables.

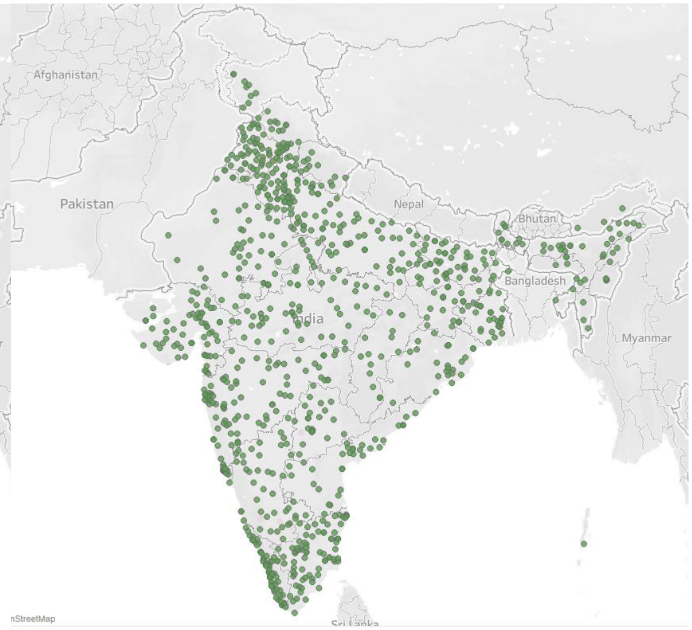
3.Exploratory Data Analysis

a)Univariate Analysis:



We see that vehicles having permit code "1" have the highest number and amount of claims. Using the reference dictionary provided to us, we come to know that 1 is attributed to vehicles having local permit. Followed by state and then national permit vehicles.

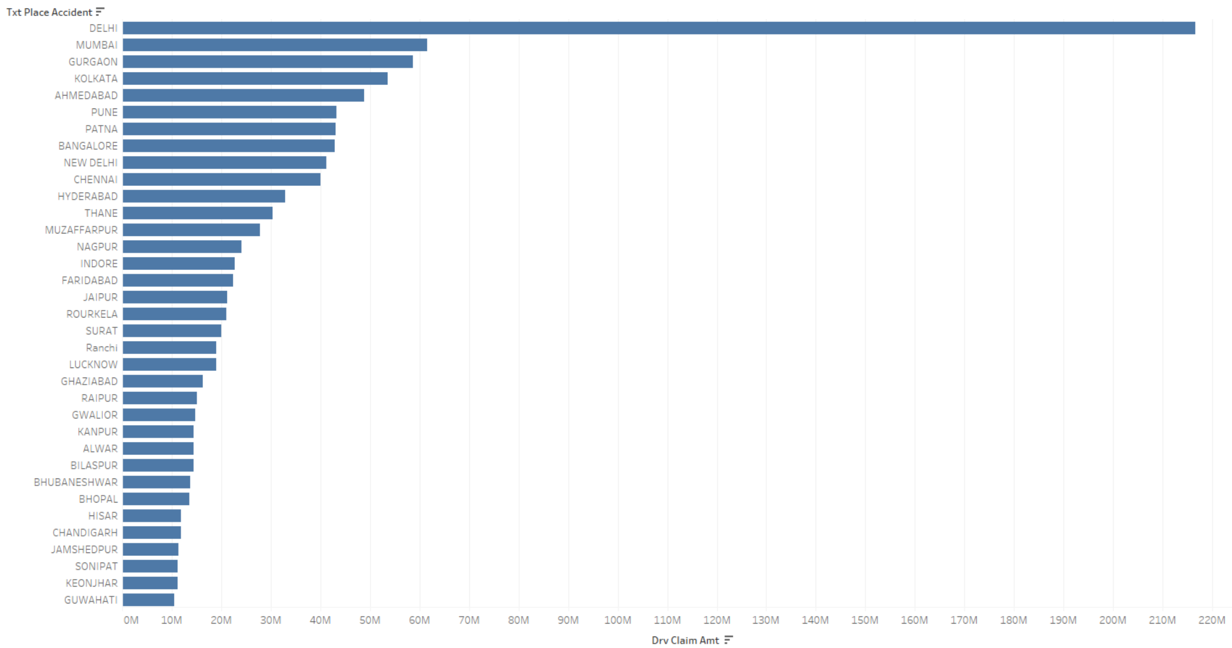




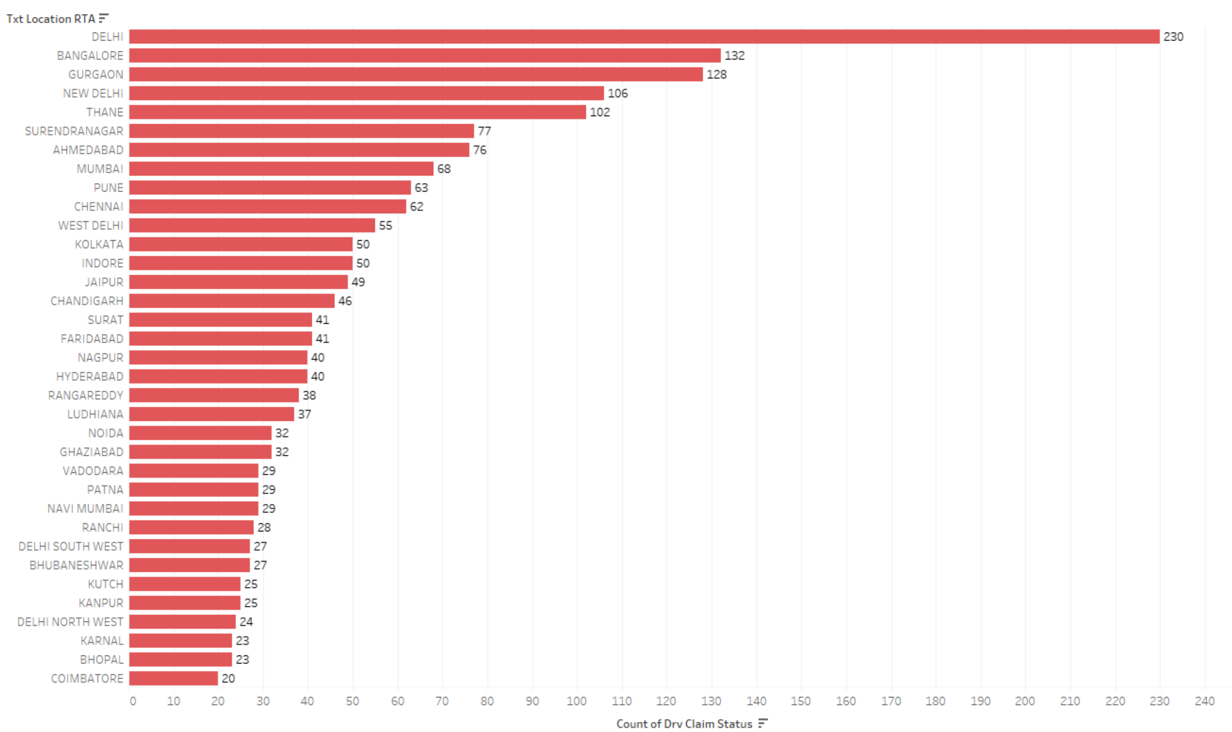
Using the geographical data provided to us we have managed to plot claims based on fraud on a map of the country. Red denotes rejected status or fraudulent claims while green denotes genuine claims. We can see how these two categories are distributed on the maps. In both cases we notice highest density in the northern and southern parts of the country.

	Drv Claim Status	
Txt Class Co..	CLOSED	REJECTED
11	1,397,570,394	1,962
13	1,600,123	
14	235,815,155	356
17	810,057,730	505
18	12,725,647	0
19	4,847,536	1
20	237,218	
21	210,223,117	239
22	9,935,439	0
23	248,740,831	8

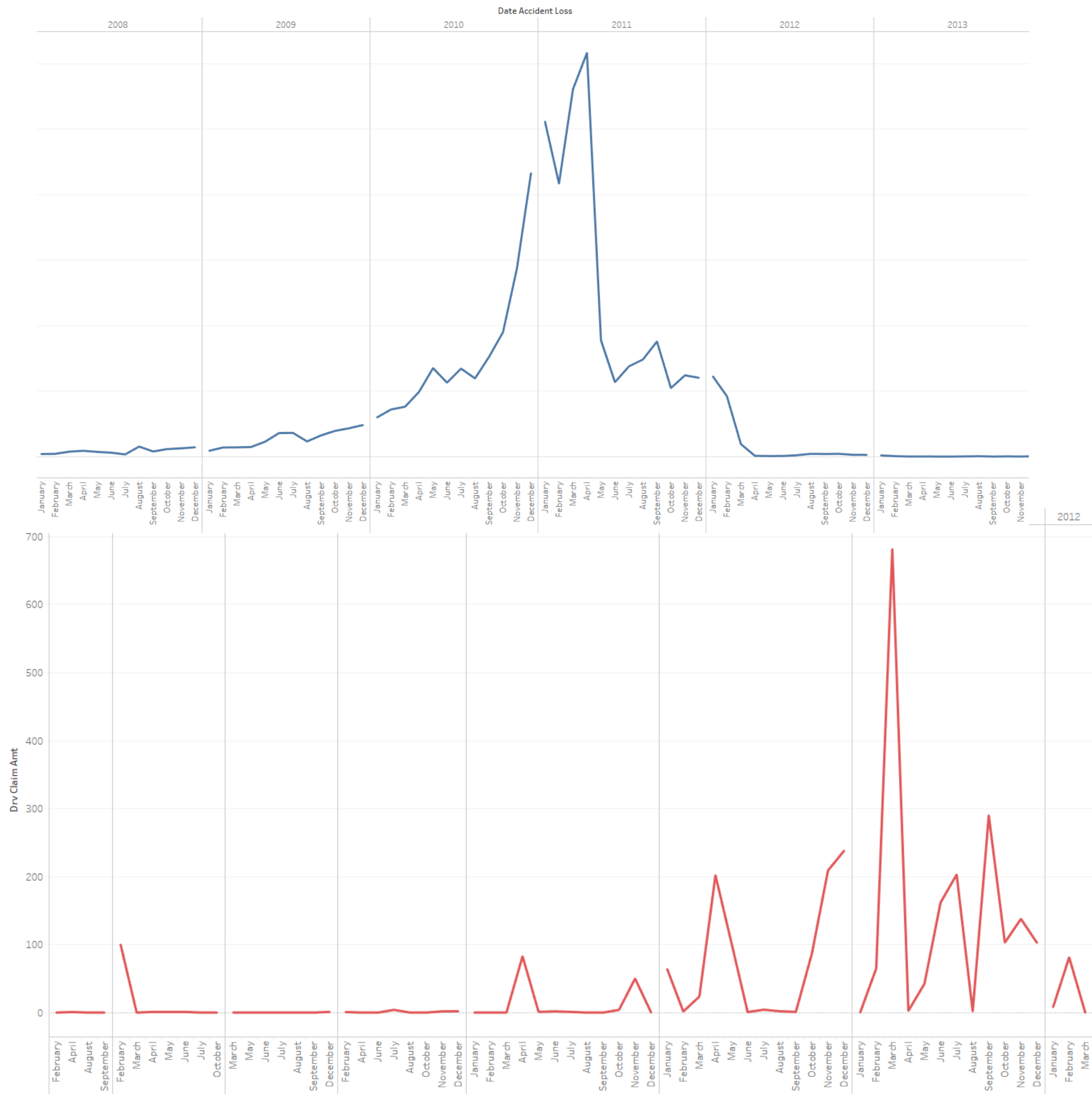
We next try to determine which category of vehicles have highest claim amount and highest fraudulent claims. Class "11" or private cars(four-wheelers) have the highest number and amount in terms of both genuine and fraudulent claims. Next are public goods(17) carrying vehicles.



In the bar chart above we can see that Delhi has the largest amount of overall claims. Followed by Mumbai, Gurgaon and Kolkata.



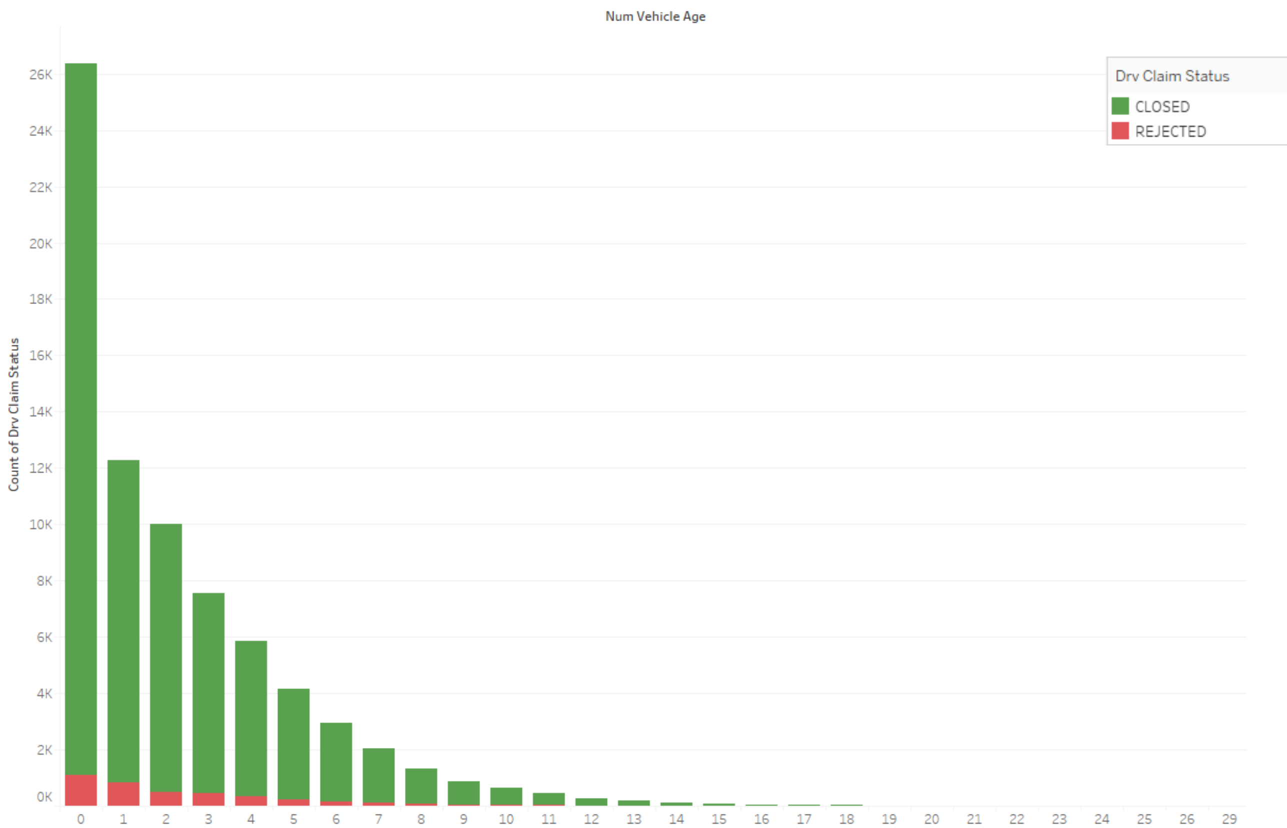
Whereas, when we plot the countries against fraudulent claims, the order slightly changes. Delhi is still at the top, now followed by Bangalore, Gurgaon, New Delhi, Thane,etc.



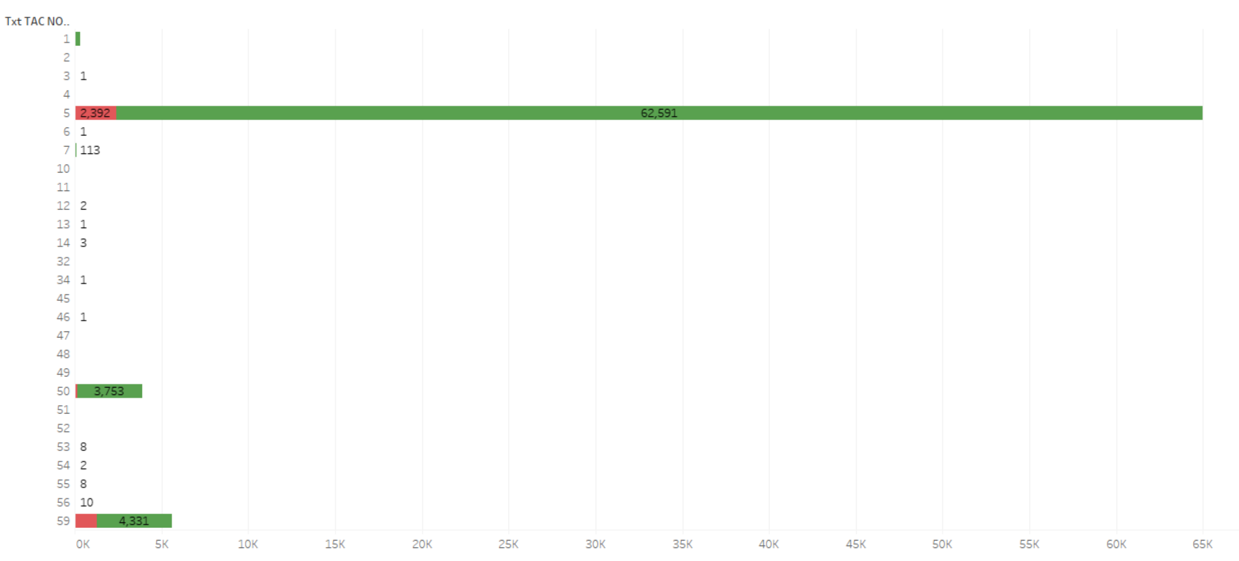
Plotting a time-series graph using the date of accident or loss, we can visualize how the claim amount has fluctuated over time. We notice that the claims have risen considerably in 2010 to 2011 with the highest amount of claims being in the month of March 2011. In the second graph we see that the number of rejections are also highest in the month of March 2011, but there are several undulations in the curve.

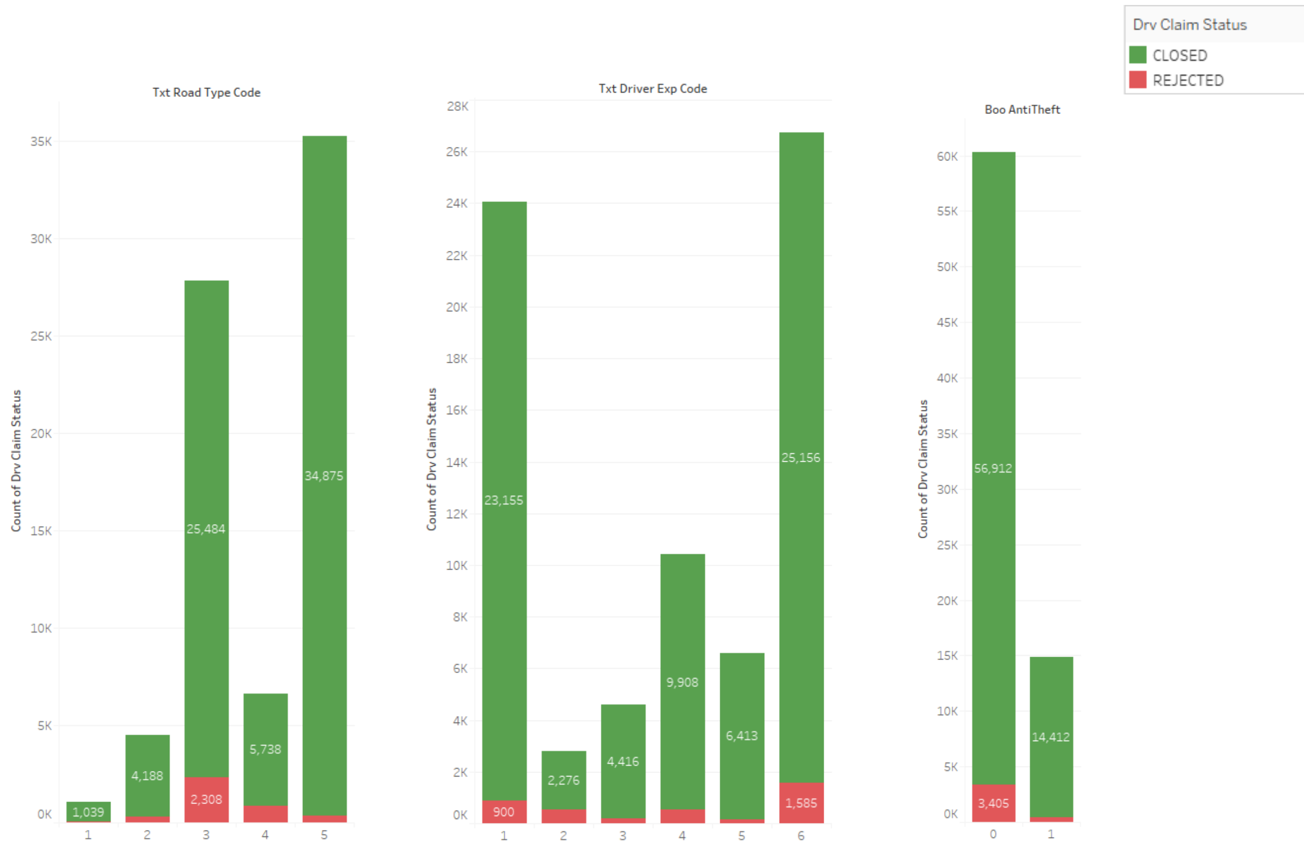
b)Bivariate Analysis:

The following graph suggests that most of the claims are for cars less than five years old. It is steep curve showing that younger the car higher the claims.

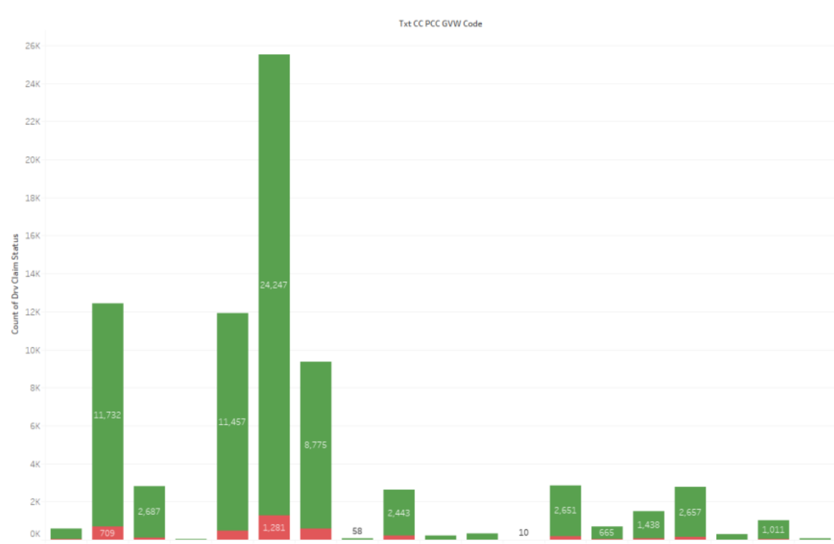


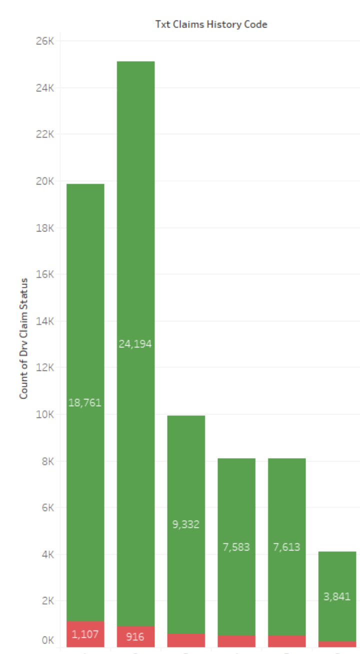
The nature of loss suggests that “5” or Accident by external means has highest claims.





These graphs tell us that drivers having 15 years or above experience, with no Anti-theft, driving private cars or taxis on City/Town Roads having no claim history have the highest number of fraudulent claims.



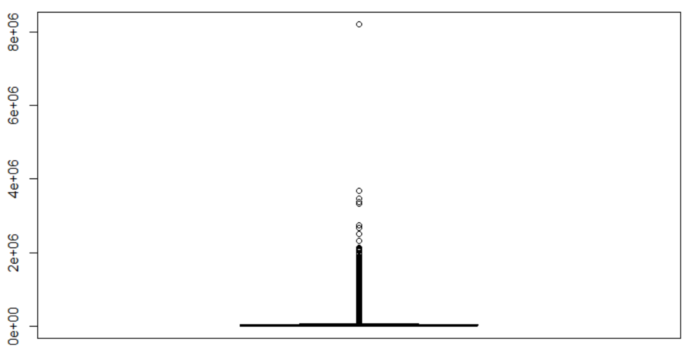


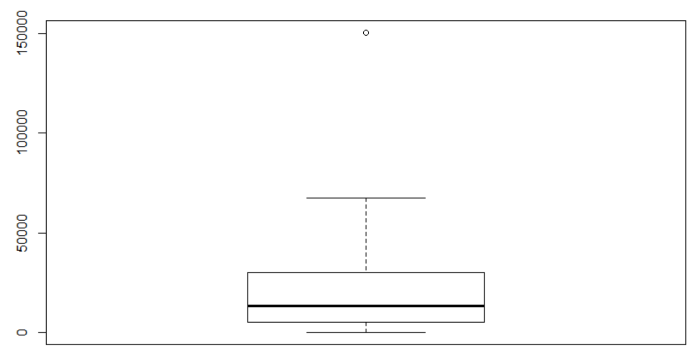
c) Missing Value Treatment:

There are 3876 missing values in the data pertaining to the empty fields in disbursement date for rejected claims. We do not need to do imputation as it is not significant for our model.

d) Outlier treatment:

DRV_CLAIM_AMT



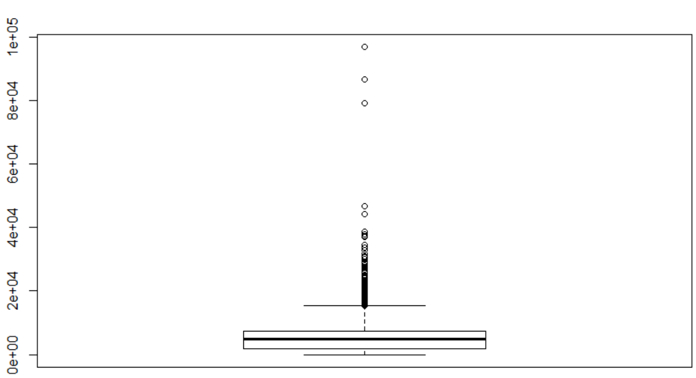


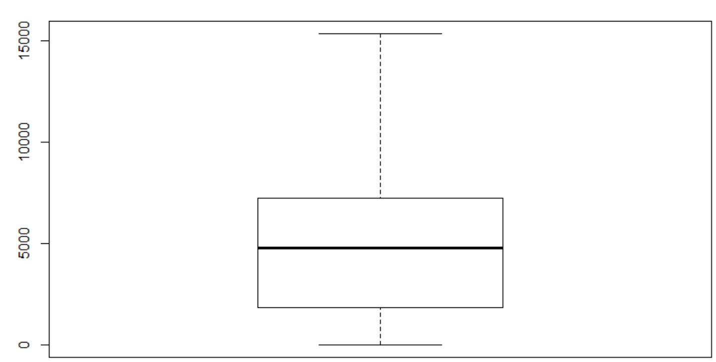
Before

After

We identify and treat the outliers in the numeric variables DRV_CLAIM_AMT, Num_Net_OD_Premium, Num_IDV respectively by capping at 5% and 95% of the data.

Num_Net_OD_Premium

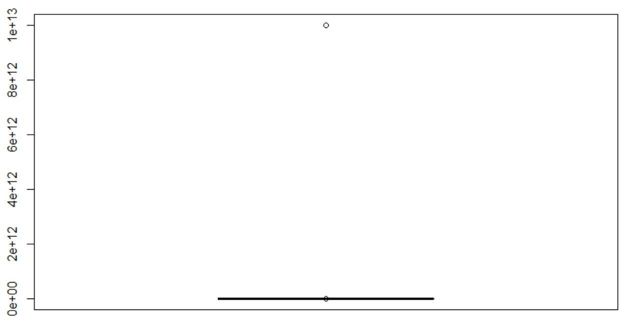


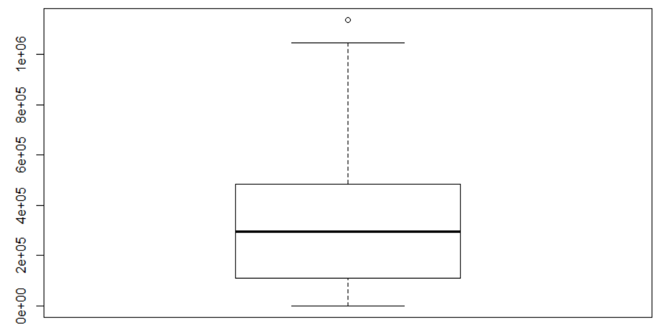


Before

After

Num_IDV





Before

After

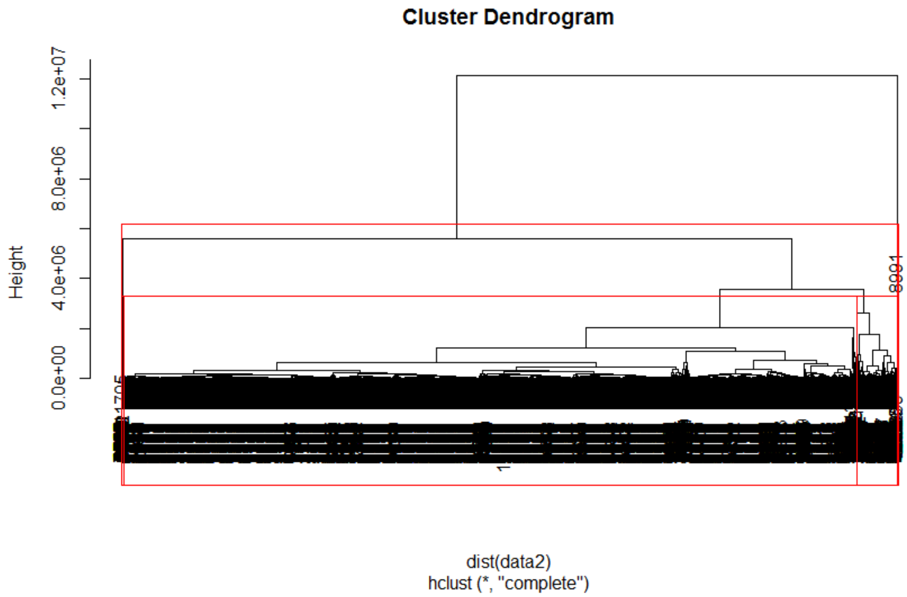
4.Insights from EDA

a) Is the data unbalanced?

Data is highly skewed and unbalanced having only 5% of the fraudulent data. Hence, we may consider SMOTE in this case.

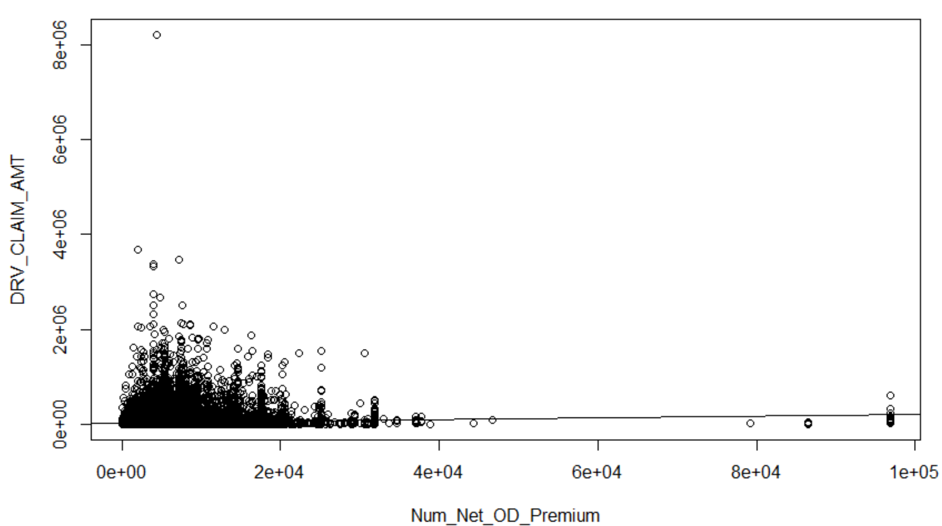
b) Clustering

We perform hierarchical clustering on a part of the data and split into 5 clusters as shown below:



c) Additional insights

The plot below shows us the liner relationship and correlation between the two numeric variables DRV_CLAIM_AMT and Num_Net_OD_Premium.



5. Installing Necessary Packages and Invoking Libraries

We have used the following libraries for this study:

library(tidyverse)

library(cluster)

library(factoextra)

library(gridExtra)

library(ggplot2)

library(ellipse)

library(ggcorrplot)

library(RColorBrewer)

library(nFactors)

library(psych)

library(lattice)

library(caTools)

library(rpart)

library(rpart.plot)

library(rattle)

library(data.table)

library(ROCR)

library(ineq)

library(StatMeasures)

library(htmlwidgets)

library(DataExplorer)

library(corrplot)

library(partykit)

library(dplyr)

library(purrr)

library(InformationValue)

library(car)

library(ROCR)

library(MASS)

library(e1071)

library(class)

library(caret)

library(readxl)

library(DMwR)

library(ipred)

library(gbm)

library(Matrix)

library(randomForest)

library(mlr)

library(xgboost)

6. Data Preparation and Environment Setup

setwd("C:/Users/Samrat/Documents/R/Directories/Capstone Project")

data = read_excel("claims data.xlsx")

summary(data)

```
Uniquekey     Txt_Policy_Year    Boo_Endorsement    Txt_Location_RTA   Txt_Policy_Code   
Txt_Class_Code     Txt_Zone_Code     
 Min.   :    1   Length:75200       Length:75200       Length:75200       Length:75200      
Length:75200       Length:75200      
 1st Qu.:36599   Class :character   Class :character   Class :character   Class :chara
cter   Class :character   Class :character  
 Median :58131   Mode  :character   Mode  :character   Mode  :character   Mode  :chara
cter   Mode  :character   Mode  :character  
 Mean   :56217                                                                                         
 3rd Qu.:80838                                                                                         
 Max.   :99991                                                                                         

 Num_Vehicle_Age    Txt_CC_PCC_GVW_Code Txt_Colour_Vehicle    Num_IDV          Txt_Per
mit_Code    Txt_Nature_Goods_Code
 Length:75200       Length:75200        Length:75200       Min.   :1.000e+00   Length:
75200       Length:75200         
 Class :character   Class :character    Class :character   1st Qu.:1.100e+05   Class :
character   Class :character     
 Mode  :character   Mode  :character    Mode  :character   Median :2.950e+05   Mode  :
character   Mode  :character     
                                                           Mean   :1.334e+08                       
                                                           3rd Qu.:4.850e+05                       
                                                           Max.   :1.000e+13                       

 Txt_Road_Type_Code Txt_Vehicle_Driven_By_Code Txt_Driver_Exp_Code Txt_Claims_History_
Code Txt_Driver_Qualification_Code
 Length:75200       Length:75200               Length:75200        Length:75200       
Length:75200                 
 Class :character   Class :character           Class :character    Class :character   
Class :character             
 Mode  :character   Mode  :character           Mode  :character    Mode  :character   
Mode  :character             



 Txt_Incurred_Claims_Code Boo_TPPD_Statutory_Cover_only Txt_Claim_Year     Date_Accide
nt_Loss           Txt_Place_Accident
 Length:75200             Length:75200                  Length:75200       Min.   :199
9-04-05 00:00:00   Length:75200      
 Class :character         Class :character              Class :character   1st Qu.:201
1-01-26 00:00:00   Class :character  
 Mode  :character         Mode  :character              Mode  :character   Median :201
1-04-06 00:00:00   Mode  :character  
                                                                           Mean   :201
1-03-28 17:48:06                     
                                                                           3rd Qu.:201
1-08-27 00:00:00                     
                                                                           Max.   :201
3-12-02 00:00:00                     

 Date_Claim_Intimation         Txt_TAC_NOL_Code   Date_Disbursement             Boo_OD
_Total_Loss   DRV_CLAIM_AMT    
 Min.   :1999-05-17 00:00:00   Length:75200       Min.   :1999-06-26 00:00:00   Length
:75200       Min.   :      0  
 1st Qu.:2011-02-03 00:00:00   Class :character   1st Qu.:2011-04-15 00:00:00   Class 
:character   1st Qu.:   5082  
```

```
 Median :2011-04-11 00:00:00   Mode  :character   Median :2011-05-25 00:00:00   Mode
:character   Median :  13000
 Mean   :2011-04-10 04:18:15                      Mean   :2011-06-16 08:09:35
Mean   :  38986
 3rd Qu.:2011-09-07 00:00:00                      3rd Qu.:2011-10-15 00:00:00
3rd Qu.:  30000
 Max.   :2013-12-07 00:00:00                      Max.   :2014-01-06 00:00:00
Max.   :8216000
                                                  NA's   :3735
 DRV_CLAIM_STATUS   Boo_AntiTheft        Boo_NCB           Num_Net_OD_Premium
 Length:75200       Length:75200       Length:75200       Min.   :   13
 Class :character   Class :character   Class :character   1st Qu.: 1850
 Mode  :character   Mode  :character   Mode  :character   Median : 4786
                                                          Mean   : 5501
                                                          3rd Qu.: 7244
                                                          Max.   :96795
```

str(data)

```
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':	75200 obs. of  32 variables:
 $ Uniquekey                       : num  20745 20752 20757 20759 20760 ...
 $ Txt_Policy_Year                 : chr  "2011-12" "2011-12" "2011-12" "2010-11" ...
 $ Boo_Endorsement                 : chr  "0" "0" "0" "0" ...
 $ Txt_Location_RTA                : chr  "GOA" "AKOLA" "NAGPUR" "MUMBAI WEST" ...
 $ Txt_Policy_Code                 : chr  "2" "2" "2" "2" ...
 $ Txt_Class_Code                  : chr  "11" "11" "14" "14" ...
 $ Txt_Zone_Code                   : chr  "36" "36" "36" "35" ...
 $ Num_Vehicle_Age                 : chr  "7" "7" "2" "3" ...
 $ Txt_CC_PCC_GVW_Code             : chr  "50" "50" "47" "47" ...
 $ Txt_Colour_Vehicle              : chr  "OTHER COLOR" "OTHER COLOR" "OTHER COLOR" "OTHE
R COLOR" ...
 $ Num_IDV                         : num  104000 88000 29850 30000 143140 ...
 $ Txt_Permit_Code                 : chr  "1" "1" "1" "1" ...
 $ Txt_Nature_Goods_Code           : chr  "2" "2" "2" "2" ...
 $ Txt_Road_Type_Code              : chr  "3" "3" "3" "3" ...
 $ Txt_Vehicle_Driven_By_Code      : chr  "1" "1" "1" "1" ...
 $ Txt_Driver_Exp_Code             : chr  "6" "6" "1" "1" ...
 $ Txt_Claims_History_Code         : chr  "4" "2" "5" "6" ...
 $ Txt_Driver_Qualification_Code: chr  "2" "1" "2" "2" ...
 $ Txt_Incurred_Claims_Code        : chr  "2" "5" "2" "8" ...
 $ Boo_TPPD_Statutory_Cover_only: chr  "0" "0" "0" "0" ...
 $ Txt_Claim_Year                  : chr  "2012-13" "2012-13" "2012-13" "2012-13" ...
 $ Date_Accident_Loss              : POSIXct, format: "2011-10-14" "2011-10-08" "2011-10-
12" "2011-10-09" ...
 $ Txt_Place_Accident              : chr  "GOA" "AKOLA" "NAGPUR" "MUMBAI WEST" ...
 $ Date_Claim_Intimation           : POSIXct, format: "2011-10-15" "2011-10-14" "2011-10-
14" "2011-10-14" ...
 $ Txt_TAC_NOL_Code                : chr  "59" "59" "59" "59" ...
 $ Date_Disbursement               : POSIXct, format: NA NA "2011-12-13" "2011-12-13" ...
 $ Boo_OD_Total_Loss               : chr  "0" "0" "0" "0" ...
 $ DRV_CLAIM_AMT                   : num  0 0 1876 5026 0 ...
 $ DRV_CLAIM_STATUS                : chr  "REJECTED" "REJECTED" "CLOSED" "CLOSED" ...
 $ Boo_AntiTheft                   : chr  "0" "0" "0" "0" ...
 $ Boo_NCB                         : chr  "0" "0" "0" "0" ...
 $ Num_Net_OD_Premium              : num  5088 4712 330 405 2832 ...
```

sum(is.na(data))

```
3735
```

attach(data)

summary(as.factor(DRV_CLAIM_STATUS))

```
CLOSED REJECTED 
   71324     3876 
```

3876/75200

```
0.05154255
```

We can see that the “REJECTED” responses of the target variable “DRV_CLAIM_STATUS” constitutes only 5.15% of the data. Hence it is highly imbalanced.

7. Variable Transformation

We do some initial data cleaning and convert the remaining variables to numeric vectors so as to easily facilitate the regression models.

```
data2 = data[,-c(1,2,4,10,21,22,23,24,26)]

data2$Boo_Endorsement = as.numeric(data2$Boo_Endorsement)
data2$Txt_Policy_Code = as.numeric(data2$Txt_Policy_Code)
data2$Txt_Class_Code = as.numeric(data2$Txt_Class_Code)
data2$Txt_Zone_Code = as.numeric(data2$Txt_Class_Code)
data2$Num_Vehicle_Age = as.numeric(data2$Num_Vehicle_Age)
data2$Txt_CC_PCC_GVW_Code = as.numeric(data2$Txt_CC_PCC_GVW_Code)
data2$Txt_Permit_Code = as.numeric(data2$Txt_Permit_Code)
data2$Txt_Nature_Goods_Code = as.numeric(data2$Txt_Nature_Goods_Code)
data2$Txt_Road_Type_Code = as.numeric(data2$Txt_Road_Type_Code)
data2$Txt_Vehicle_Driven_By_Code = as.numeric(data2$Txt_Vehicle_Driven_By_Code)
data2$Txt_Driver_Exp_Code = as.numeric(data2$Txt_Driver_Exp_Code)
data2$Txt_Claims_History_Code = as.numeric(data2$Txt_Claims_History_Code)
data2$Txt_Driver_Qualification_Code = as.numeric(data2$Txt_Driver_Qualification_Code)
data2$Txt_Incurred_Claims_Code = as.numeric(data2$Txt_Incurred_Claims_Code)
data2$Boo_TPPD_Statutory_Cover_only = as.numeric(data2$Boo_TPPD_Statutory_Cover_only)
data2$Txt_TAC_NOL_Code = as.numeric(data2$Txt_TAC_NOL_Code)
data2$Boo_AntiTheft = as.numeric(data2$Boo_AntiTheft)
data2$Boo_NCB = as.numeric(data2$Boo_NCB)
data2$Boo_OD_Total_Loss = as.numeric(data2$Boo_OD_Total_Loss)
data2$DRV_CLAIM_STATUS = as.factor(data2$DRV_CLAIM_STATUS)
data2$DRV_CLAIM_STATUS = as.numeric(data2$DRV_CLAIM_STATUS)
data2$DRV_CLAIM_STATUS[data2$DRV_CLAIM_STATUS == 1] = 0
data2$DRV_CLAIM_STATUS[data2$DRV_CLAIM_STATUS == 2] = 1
```

We perform linear regression on the transformed data to identify the factors significant in determining the variability of “DRV_CLAIM STATUS”.

```
model1 = lm(DRV_CLAIM_STATUS~., data = data2)
summary(model1)
```

```
Call:
lm(formula = DRV_CLAIM_STATUS ~ ., data = data2)

Residuals:
```

```
     Min       1Q   Median       3Q      Max 
-0.35148 -0.06516 -0.03576 -0.00860  1.44809 

Coefficients: (1 not defined because of singularities)
                                Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   -1.130e-01  2.776e-02  -4.069 4.72e-05 ***
Boo_Endorsement               -6.347e-02  2.587e-03 -24.531  < 2e-16 ***
Txt_Policy_Code               -9.495e-03  1.079e-02  -0.880 0.378904    
Txt_Class_Code                 1.172e-03  3.180e-04   3.685 0.000229 ***
Txt_Zone_Code                         NA         NA      NA       NA    
Num_Vehicle_Age               -5.511e-04  2.972e-04  -1.854 0.063739 .  
Txt_CC_PCC_GVW_Code            2.195e-03  2.889e-04   7.599 3.03e-14 ***
Num_IDV                        5.753e-14  2.146e-14   2.681 0.007351 ** 
Txt_Permit_Code                8.879e-03  9.113e-04   9.743  < 2e-16 ***
Txt_Nature_Goods_Code          4.325e-02  2.831e-03  15.278  < 2e-16 ***
Txt_Road_Type_Code            -1.867e-02  9.500e-04 -19.653  < 2e-16 ***
Txt_Vehicle_Driven_By_Code    -1.784e-02  2.179e-03  -8.190 2.64e-16 ***
Txt_Driver_Exp_Code            5.156e-03  4.366e-04  11.809  < 2e-16 ***
Txt_Claims_History_Code       -7.348e-04  7.416e-04  -0.991 0.321773    
Txt_Driver_Qualification_Code  9.939e-03  8.081e-04  12.299  < 2e-16 ***
Txt_Incurred_Claims_Code       4.455e-03  3.522e-04  12.650  < 2e-16 ***
Boo_TPPD_Statutory_Cover_only -1.059e-02  6.828e-03  -1.552 0.120721    
Txt_TAC_NOL_Code               2.275e-03  4.909e-05  46.335  < 2e-16 ***
Boo_OD_Total_Loss             -4.685e-02  3.441e-03 -13.617  < 2e-16 ***
DRV_CLAIM_AMT                 -1.750e-07  7.545e-09 -23.195  < 2e-16 ***
Boo_AntiTheft                  5.430e-02  3.421e-03  15.875  < 2e-16 ***
Boo_NCB                       -1.867e-04  2.668e-03  -0.070 0.944202    
Num_Net_OD_Premium             5.786e-07  1.667e-07   3.470 0.000520 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2128 on 75178 degrees of freedom
Multiple R-squared:  0.07374,	Adjusted R-squared:  0.07348 
F-statistic:   285 on 21 and 75178 DF,  p-value: < 2.2e-16
```

We notice that vehicle age is not a significant determinant in explaining the variability of the Y variable in spite of EDA yielding contrasting results. To understand this better we investigate further.

```
model2 = lm(DRV_CLAIM_STATUS ~ Num_Vehicle_Age, data = data2)
summary(model2)
```

```
Call:
lm(formula = DRV_CLAIM_STATUS ~ Num_Vehicle_Age, data = data2)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.05858 -0.05173 -0.05121 -0.05094  0.94906 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     0.0509420  0.0010505  48.492   <2e-16 ***
Num_Vehicle_Age 0.0002635  0.0002955   0.892    0.373    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2211 on 75198 degrees of freedom
Multiple R-squared:  1.058e-05,	Adjusted R-squared:  -2.723e-06 
F-statistic: 0.7952 on 1 and 75198 DF,  p-value: 0.3725
```

By performing simple linear regression between claim status and vehicle age, we get a p-value of 0.37 (which is quite high) which leads us to conclude that vehicle age may be excluded as insignificant in this context.

Next, we go ahead and remove the insignificant variables and take a look at the data so far. This is an important step of variable reduction.

data2 = data2[,-c(2,4,5,7,11,13,16,19,22,23)]

str(data2)

```
Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 75200 obs. of  13 variables:
 $ Boo_Endorsement                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Txt_Class_Code                 : num  11 11 14 14 11 14 11 11 14 14 ...
 $ Txt_CC_PCC_GVW_Code            : num  50 50 47 47 50 47 50 51 47 47 ...
 $ Txt_Permit_Code                : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Txt_Nature_Goods_Code          : num  2 2 2 2 2 2 2 2 2 2 ...
 $ Txt_Road_Type_Code             : num  3 3 3 3 3 3 3 3 3 3 ...
 $ Txt_Driver_Exp_Code            : num  6 6 1 1 6 1 6 6 1 1 ...
 $ Txt_Driver_Qualification_Code: num  2 1 2 2 4 4 1 2 1 2 ...
 $ Txt_Incurred_Claims_Code       : num  2 5 2 8 1 5 9 3 1 3 ...
 $ Txt_TAC_NOL_Code               : num  59 59 59 59 59 59 59 59 59 59 ...
 $ Boo_OD_Total_Loss              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ DRV_CLAIM_STATUS               : num  1 1 0 0 1 1 0 1 0 1 ...
 $ Boo_AntiTheft                  : num  0 0 0 0 0 0 0 0 0 0 ...
```

summary(data2)

```
Boo_Endorsement  Txt_Class_Code  Txt_CC_PCC_GVW_Code Txt_Permit_Code Txt_Nature_Goods_
Code Txt_Road_Type_Code Txt_Driver_Exp_Code
 Min.   :0.0000   Min.   :11.00   Min.   :46.00       Min.   :1.000   Min.   :1.000   
Min.   :1.000     Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:11.00   1st Qu.:50.00       1st Qu.:1.000   1st Qu.:2.000   
1st Qu.:3.000     1st Qu.:1.000  
 Median :0.0000   Median :11.00   Median :51.00       Median :1.000   Median :2.000   
Median :4.000     Median :4.000  
 Mean   :0.3278   Mean   :13.13   Mean   :51.41       Mean   :1.534   Mean   :1.812   
Mean   :3.937     Mean   :3.703  
 3rd Qu.:1.0000   3rd Qu.:14.00   3rd Qu.:52.00       3rd Qu.:2.000   3rd Qu.:2.000   
3rd Qu.:5.000     3rd Qu.:6.000  
 Max.   :1.0000   Max.   :23.00   Max.   :64.00       Max.   :5.000   Max.   :2.000   
Max.   :5.000     Max.   :6.000  
 Txt_Driver_Qualification_Code Txt_Incurred_Claims_Code Txt_TAC_NOL_Code Boo_OD_Total_
Loss DRV_CLAIM_STATUS  Boo_AntiTheft  
 Min.   :1.000                 Min.   :1.00             Min.   : 1.00    Min.   :0.000
00   Min.   :0.00000   Min.   :0.0000  
 1st Qu.:1.000                 1st Qu.:1.00             1st Qu.: 5.00    1st Qu.:0.000
00   1st Qu.:0.00000   1st Qu.:0.0000  
 Median :3.000                 Median :3.00             Median : 5.00    Median :0.000
00   Median :0.00000   Median :0.0000  
 Mean   :2.579                 Mean   :3.49             Mean   :11.47    Mean   :0.067
21   Mean   :0.05154   Mean   :0.1979  
 3rd Qu.:4.000                 3rd Qu.:6.00             3rd Qu.: 5.00    3rd Qu.:0.000
00   3rd Qu.:0.00000   3rd Qu.:0.0000  
 Max.   :4.000                 Max.   :9.00             Max.   :59.00    Max.   :1.000
00   Max.   :1.00000   Max.   :1.0000  
```

8. Checking for Multicollinearity

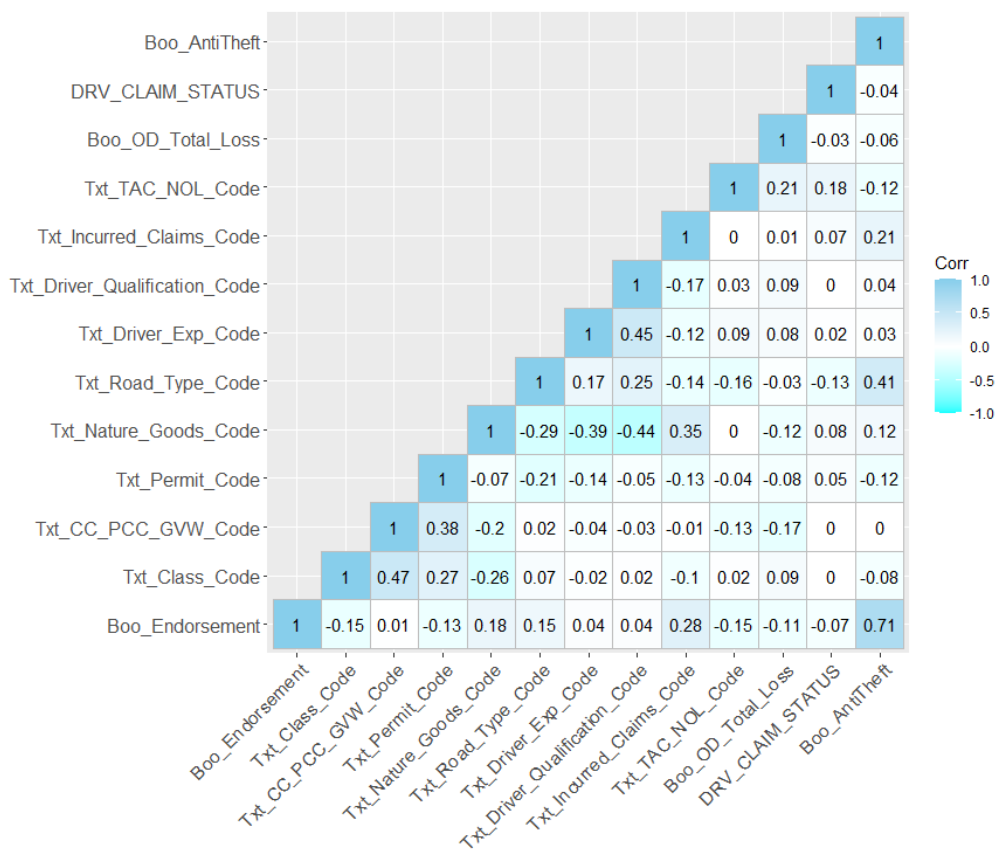
We do a quick VIF test:

vif(model1)

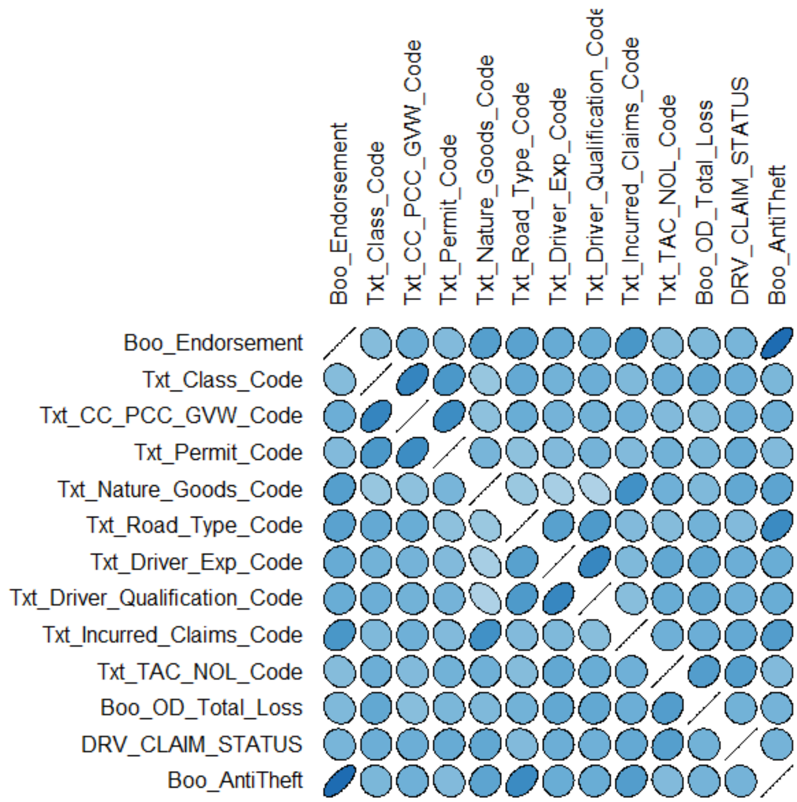
```
  Boo_Endorsement                       Txt_Class_Code              Txt_CC_PCC_GVW_Code         
Txt_Permit_Code
                          2.313075                             1.436071                             1.573
668                                 1.338898
        Txt_Nature_Goods_Code                   Txt_Road_Type_Code                   Txt_Driver_Exp_C
ode Txt_Driver_Qualification_Code
                          1.763373                             1.642934                             1.385
418                                 1.459426
     Txt_Incurred_Claims_Code                       Txt_TAC_NOL_Code                       Boo_OD_Total_L
oss                                 Boo_AntiTheft
                          1.262461                             1.115497                             1.139
700                                 2.604227
```

Next, to proceed to visualize the correlations in order to understand it better.

corr.matrix = round(cor(data2),3)
ggcorrplot(corr.matrix, type = "lower", ggtheme = ggplot2::theme_gray,
        show.legend = TRUE, show.diag = TRUE, colors = c("cyan","white","sky blue"),
        lab = TRUE)



```
my_colors = brewer.pal(7, "Blues")
my_colors = colorRampPalette(my_colors)(100)
plotcorr(corr.matrix , col=my_colors[corr.matrix*50+50] , mar=c(1,1,1,1), )
```



As displayed by the variable inflation factors as well as the correlation plots, we can see that there is hardly any evidence of multicollinearity in the data. Only “Boo_Endorsement” and “Boo_AntiTheft” show signs of relatively high correlation. However, we proceed with principal component analysis in the interest of variable reduction and ease of calculation.

9. Principal Component Analysis

cortest.bartlett(corr.matrix)

```
$chisq
[1] 264.7957

$p.value
[1] 3.617572e-22

$df
[1] 78
```

KMO(corr.matrix)

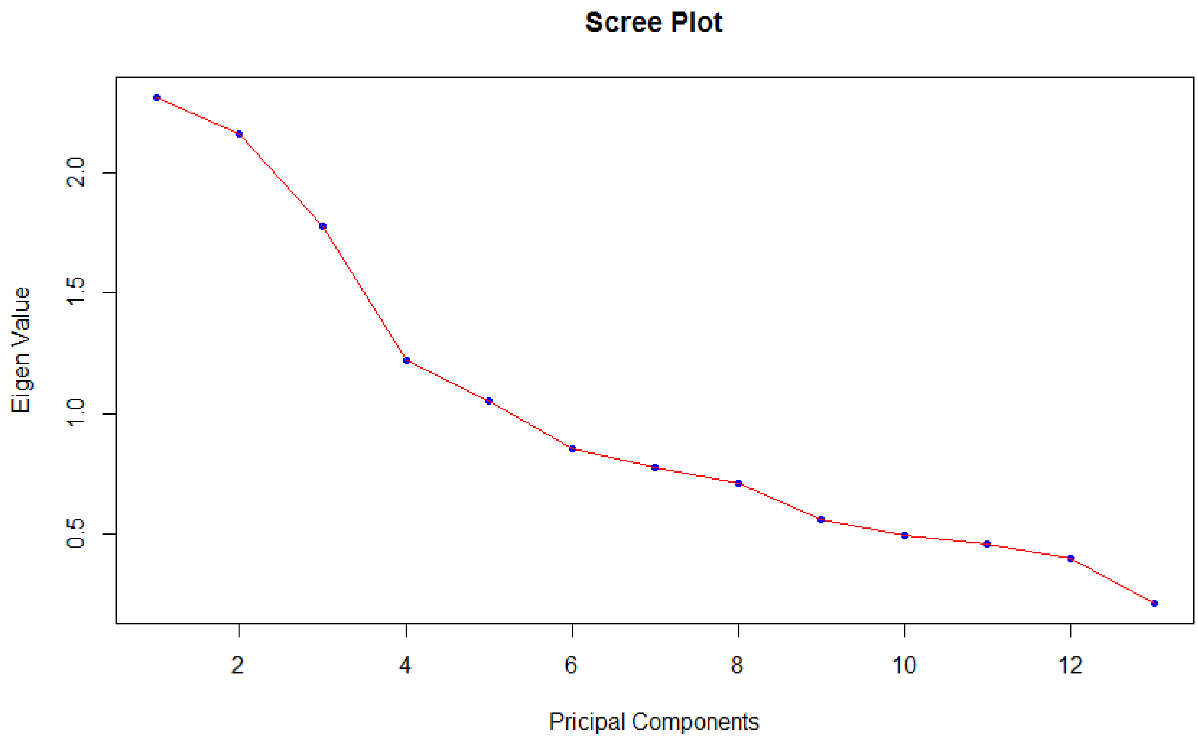
```
Kaiser-Meyer-Olkin factor adequacy
Call: KMO(r = corr.matrix)
Overall MSA =  0.6
MSA for each item = 
              Boo_Endorsement                Txt_Class_Code           Txt_CC_PCC_GVW_C
ode              Txt_Permit_Code 
                         0.55                          0.64                          0
.57                         0.62 
        Txt_Nature_Goods_Code            Txt_Road_Type_Code           Txt_Driver_Exp_C
ode Txt_Driver_Qualification_Code 
                         0.70                          0.52                          0
.70                         0.70 
     Txt_Incurred_Claims_Code               Txt_TAC_NOL_Code             Boo_OD_Total_L
oss             DRV_CLAIM_STATUS 
                         0.73                          0.62                          0
.50                         0.54 
                Boo_AntiTheft 
                         0.52 
```

The Cortest Bartlett test gives a very low p-value of `3.617572e-22` and the Kaiser Meyer Olkin test gives us an adequacy ratio of 60%. Hence, we may safely proceed with PCA.

e = eigen(corr.matrix)
ev = e$values
ev

```
[1] 2.3110226 2.1605192 1.7812019 1.2192042 1.0528413 0.8561576 0.7748363 0.7127855 0.
5625593 0.4953451 0.4581782 0.3990491 0.2162995
```

plot(ev, xlab = "Pricipal Components", ylab="Eigen Value", main = "Scree Plot", pch=20, col="blue")
lines(ev, col="red")



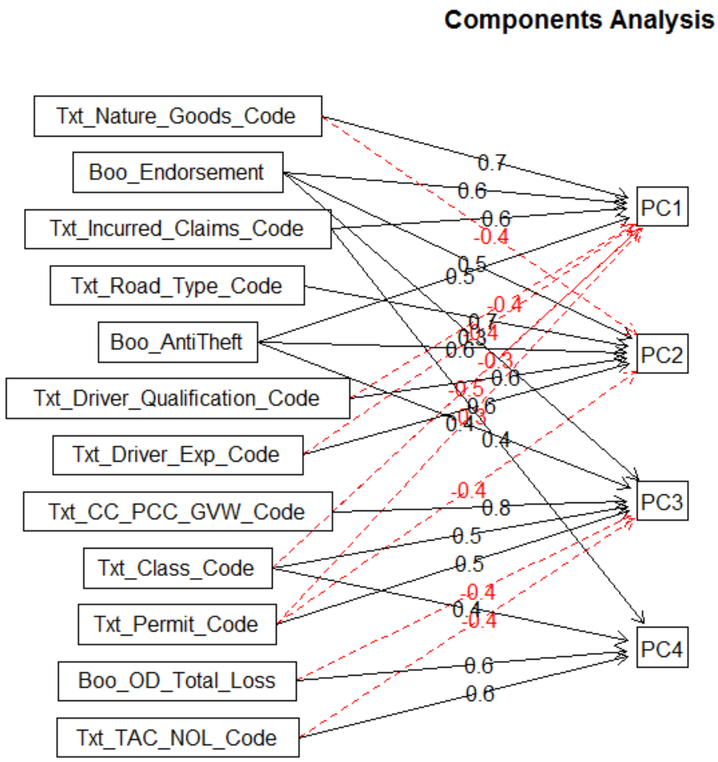
We plot out the eigen values to determine the optimal number of components. We proceed with 4 factors following the elbow rule.

PCA = principal(data2[,-12], nfactors = 4, rotate = "none")
PCA

```
Principal Components Analysis
Call: principal(r = data2[, -12], nfactors = 4, rotate = "none")
Standardized loadings (pattern matrix) based upon correlation matrix
                                PC1   PC2   PC3   PC4   h2   u2 com
Boo_Endorsement                0.59  0.51  0.33  0.20 0.75 0.25 2.8
Txt_Class_Code                -0.49 -0.12  0.51  0.38 0.66 0.34 3.0
Txt_CC_PCC_GVW_Code           -0.33 -0.12  0.75  0.17 0.72 0.28 1.5
Txt_Permit_Code               -0.32 -0.38  0.50  0.09 0.50 0.50 2.7
Txt_Nature_Goods_Code          0.74 -0.40 -0.04  0.01 0.70 0.30 1.5
Txt_Road_Type_Code            -0.09  0.69  0.18 -0.16 0.54 0.46 1.3
Txt_Driver_Exp_Code           -0.36  0.56 -0.25  0.09 0.51 0.49 2.2
Txt_Driver_Qualification_Code -0.43  0.59 -0.15  0.02 0.56 0.44 2.0
Txt_Incurred_Claims_Code       0.57 -0.05  0.07  0.43 0.52 0.48 1.9
Txt_TAC_NOL_Code              -0.14 -0.14 -0.44  0.55 0.53 0.47 2.2
Boo_OD_Total_Loss             -0.18  0.04 -0.39  0.61 0.56 0.44 1.9
Boo_AntiTheft                  0.51  0.61  0.36  0.20 0.81 0.19 2.9

                       PC1  PC2  PC3  PC4
SS loadings           2.31 2.14 1.77 1.15
Proportion Var        0.19 0.18 0.15 0.10
Cumulative Var        0.19 0.37 0.52 0.61
Proportion Explained  0.31 0.29 0.24 0.16
Cumulative Proportion 0.31 0.60 0.84 1.00

Mean item complexity =  2.2
Test of the hypothesis that 4 components are sufficient.

The root mean square of the residuals (RMSR) is  0.09
 with the empirical chi square  86890.9  with prob <  0

Fit based upon off diagonal values = 0.79
```

We see that the 4 factors are able to explain 79% of the data fluctuations which is a good fit.

```
fa.diagram(PCA, simple = FALSE)
```



By studying the factor-loadings of the individual components we rename them according to their dominant characteristics. We name PC1, PC2, PC3, PC4 as "Nature of Goods, Edorsements and Discounts", "Driver Details and Road Type", "Vehicle Details" and "Loss and Claim Details" respectively.

```
PCAdata = data.frame(PCA$scores)

colnames(PCAdata)[1] = "Nature of Goods, Edorsements and Discounts"
colnames(PCAdata)[2] = "Driver Details and Road Type"
colnames(PCAdata)[3] = "Vehicle Details"
colnames(PCAdata)[4] = "Loss and Claim Details"

new.data = cbind(PCAdata,data2$DRV_CLAIM_STATUS)
colnames(new.data)[5] = "Fraudulent Claim"
attach(new.data)
```

After finalizing the data and attaching the dependent variable, we proceed with model building.

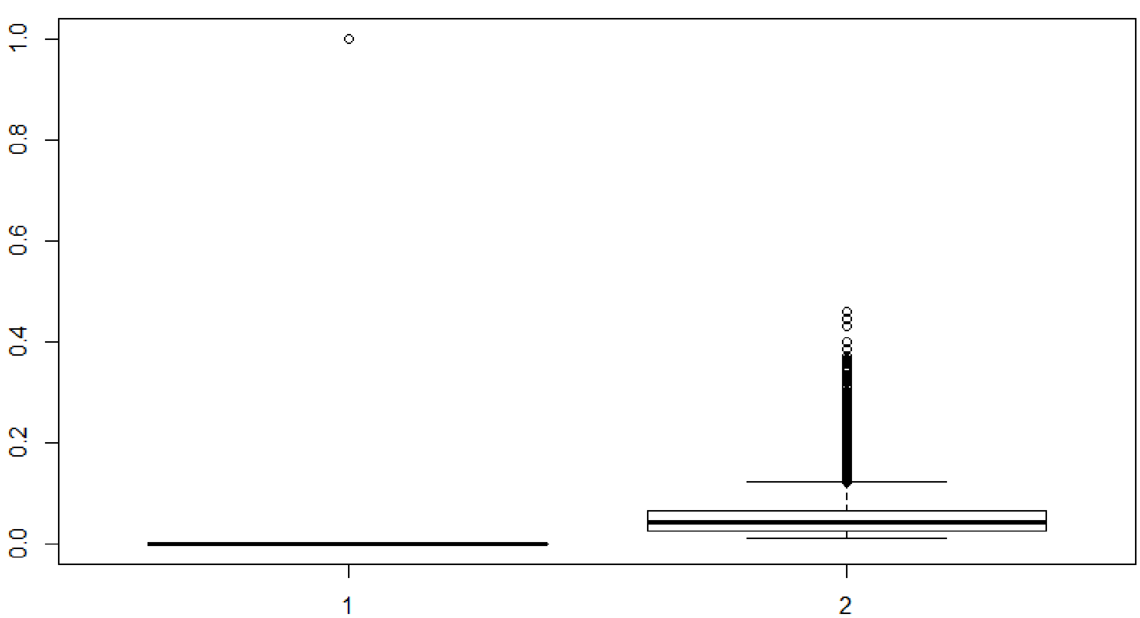
10. Logistic Regression

We begin by splitting the data into train and test data in 75:25 ratio.

```
set.seed(100)
split = sample.split(new.data$`Fraudulent Claim`, SplitRatio = 0.75)
train.data = subset(new.data, split == TRUE)
test.data = subset(new.data, split == FALSE)

model3 = glm(`Fraudulent Claim`~., data = train.data, family = "binomial")
summary(model3)
```

```
Call:
glm(formula = `Fraudulent Claim` ~ ., family = "binomial", data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1113  -0.3540  -0.2837  -0.2089   3.0673

Coefficients:
                                             Estimate Std. Error  z value Pr(>|z|)
(Intercept)                                  -3.13032    0.02282 -137.148   <2e-16 ***
`Nature of Goods, Edorsements and Discounts`  0.07107    0.02288    3.106   0.0019 **
`Driver Details and Road Type`               -0.55033    0.02133  -25.802   <2e-16 ***
`Vehicle Details`                            -0.20582    0.01831  -11.242   <2e-16 ***
`Loss and Claim Details`                      0.37859    0.01724   21.962   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22902  on 56399  degrees of freedom
Residual deviance: 21553  on 56395  degrees of freedom
AIC: 21563

Number of Fisher Scoring iterations: 6
```

We compare in the boxplot how the fitted values of our model stack against the original variable. Due to the imbalance in data we notice a skewness in our model values.



Let us next test our model predictions using test data. We have shown the performance scores below:

prediction1 = predict(model3, newdata = test.data, type = "response")
cmLR = table(test.data$`Fraudulent Claim`,prediction1 > 0.1)
cmLR

```
FALSE  TRUE
  0 15987  1844
  1   722   247
> #Accuracy
> sum(diag(cmLR))/sum(cmLR)
[1] 0.8635106
> #Recall or TPR
> recall = 15987/(15987+722)
> print(recall)
[1] 0.9567898
> #Precision
> precision = 15987/(15987+1844)
> print(precision)
[1] 0.8965846
> #Specificity
> 247/(247+722)
[1] 0.254902
> #FPR
> 1844/(1844+247)
[1] 0.8818747
> #F1 Score
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.9257093
```

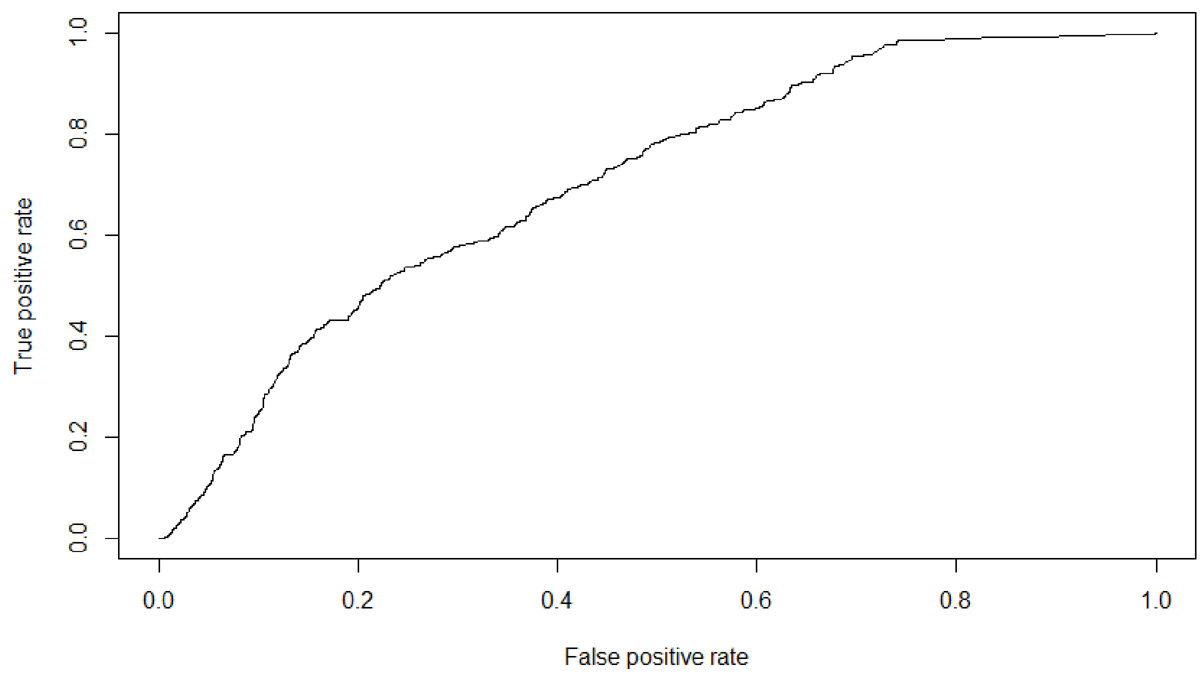
While Logistic Regression on the PCA data gives us a high accuracy of 86.35%, we face the challenge of
A very low specificity of only 25.49%. This is a problem since we are required to correctly identify the fraudulent claims and that requires a much higher specificity score.

ROCRpred = prediction(prediction1, test.data$`Fraudulent Claim`)
as.numeric(ROCR::performance(ROCRpred,"auc")@y.values)

```
0.7050193
```

perf = ROCR::performance(ROCRpred, "tpr","fpr")

plot(perf)



We get an area under the curve(AUC) of 70.50% and we plot the ROC curve.

We proceed with logistic regression using the original numeric dataset without PCA to see if get better results

set.seed(100)
split = sample.split(data2$DRV_CLAIM_STATUS, SplitRatio = 0.75)
train.data2 = subset(data2, split == TRUE)
test.data2 = subset(data2, split == FALSE)

model4 = glm(DRV_CLAIM_STATUS~., data = train.data2, family = "binomial")
summary(model4)

```
Call:
glm(formula = DRV_CLAIM_STATUS ~ ., family = "binomial", data = train.data2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8298  -0.3220  -0.2452  -0.1788   3.6607

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -7.9302145  0.4068737 -19.491  < 2e-16 ***
Boo_Endorsement               -1.1279798  0.0773676 -14.579  < 2e-16 ***
```

```
Txt_Class_Code                  0.0167082  0.0076003   2.198   0.0279 *
Txt_CC_PCC_GVW_Code             0.0292536  0.0066980   4.368 1.26e-05 ***
Txt_Permit_Code                 0.1550796  0.0199962   7.755 8.80e-15 ***
Txt_Nature_Goods_Code           1.6395616  0.0957187  17.129  < 2e-16 ***
Txt_Road_Type_Code             -0.3801424  0.0239475 -15.874  < 2e-16 ***
Txt_Driver_Exp_Code             0.0695066  0.0103785   6.697 2.13e-11 ***
Txt_Driver_Qualification_Code   0.1984241  0.0194633  10.195  < 2e-16 ***
Txt_Incurred_Claims_Code        0.0905171  0.0075916  11.923  < 2e-16 ***
Txt_TAC_NOL_Code                0.0251907  0.0008838  28.502  < 2e-16 ***
Boo_OD_Total_Loss              -0.9362629  0.1124004  -8.330  < 2e-16 ***
Boo_AntiTheft                   0.9212690  0.0982257   9.379  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22902  on 56399  degrees of freedom
Residual deviance: 19911  on 56387  degrees of freedom
AIC: 19937

Number of Fisher Scoring iterations: 7
```

prediction2 = predict(model4, newdata = test.data2, type = "response")
cmLR2 = table(test.data2$DRV_CLAIM_STATUS,prediction2 > 0.1)
cmLR2

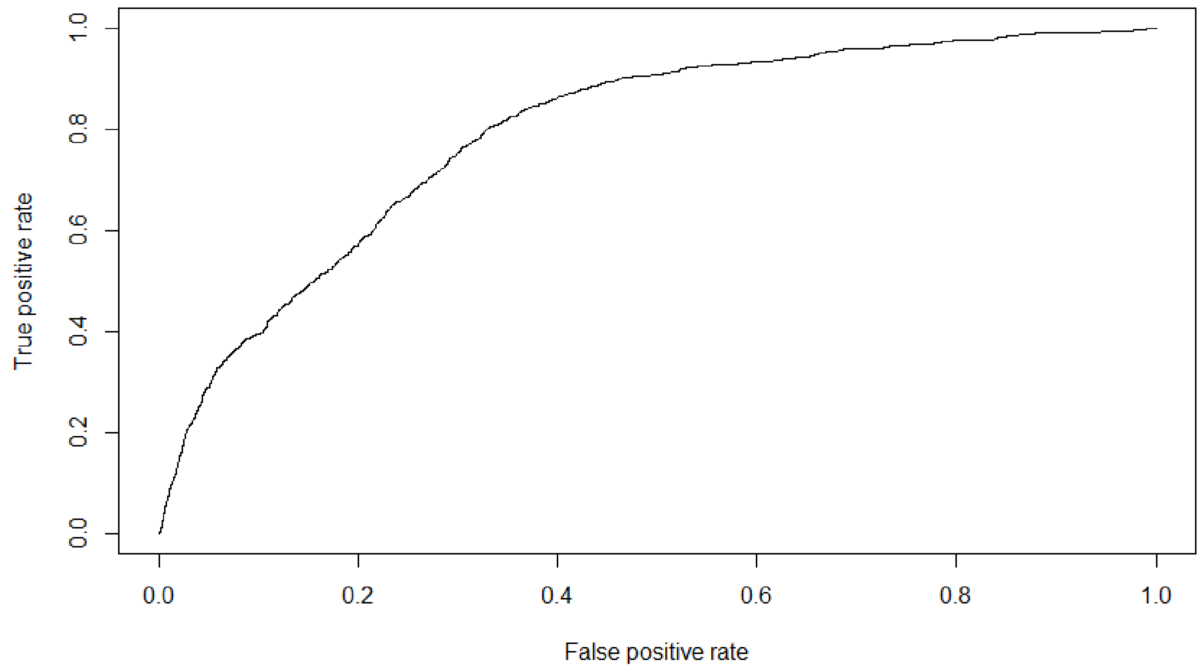
```
FALSE  TRUE
  0 15977  1854
  1   584   385
> #Accuracy
> sum(diag(cmLR2))/sum(cmLR2)
[1] 0.8703191
> #Recall or TPR
> recall = 15977/(15977+584)
> print(recall)
[1] 0.9647364
> #Precision
> precision = 15977/(15977+1854)
> print(precision)
[1] 0.8960238
> #Specificity
> 385/(584+385)
[1] 0.3973168
> #FPR
> 1854/(1854+385)
[1] 0.8280482
> #F1 Score
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.9291114
```

Here, we get a slightly improved accuracy of 87.03% and a specificity of 39.73% which is better than the previous model.

ROCRpred1 = prediction(prediction2, test.data2$DRV_CLAIM_STATUS)
as.numeric(ROCR::performance(ROCRpred1, "auc")@y.values)

```
0.7940176
```

perf1 = ROCR::performance(ROCRpred1, "tpr","fpr")
plot(perf1)



We see that the AUC and ROC has also improved. AUC has increased by nearly 13% to 79.40%.

11. SMOTE

We perform SMOTE to try and balance out the data. Maybe, this will give us better results.

train.data$`Fraudulent Claim` = as.factor(train.data$`Fraudulent Claim`)
str(train.data)

```
data.frame':      56400 obs. of  5 variables:
 $ Nature of Goods, Edorsements and Discounts: num  -0.195 0.247 0.085 0.647 -0.611 ..
.
 $ Driver Details and Road Type              : num  -0.5524 -0.8176 -1.1768 -1.2329 -0
.0687 ...
 $ Vehicle Details                           : num  -1.58 -1.46 -1.33 -1.23 -1.75 ...
 $ Loss and Claim Details                    : num  0.669 1.075 0.691 1.525 0.554 ...
 $ Fraudulent Claim                          : Factor w/ 2 levels "0","1": 2 2 1 1 2 2
1 2 1 2 ...
```

set.seed(1000)
balanced.data = SMOTE(`Fraudulent Claim` ~.,perc.over = 500 , data = train.data , k = 5, perc.under = 800)
table(balanced.data$`Fraudulent Claim`)

```
     0      1
116280  17442
```

We perform logistic regression using the SMOTE data:

LR.smote = glm(`Fraudulent Claim`~., data = balanced.data, family = "binomial")
summary(LR.smote)

```
Call:
glm(formula = `Fraudulent Claim` ~ ., family = "binomial", data = balanced.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6165  -0.5632  -0.4426  -0.3037   2.7530

Coefficients:
                                              Estimate Std. Error  z value Pr(>|z|)
(Intercept)                                  -2.125419   0.009721 -218.632  < 2e-16 **
*
`Nature of Goods, Edorsements and Discounts`  0.075002   0.009955    7.534 4.91e-14 **
*
`Driver Details and Road Type`               -0.580003   0.009497  -61.073  < 2e-16 **
*
`Vehicle Details`                            -0.199688   0.007818  -25.541  < 2e-16 **
*
`Loss and Claim Details`                      0.395724   0.007780   50.866  < 2e-16 **
*
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 103558  on 133721  degrees of freedom
Residual deviance:  95986  on 133717  degrees of freedom
AIC: 95996

Number of Fisher Scoring iterations: 5
```

prediction3 = predict(LR.smote, newdata = test.data, type = "response")
cmLR3 = table(test.data$`Fraudulent Claim`,prediction3 > 0.1)
cmLR3

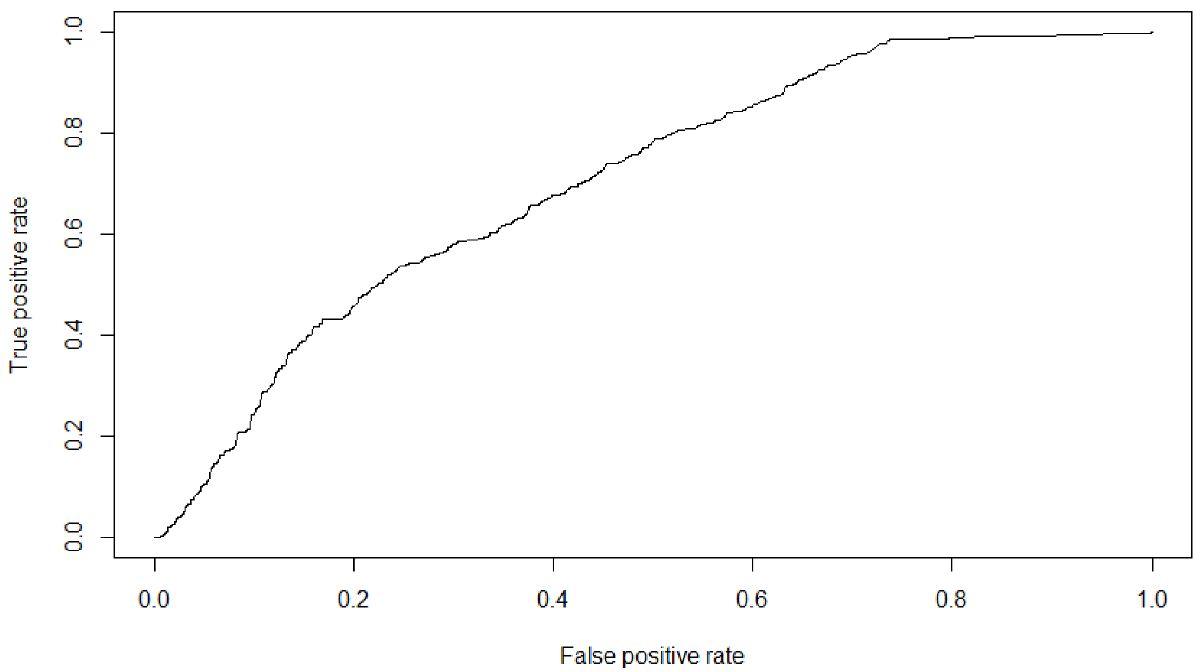
```
  FALSE  TRUE
  0  7256 10575
  1   147   822
> #Accuracy
> sum(diag(cmLR3))/sum(cmLR3)
[1] 0.4296809
> #Recall or TPR
> recall = 10575/(10575+147)
> print(recall)
[1] 0.9862899
> #Precision
> precision = 10575/(10575+7256)
> print(precision)
[1] 0.5930683
> #Specificity
> 822/(147+822)
[1] 0.8482972
> #FPR
> 7256/(7256+822)
[1] 0.8982421
> #F1 Score
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.7407278
```

Logistic Regression with SMOTE gives us a model with a much lower overall accuracy of 42.96% but a much higher specificity of 84.82%. So this would be a better model for solely identifying fraudulent claims.

ROCRpred2 = prediction(prediction3, test.data$`Fraudulent Claim`)
as.numeric(ROCR::performance(ROCRpred2, "auc")@y.values)

```
0.7049104
```

perf2 = ROCR::performance(ROCRpred2, "tpr","fpr")
plot(perf2)



AUC value gives us 70.49% and the performance chart is displayed above.

12. Naïve Bayes

NBmodel = naiveBayes(`Fraudulent Claim` ~., data = train.data)
NBpredTest = predict(NBmodel, newdata = test.data, type = "class")
cmNB = table(test.data$`Fraudulent Claim`, NBpredTest)
cmNB

```
NBpredTest
         0     1
  0 17812    19
  1   969     0
> #Accuracy
> sum(diag(cmNB))/sum(cmNB)
[1] 0.9474468
> #Recall or TPR
> recall = 17812/(17812+969)
> print(recall)
[1] 0.9484053
```

```
> #Precision
> precision = 17812/(17812+19)
> print(precision)
[1] 0.9989344
> #Specificity
> 0/969
[1] 0
> #FPR
> 19/(19+0)
[1] 1
> #F1 Score
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.9730143
```

Naïve Bayes gives us 0 specificity and 1 false positive rate which means it has identified all occurrences as positive. This model is not of much use in this regard.

Using SMOTE:

NBmodel.bal = naiveBayes(`Fraudulent Claim` ~., data = balanced.data)
NBpredTest.bal = predict(NBmodel.bal, newdata = test.data)
cmNB.bal = table(test.data$`Fraudulent Claim`, NBpredTest.bal)
cmNB.bal

```
NBpredTest.bal
        0     1
  0 16912   919
  1   775   194
> #Accuracy
> sum(diag(cmNB.bal))/sum(cmNB.bal)
[1] 0.9098936
> #Recall or TPR
> recall = 17286/(17286+896)
> print(recall)
[1] 0.9507205
> #Precision
> precision = 17286/(17286+545)
> print(precision)
[1] 0.9694353
> #Specificity
> 194/(194+775)
[1] 0.2002064
> #FPR
> 545/(71+545)
[1] 0.8847403
> #F1 Score
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.9599867
```

Again, a dreadfully low specificity of 20% negates the usefulness of naïve bayes in this problem.

13. Random Forest

Before we proceed with Random forest we convert the data to factor for classification model.

data3 = data2

data3$Boo_Endorsement = as.factor(data3$Boo_Endorsement)
data3$Txt_Class_Code = as.factor(data3$Txt_Class_Code)
data3$Txt_CC_PCC_GVW_Code = as.factor(data3$Txt_CC_PCC_GVW_Code)
data3$Txt_Permit_Code = as.factor(data3$Txt_Permit_Code)
data3$Txt_Nature_Goods_Code = as.factor(data3$Txt_Nature_Goods_Code)
data3$Txt_Road_Type_Code = as.factor(data3$Txt_Road_Type_Code)
data3$Txt_Driver_Exp_Code = as.factor(data3$Txt_Driver_Exp_Code)
data3$Txt_Driver_Qualification_Code = as.factor(data3$Txt_Driver_Qualification_Code)
data3$Txt_Incurred_Claims_Code = as.factor(data3$Txt_Incurred_Claims_Code)
data3$Txt_TAC_NOL_Code = as.factor(data3$Txt_TAC_NOL_Code)
data3$Boo_OD_Total_Loss = as.factor(data3$Boo_OD_Total_Loss)
data3$DRV_CLAIM_STATUS = as.factor(data3$DRV_CLAIM_STATUS)
data3$Boo_AntiTheft = as.factor(data3$Boo_AntiTheft)

set.seed(100)
split = sample.split(data3$DRV_CLAIM_STATUS, SplitRatio = 0.75)
train.data3 = subset(data3, split == TRUE)
test.data3 = subset(data3, split == FALSE)

RF.model = randomForest(DRV_CLAIM_STATUS~., data = train.data3, ntree = 500, mtry = 5, nodesize = 10, importance = TRUE)

RFpredTest = predict(RF.model, newdata = test.data3, type = "class")
cmRF = table(test.data3$DRV_CLAIM_STATUS, RFpredTest)
cmRF

```
RFpredTest
        0     1
  0 17790    41
  1   917    52
> #Accuracy
> sum(diag(cmRF))/sum(cmRF)
[1] 0.9490426
> #Recall or TPR
> recall = 17790/(17790+917)
> print(recall)
[1] 0.9509809
> #Precision
> precision = 17790/(17790+41)
> print(precision)
[1] 0.9977006
> #Specificity
> 52/(917+52)
[1] 0.05366357
> #FPR
> 41/(41+52)
[1] 0.4408602
> #F1 Score
```

```
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.9737807
```

Random Forest gives good accuracy and precision. Even though the Specificity is less so is the FPR. It is a good model but may not be that useful in the present scenario.

We tune the random forest to get a better score and fit:

tuned.RFmodel = tuneRF(x=train.data3[,-12], y = train.data3$DRV_CLAIM_STATUS, mtryStart = 3, stepFactor = 1.5, ntreeTry = 501, improve = 0.0001,
trace = TRUE, plot = TRUE, doBest = TRUE, importance = TRUE)

train.data3$Predict.Class = predict(tuned.RFmodel, train.data3, type = "class")

cmtRF = table(train.data3$DRV_CLAIM_STATUS,train.data3$Predict.Class)
cmtRF

```
        0     1
  0 53491     2
  1  2869    38
> #Accuracy
> sum(diag(cmtRF))/sum(cmtRF)
[1] 0.9490957
> #Recall or TPR
> recall = 53491/(53491+2869)
> print(recall)
[1] 0.9490951
> #Precision
> precision = 53491/(53491+2)
> print(precision)
[1] 0.9999626
> #Specificity
> 38/(2869+38)
[1] 0.0130719
> #FPR
> 2/(2+38)
[1] 0.05
> #F1 Score
> F1 = (2*precision*recall)/(precision+recall)
> print(F1)
[1] 0.9738651
```

Very high precision but very low specificity.

14. Bagging

bag.model = bagging(DRV_CLAIM_STATUS ~.,data = train.data2, control=rpart.control(xval = 0, maxdepth = 20, minsplit = 10), coob = TRUE)

BagpredTest = predict(bag.model, newdata = test.data2, type = "class")

cmBag = table(test.data2$DRV_CLAIM_STATUS, BagpredTest)

cmBag

```
#Accuracy
> sum(diag(cmLR3))/sum(cmLR3)
[1] 0.4296809
```

Bagging does not provide any better results.

15. XGBoost

setDT(balanced.data)
setDT(test.data)

features.train = as.matrix(balanced.data[,-5])
label.train = as.matrix(balanced.data$`Fraudulent Claim`)
features.test = as.matrix(test.data[,-5])

XGtrain = xgb.DMatrix(data = features.train, label = label.train)

```
XGBmodel = xgboost(data = features.train , label = label.train,  eta = 0.1,
              max_depth = 3,
              nrounds = 10,
              nfold = 5,
              objective = "binary:logistic",
              verbose = 0,
              early_stopping_rounds = 10)
```

XGBpredTest = predict(XGBmodel, features.test)
cmXGB = table(test.data$`Fraudulent Claim`, XGBpredTest>0.1)
cmXGB
sum(diag(cmXGB))/sum(cmXGB)

```
     TRUE
  0 17831
  1   969
> sum(diag(cmXGB))/sum(cmXGB)
[1] 0.9484574
```

XGBoost gives an Accuarcy of 95%.

16. Conclusion and Model Comparison

We have built the following models in R and carried out the mentioned performance measures to get an idea of the best model and how it fits into the current situation to bring about a solution. Out of all the models random forest has given us the highest accuracy, recall and precision with the lowest False Positive rate(FPR) which determines the probability of false alarm. Random Forest also gives us the highest F1 score. Naïve Bayes is the second best model due to the false positive rate being 1. However, even though we may correctly identify all non-fraudulent claims we may also identify fraudulent ones as genuine using the naïve bayes model. Looking at the "Specificity" score we proceed to select logistic regression with SMOTE. Since in the context of the problem it is imperative that we correctly identify fraudulent claims, Random Forest or logistic regression using SMOTE is a much better option. Using these models we can correctly identify about 95% of fraudulent claims with a 47% chance of those claims turning out to be genuine, which is a good tradeoff.

We have also used several model tuning parameters as evidenced by the R code and used ensemble techniques like bagging and boosting to try and get a better accuracy reading.

EDA tells us that cases coming in from Delhi have the highest number of frauds and measure may be implemented to filter those cases. Also, newer claims with little to no claim history are more likely to be fraudulent. Also cases in which the driver involved is very experienced also seems to indicate high probability of fraud. So, these kinds of cases should be scrutinized further before insuring them.

Therefore, by isolating the significant variables and using them to build a logistic regression or random forest model we are able to preconceive the population among the clients of the general insurance company who may be potential fraudsters. Thereby, we provide an opportunity to mitigate risk for the company by identifying clients beforehand who should not be insured based on their demographic and financial data. Such insights can spare the company from suffering huge losses due to information fraud and such statistical models can shed light on how the company should proceed in the future to avoid those kinds of losses completely.

Train Data Model Performance:

	Accuracy	Recall	Precision	Specificity	FPR	F1 Score
Logistic Regression with PCA	0.87	0.96	0.90	0.26	0.88	0.93
Logistic Regression	0.87	0.96	0.90	0.40	0.82	0.93
Logistic Regression with SMOTE	0.48	0.97	0.47	0.78	0.93	0.63
Naïve Bayes	0.95	0.95	0.99	0.00	1.00	0.97
Naïve Bayes with SMOTE	0.92	0.95	0.96	0.16	0.84	0.96
Random Forest	0.95	0.95	0.99	0.01	0.99	0.97
Bagging	0.48					
XGBoost	0.95					

Test Data Model Performance:

	Accuracy	Recall	Precision	Specificity	FPR	F1 Score
Logistic Regression with PCA	0.86	0.96	0.90	0.25	0.75	0.93
Logistic Regression	0.87	0.96	0.90	0.40	0.60	0.93
Logistic Regression with SMOTE	0.48	0.97	0.52	0.85	0.15	0.69
Naïve Bayes	0.95	0.95	0.99	0.00	1.00	0.97
Naïve Bayes with SMOTE	0.92	0.95	0.97	0.20	0.80	0.96
Random Forest	0.95	0.95	0.99	0.05	0.95	0.97
Bagging	0.43					
XGBoost	0.95					