Credit Card Fraud Detection

Introduction

Credit card fraud is a major concern for financial institutions and consumers alike. Fraudulent transactions not only result in financial loses for banks, but can damage their reputation and lose the trust of customers. The increase in online transactions over the past decade has made real-time detection and prevention of financial fraud even more important.

In the banking industry, detecting credit card fraud using machine learning is not just a trend; it is a necessity for banks, as they need to put proactive monitoring and fraud prevention mechanisms in place. Machine learning helps these institutions reduce time-consuming manual reviews, costly chargebacks and fees, and denial of legitimate transactions.

The aim of this project is to analyse the data within the dataset and recommend a ML model that could be used for a practical credit card fraud detection system.

Primary Objective

1. The main objective of the project is to train a machine learning algorithm on the dataset to successfully predict fraudulent transactions
2. Data Understanding- all feature/variables and data types
3. Performing the EDA (exploratory data analysis)- pre-process the dataset, briefly exploring features & engineering new features where possible
4. Build the most effective ML model to detect the frauds
5. Select an optimal cut-off for the model
6. Estimate the business impact and potential savings

Project Outline

1. Data Collection and Understanding
2. Data Cleaning and Analysis
    - EDA • Univariate analysis and bivariate analysis • Descriptive statistics
3. Data Modelling
    - Logistic Regression
    - Decision Tree
    - Random Forest
4. Model Evaluation & Selection
5. Performing CBA- Cost Benefit Analysis
6. Conclusion

Data Collection and Understanding:

The data set contains credit card transactions of around 1,000 cardholders with a pool of 800 merchants from 1 Jan 2019 to 31 Dec 2020. This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It contains credit card transactions of around 1,000 cardholders with a pool of 800 merchants. It contains a total of 18,52,394 transactions, out of which 9,651 are fraudulent transactions.

The data set is highly imbalanced, with the positive class (frauds) accounting for 0.52% of the total transactions. The variable details of the data set are as follows:

- 'Is_fraud' column is our response variable and it takes value 1 in case of fraud and 0 otherwise.
- All other variables mentioned in below table are predictors

Variable Name	**Description**
trans_date_trans_time	Time stamp of transaction date and time
cc_num	Credit card number
Merchant	Merchant description (1000 distinct values) with whom the transaction is made
Category	Category of the merchant (14 distinct categories)
Amt	transaction amount
First	First name of the Credit card holder
Last	Last name of the Credit card holder
Gender	Gender of the Credit card holder
Street	Address of the Credit card holder
City	City of the Credit card holder
State	State of the Credit card holder
Zip	Zipcode
Lat	Latitude of the Credit card holder location
Long	longitude of the Credit card holder location
City_pop	Population of the city
Job	Job of the Credit card holder
Dob	Date of birth of the Credit card holder
Trans_num	Transaction number
Unix time	
Merch_lat	Latitude of the merchant location
Merch_long	longitude of the merchant location
Is_fraud	Is transaction is fraudulent or not

The data type of all variables are as follows:

1. Categorical Variables (14): cc_num, Merchant, Category, Gender, city, state, ZIP,job,is_fraud,City_pop
2. Continuous variables (5): row seq no, amt,lat,long, dob,Trans_num,Unix_time, Merch_lat, Merch_long

Data Cleaning and Analysis (EDA):

*****Since the unsampled data on Kaggle was huge in numbers so to perform EDA, we have merged the test and train data given by upGrad to make it unaltered and raw*****

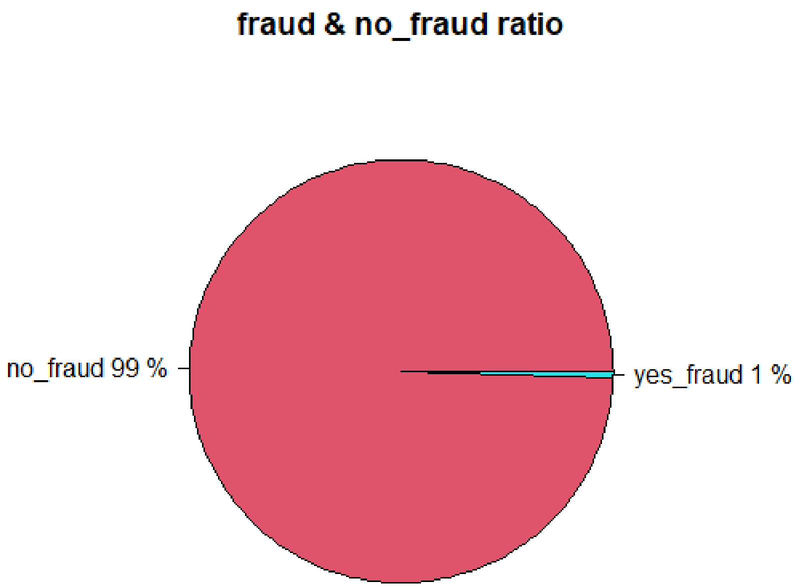
The merged dataset has 463099 records in total, out of which 'is_fraud' is a response variable (value 1 in case of fraud and 0 non-fraud) and all other variables are predictors.

i. **Data Analysis: observations**

   i. There is no null or missing value in the data:

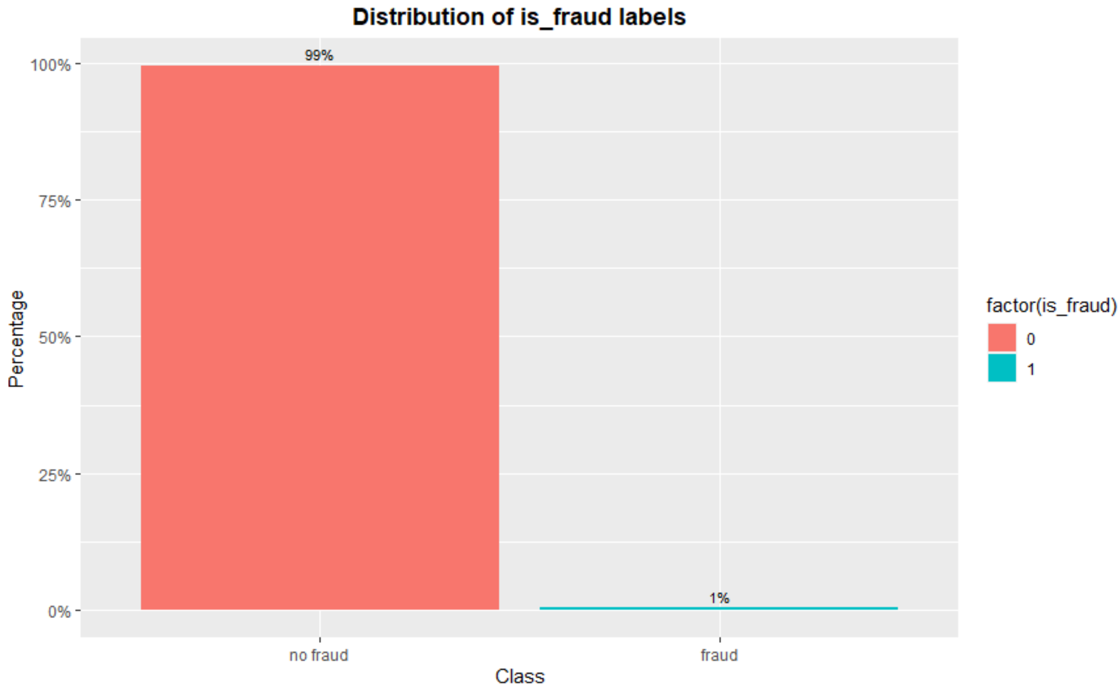
```
> a=sum(is.na(unsampled$is_fraud))
> a
[1] 0
```

   ii. Fraud and no_fraud ratio




```
##pie chart
unsampled$is_fraud=ifelse(unsampled$is_fraud== 1,"yes_fraud","no_fraud")
Fraud1=unsampled$is_fraud
Fraud1=as.factor(unsampled$is_fraud)
table(Fraud1)
freqfraud= table(Fraud1)
pie(freqfraud)
perce=round(freqfraud/463099*100)
perce
label=paste(names(freqfraud), perce, "%",sep=" ")
label
pie(freqfraud , main="fraud & no_fraud ratio", col= c(2,5), labels = label)
```

iii. Distribution of response variable ‘is_fraud’



R Script

```
ggplot(data = unsampled, aes(x = factor(is_fraud),
                             y = prop.table(stat(count)), fill = factor(is_fraud),
                             label = scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge") +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_x_discrete(labels = c("no fraud", "fraud"))+
  scale_y_continuous(labels = scales::percent)+
  labs(x = 'Class', y = 'Percentage') +
  ggtitle("Distribution of is_fraud labels") +
  common_theme
```

There is high skewness in the data. The number of fraud transactions are very less as compared to non-fraud ones, comprising of only 2409 frauds out of 463099 transactions (0.005% of the data set).

```
> # checking class imbalance
> table(unsampled$is_fraud)

     0      1
460690   2409
> # class imbalance in percentage
> prop.table(table(unsampled$is_fraud))

          0           1
0.994798089 0.005201911
```

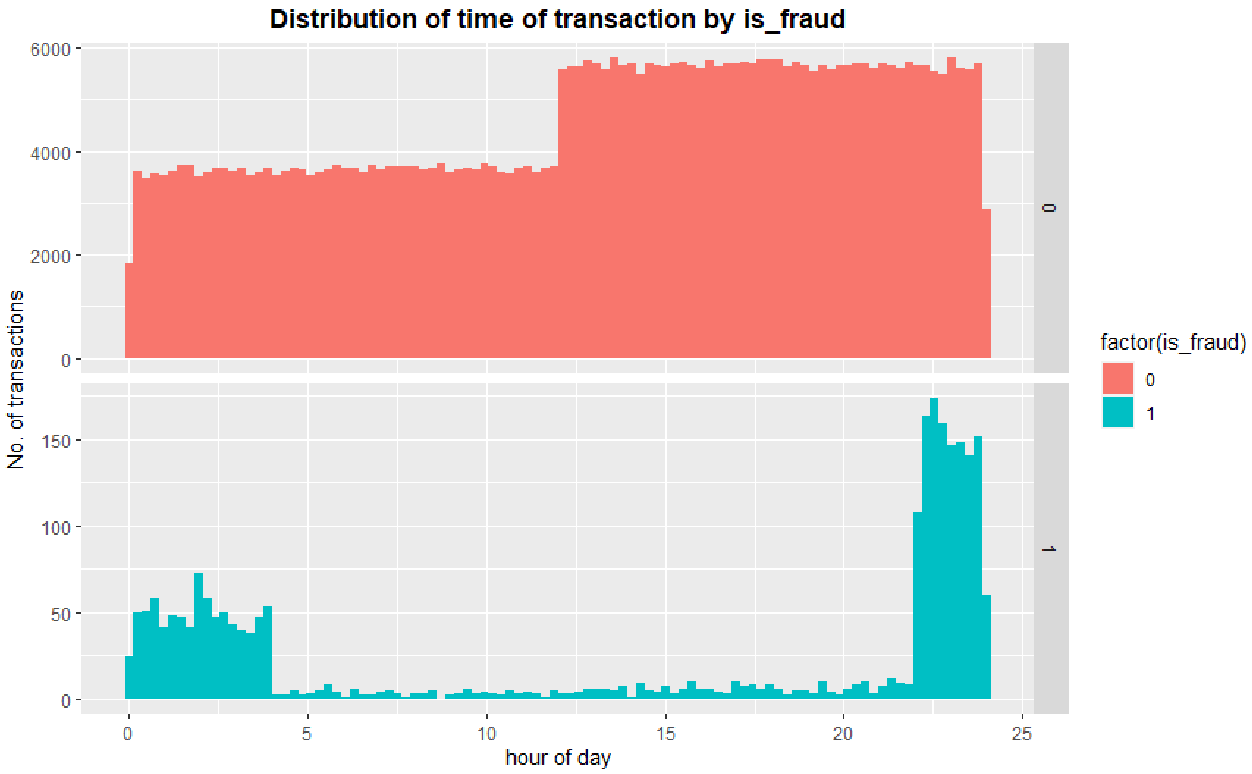
Given the class imbalance ratio with 99.4% of cases being non-fraudulent transactions, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

2. Data Visualization: Bivariate analysis\Feature Engineering

iv. Distribution of variable ‘unix_time’ by ‘is_fraud’

- here ‘unix_time’ feature got engineered to hour_of_day

```
unsampled$hour_of_day <- (unsampled$unix_time/3600) %% 24
```

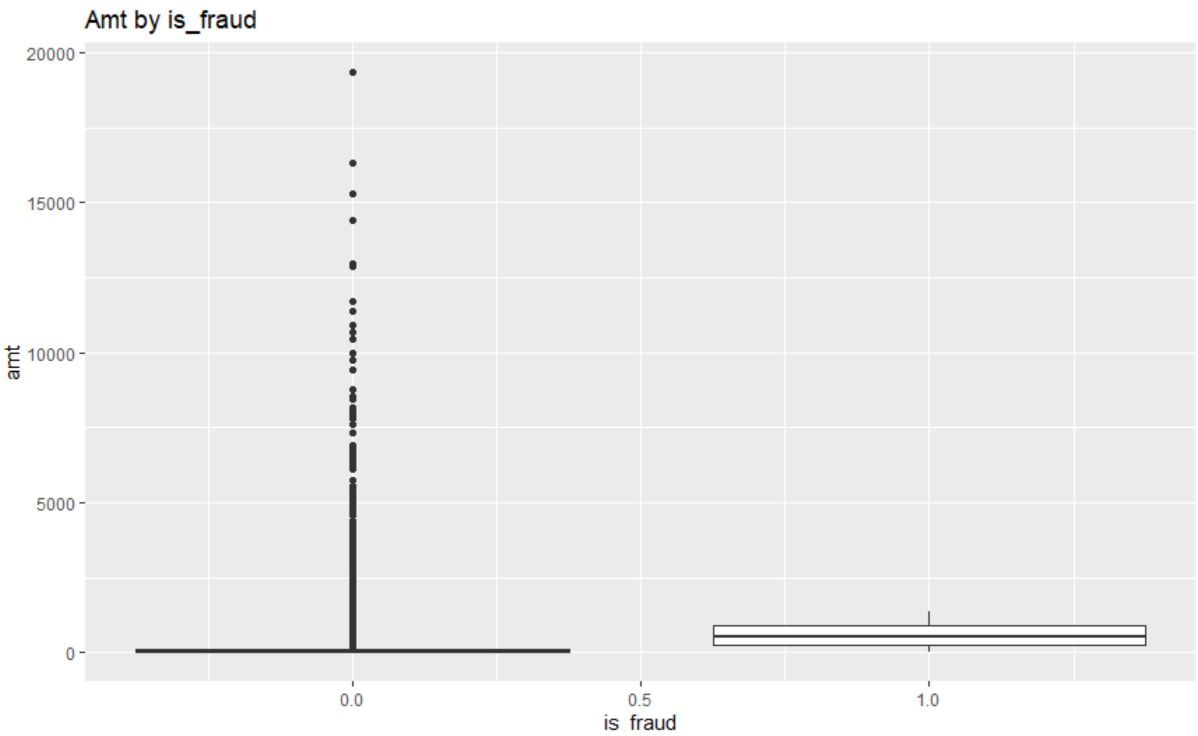



```
unsampled %>%
  ggplot(aes(x = hour_of_day, fill = factor(is_fraud))) + geom_histogram(bins = 100)+
  labs(x = 'hour of day', y = 'No. of transactions') +
  ggtitle('Distribution of time of transaction by is_fraud') +
  facet_grid(is_fraud ~ ., scales = 'free_y') + common_theme
```

Inference: it appears that the fraud transactions occur more between 9 PM to 2 AM at night.

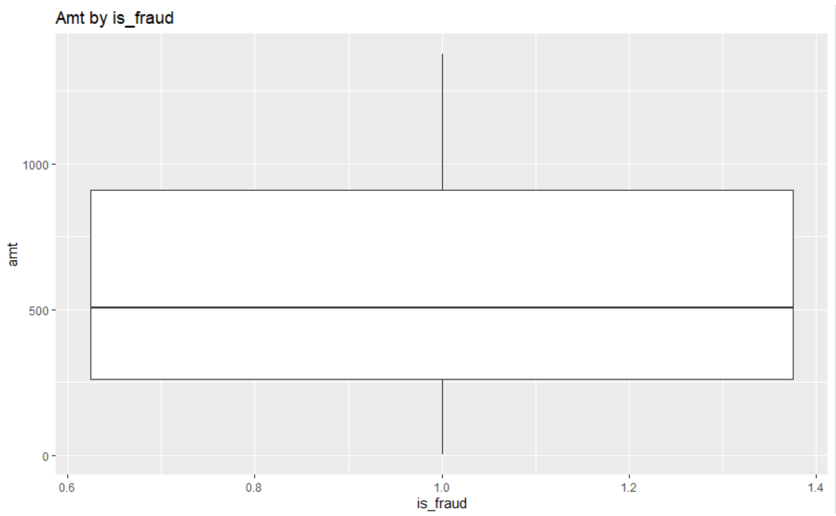
v. Distribution of variable 'Amt' by 'is_fraud'

- Fraudulent transactions are normally of lower amount




```
#Distribution of Amt & is_fraud
ggplot(unsampled,aes(x=is_fraud, y=amt,group=is_fraud))+
  geom_boxplot()+
  ggtitle("Amt by is_fraud")
```

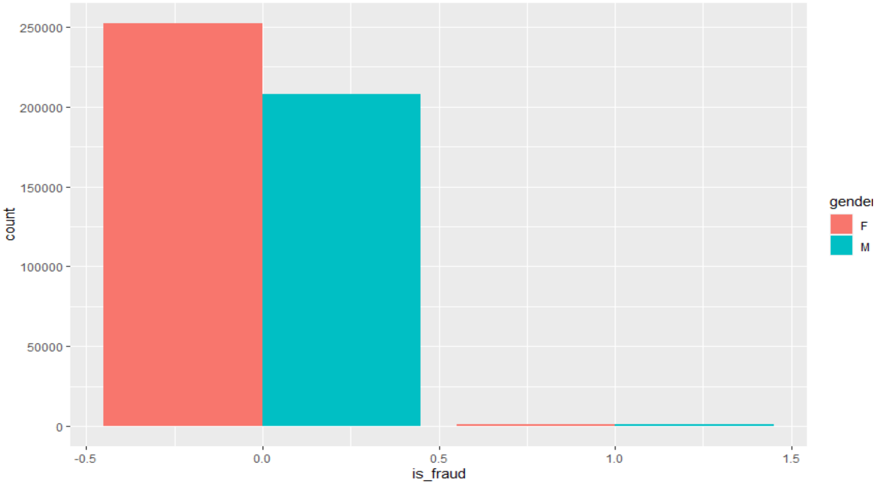
Further we would like to see the distribution of 'Amt' variable with that of 'fraud' transactions only, refer below bar graph:




```
unsampled %>%
  filter(is_fraud == "1") %>%
  ggplot(aes(x=is_fraud, y=amt,group=is_fraud))+
  geom_boxplot()+
  ggtitle("Amt by is_fraud")
```

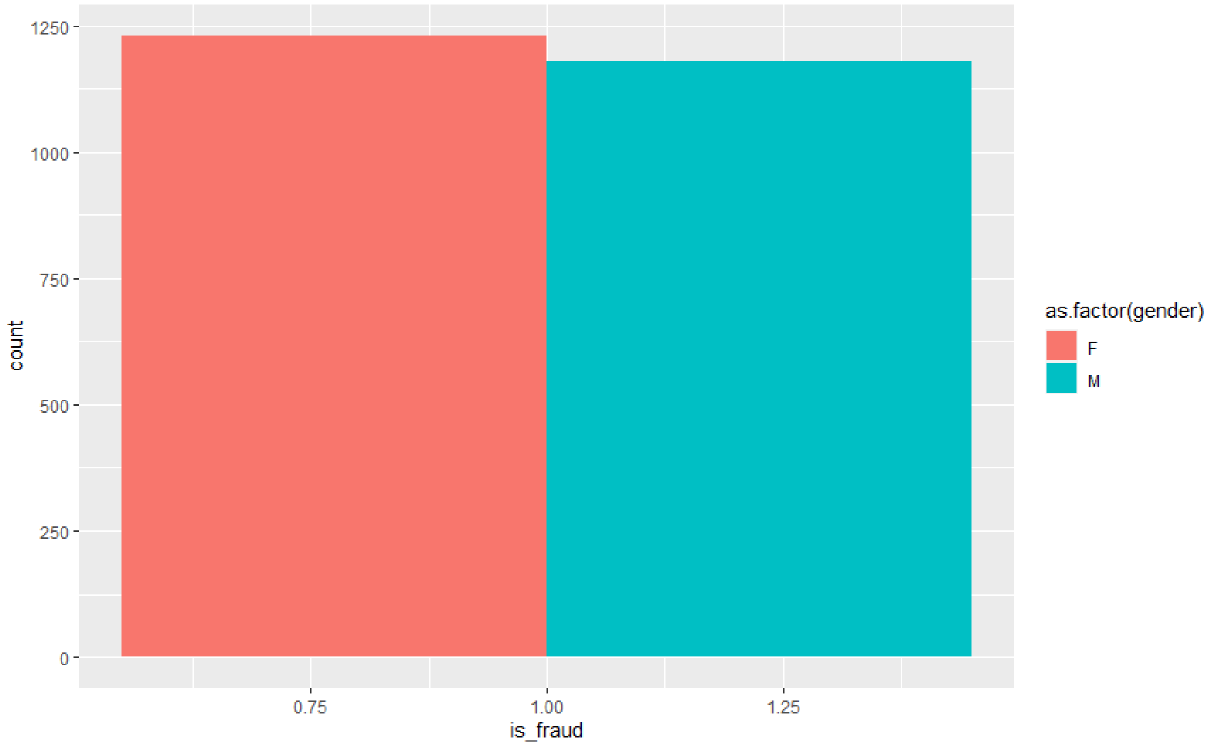
From the above diagram, it can be inferred that the amount is spread between 259 to approx. 800 with mean at 500.

vi. Distribution of variable ‘gender’ by ‘is_fraud’




```
ggplot(unsampled, aes(is_fraud)) + geom_bar(aes(fill = gender), position = "dodge")
```

Further carving put only fraud transactions on gender basis to know the only fraud distribution basis male and female:

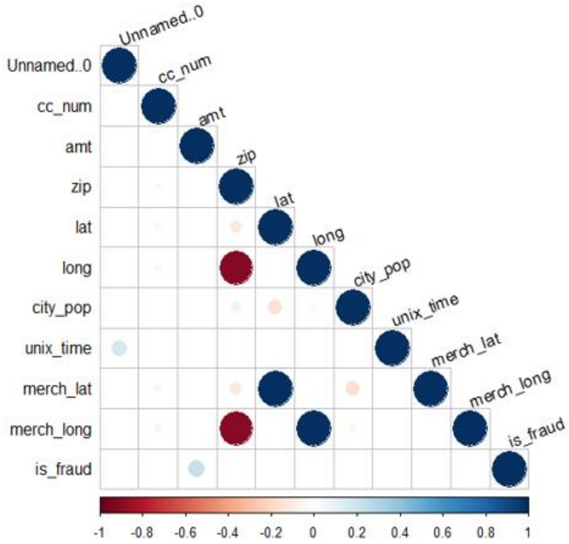


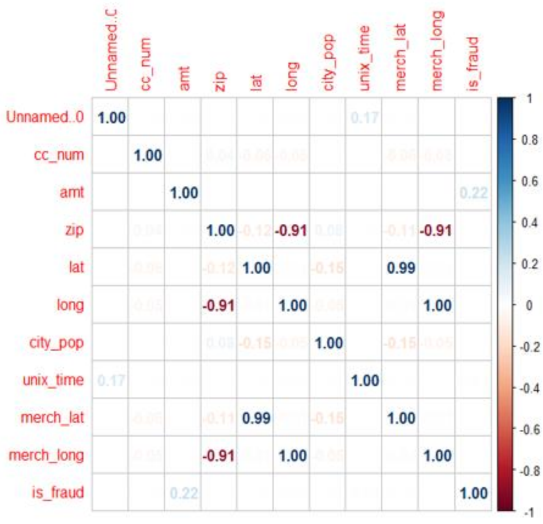

```
unsampled %>%
  filter(is_fraud == "1") %>%
  ggplot(aes(x=is_fraud)) + geom_bar(aes(fill = as.factor(gender)), position = "dodge")
```

It can be drawn that females contribute more into fraud transactions.

vii. Correlation Metrix- performed on unsampled data

```
corelation<-cor(unsampled[,sapply(unsampled,is.numeric)],use="complete.obs",method="pearson")
library(corrplot)
corrplot(corelation, type = "lower", tl.col = 'black',  tl.srt = 20)
corrplot(corelation, method = 'number')
```





<u>Inference</u>- It can be inferred that there is no correlation between the variables, only the response (is_fraud) variable seems to have positively correlated with 'amt' feature which seems significant.

Secondly, it can also be drawn that a few variables are highly correlated with each other like 'merch_long' is negatively related 'zip' and positively correlated with 'long' and likewise 'long' with 'zip' and 'merch_lat' with 'lat'. There could be a multicollinearity among these features which can influence the model performance in an unfavourable way.

<u>Descriptive Analysis:</u>

```
> summary(unsampled)
   Unnamed..0      trans_date_trans_time      cc_num               merchant            category              amt
 Min.   :      0   Length:463099          Min.   :6.042e+10   Length:463099      Length:463099      Min.   :    1.00
 1st Qu.: 231382   Class :character       1st Qu.:1.800e+14   Class :character   Class :character   1st Qu.:    9.64
 Median : 462977   Mode  :character       Median :3.521e+15   Mode  :character   Mode  :character   Median :   47.43
 Mean   : 536737                          Mean   :4.188e+17                                         Mean   :   69.90
 3rd Qu.: 830821                          3rd Qu.:4.642e+15                                         3rd Qu.:   83.06
 Max.   :1296665                          Max.   :4.992e+18                                         Max.   :19364.91
    first               last               gender              street               city               state
 Length:463099      Length:463099      Length:463099      Length:463099      Length:463099      Length:463099
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character


      zip             lat             long            city_pop            job                dob
 Min.   : 1257   Min.   :20.03   Min.   :-165.67   Min.   :     23   Length:463099      Length:463099
 1st Qu.:26237   1st Qu.:34.67   1st Qu.: -96.80   1st Qu.:    743   Class :character   Class :character
 Median :48174   Median :39.37   Median : -87.46   Median :   2456   Mode  :character   Mode  :character
 Mean   :48799   Mean   :38.54   Mean   : -90.22   Mean   :  88578
 3rd Qu.:72011   3rd Qu.:41.94   3rd Qu.: -80.16   3rd Qu.:  20328
 Max.   :99921   Max.   :66.69   Max.   : -67.95   Max.   :2906700
  trans_num           unix_time            merch_lat       merch_long          is_fraud
 Length:463099      Min.   :1.325e+09   Min.   :19.03   Min.   :-166.67   Min.   :0.000000
 Class :character   1st Qu.:1.343e+09   1st Qu.:34.74   1st Qu.: -96.89   1st Qu.:0.000000
 Mode  :character   Median :1.357e+09   Median :39.37   Median : -87.42   Median :0.000000
                    Mean   :1.359e+09   Mean   :38.54   Mean   : -90.22   Mean   :0.005202
                    3rd Qu.:1.375e+09   3rd Qu.:41.96   3rd Qu.: -80.24   3rd Qu.:0.000000
                    Max.   :1.389e+09   Max.   :67.51   Max.   : -66.95   Max.   :1.000000
> |
```

Data Pre-Processing for Model Building:

From the dataset and the description, our aim is to utilise optimum variables to predict response variable; 'is_fraud', which has binary response, and it can be observed that there are 3 types of data in our dataset: numerical, date and categorical. Therefore, we should do data processing before to establish a machine learning model.

i. Check Missing Values:
As checked earlier, there are no missing or NULL in the dataset

ii. Check for Duplicate Rows: no duplicate rows in the dataset

```
> #finding duplicate rows
> nrow(unsampled[duplicated(unsampled), ])
[1] 0
```

iii. Categorical data
For categorical data, the variables are to be converted to factor to do dummy coding.
Following variables have been converted to factor and below is the R script:
Is_fraud, zip, state, gender ,category and trans_hour

```
> ###Converting the categorical variables into Factor
> unsampled$is_fraud <- as.factor(unsampled$is_fraud)
> unsampled$zip <- as.factor(unsampled$zip)
> unsampled$state <- as.factor(unsampled$state)
> unsampled$gender <- as.factor(unsampled$gender)
> unsampled$category <- as.factor(unsampled$category)
> unsampled$trans_hour<-as.factor(unsampled$trans_hour)
```

iv. Numerical data
For numeric data, scaling is used to standardize the independent features in a dataset that contains continuous features that are on different scales, following features have been scaled comparatively: **amt, lat,long,merch_lat,merch_long and city_pop**

```
> #Scaling numeric variables
> unsampled$amt <- scale(unsampled$amt)
> unsampled$merch_lat <- scale(unsampled$merch_lat)
> unsampled$merch_long <- scale(unsampled$merch_long)
> unsampled$lat <- scale(unsampled$lat)
> unsampled$long <- scale(unsampled$long)
> unsampled$city_pop <- scale(unsampled$city_pop)
```

v. Encoding variables
'gender' variable is encoded as M= 1 and F=0 to make this variable more compatible for model building

```
> unsampled$gender=ifelse(unsampled$gender=="M",1,0)
```

vi. Creating new features from date column for better analysis

From ‘trans_date_trans_time’ a new variable is extracted namely ‘trans_month’ to be able to analyse the data from month perspective

```
> unsampled$trans_hour=format(as.POSIXct(unsampled$trans_date_trans_time), format = "%H")
```

vii. Eliminating unnecessary variables/features

Below mentioned features have been kept for model building and everything else has been dropped.

"category", "amt", "gender", "state", "zip", “lat”,”long”,”city_pop", "merch_lat", "merch_long" "is_fraud", "trans_hour",”age”

```
> data_cc=subset(unsampled,select=c(5,6,9,12,14,15,16,21,22,23,26,29))
> colnames(data_cc)
 [1] "category"   "amt"        "gender"     "state"      "lat"        "long"       "city_pop"  
 [8] "merch_lat"  "merch_long" "is_fraud"   "trans_hour" "age"
```

viii. Understanding the imbalance

Handling Imbalance data- it is clear from initial data understanding and EDA exercises that our data is highly imbalanced, so we need to use sampling techniques (please refer next section for sampling in detail) to deal with imbalance, below R code and its output shows the imbalance problem in data; 99.5% no_fraud and .05% fraud:

```
> ################  measure of imbalance###############
> table(data_cc$is_fraud)

     0      1 
460690   2409 
> # class imbalance in percentage
> prop.table(table(data_cc$is_fraud))

          0           1 
0.994798089 0.005201911 
```

ix. Splitting into train and test

```
> #####splitting Train and Test
> library(caTools)
> set.seed(123)
> split = sample.split(data_cc$is_fraud,SplitRatio=0.80)
> train_data = subset(data_cc,split==TRUE)
> test_data = subset(data_cc,split==FALSE)
> dim(train_data)
[1] 370479     12
> dim(test_data)
[1] 92620    12
```

Sampling technique to tackle with imbalance data:

Since our data is highly imbalanced and ML algorithms struggle with accuracy because of the unequal distribution in dependent variable. This causes the performance of existing classifiers to get biased towards majority class. The methods to deal with this problem are widely known as **'Sampling Methods'**.

These methods aim to modify an imbalanced data into balanced distribution using some mechanism. The modification occurs by altering the size of original data set and provide the same proportion of balance.

Below are some available methods used to treat the imbalanced dataset:

1. Undersampling, 2. Oversampling, 3. Both (Over & under) 3. ROSE

"under" determines simple undersampling without replacement of the majority class until either the specified sample size N is reached or the positive examples has probability p of occurring. It then turns out that when method = "under", a sample of reduced size is returned

Option "over" determines simple oversampling with replacement from the minority class until either the specified sample size N is reached or the positive examples have probability p of occurrence

When method = "both" is selected, both the minority class is oversampled with replacement and the majority class is undersampled without replacement. In this case, both the arguments N and p have to be set to establish the amount of oversampling and undersampling. Essentially, the minority class is oversampled to reach a size determined as a realization of a binomial random variable with size N and probability p. Undersampling is then performed accordingly, to abide by the specified N.

ROSE (Random Over-Sampling Examples) aids the task of binary classification in the presence of rare classes. It produces a synthetic, possibly balanced, sample of data simulated according to a smoothed-bootstrap approach. Therefore, the combination of over & under-sampling will be used to process the dataset before modelling. By doing that, the proportion of fraud transaction in training set would be about 50%, (refer below R code and its output):

Over & Under Sampling method: below is the snippet of R code before and after along with output

- Before Sampling the proportion is as below:

```
> table(data_cc$is_fraud)

     0      1
460690   2409
```

- After Sampling with ROSE, the proportion is as below:

```
> fraction_fraud_new <- 0.5
> traindata_cc <- ROSE::ovun.sample(is_fraud~., train_data, method = "both", p= fraction_fraud_new, seed = 1234)$data
> table(traindata_cc$is_fraud)

     0      1
185491 184988
```

Model Building and Evaluation

1. **Full Logistic Regression Model** on Train data using R:

```
fit.lm=glm(is_fraud~., data = traindata_cc, family = binomial)
summary(fit.lm)
```

1.1. Output of the logistic regression model:

```
> fit.lm=glm(is_fraud~., data = traindata_cc, family = binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(fit.lm)

Call:
glm(formula = is_fraud ~ ., family = binomial, data = traindata_cc)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-8.4904  -0.3564  -0.0248   0.2997   2.6682

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -1.5272264  0.2899938  -5.266 1.39e-07 ***
categoryfood_dining    1.4522281  0.0401629  36.158  < 2e-16 ***
categorygas_transport  3.4888943  0.0407400  85.638  < 2e-16 ***
categorygrocery_net    2.9057225  0.0468505  62.021  < 2e-16 ***
categorygrocery_pos    2.5033999  0.0379861  65.903  < 2e-16 ***
categoryhealth_fitness 1.6889262  0.0405243  41.677  < 2e-16 ***
categoryhome           0.2262875  0.0417072   5.426 5.78e-08 ***
categorykids_pets      1.7108864  0.0392154  43.628  < 2e-16 ***
categorymisc_net      -0.5014091  0.0539373  -9.296  < 2e-16 ***
categorymisc_pos       2.1130152  0.0405197  52.148  < 2e-16 ***
categorypersonal_care  1.9951950  0.0398137  50.113  < 2e-16 ***
categoryshopping_net  -1.8054212  0.0580447 -31.104  < 2e-16 ***
categoryshopping_pos  -1.6248262  0.0524714 -30.966  < 2e-16 ***
categorytravel         2.2226702  0.0457218  48.613  < 2e-16 ***
amt                    1.5655800  0.0082819 189.037  < 2e-16 ***
gender1               -0.0248163  0.0122702  -2.022 0.043126 *
stateAL               -1.2753905  0.3029689  -4.210 2.56e-05 ***
stateAR               -0.8270135  0.2862124  -2.890 0.003858 **
stateAZ               -1.8016423  0.2511544  -7.173 7.31e-13 ***
stateCA               -1.1435266  0.2211443  -5.171 2.33e-07 ***
stateCO               -1.0863688  0.2537666  -4.281 1.86e-05 ***
stateCT               -0.6660060  0.3408243  -1.954 0.050689 .
stateDC               -0.3677205  0.3384792  -1.086 0.277306
stateDE               11.8783828 23.3286587   0.509 0.610629
stateFL               -1.3419391  0.3224059  -4.162 3.15e-05 ***
stateGA               -0.6862896  0.3138080  -2.187 0.028745 *
stateHI               -3.8404935  0.3263577 -11.768  < 2e-16 ***
stateIA               -0.8587321  0.2760370  -3.111 0.001865 **
stateID               -0.5249562  0.2332525  -2.251 0.024411 *
stateIL               -0.7926631  0.2882446  -2.750 0.005960 **
stateIN               -1.0399222  0.2970241  -3.501 0.000463 ***
stateKS               -0.5967428  0.2690443  -2.218 0.026554 *
stateKY               -1.0624156  0.3035372  -3.500 0.000465 ***
stateLA               -1.6251414  0.2979886  -5.454 4.93e-08 ***
stateMA               -1.4568956  0.3430034  -4.247 2.16e-05 ***
stateMD               -0.9752920  0.3263674  -2.988 0.002805 **
stateME               -0.9644312  0.3463126  -2.785 0.005355 **
stateMI               -1.0482920  0.2994935  -3.500 0.000465 ***
stateMN               -0.3879988  0.2705364  -1.434 0.151520
```

```
stateMO               -0.8442847  0.2803075  -3.012 0.002595 **
stateMS               -1.0091579  0.2963076  -3.406 0.000660 ***
stateMT               -0.5017898  0.2304791  -2.177 0.029469 *
stateNC               -0.8983953  0.3194497  -2.812 0.004919 **
stateND               -1.1078776  0.2601392  -4.259 2.06e-05 ***
stateNE               -1.1282381  0.2644447  -4.266 1.99e-05 ***
stateNH               -0.0544527  0.3426382  -0.159 0.873730
stateNJ               -1.4955793  0.3330227  -4.491 7.09e-06 ***
stateNM               -1.0285330  0.2555365  -4.025 5.70e-05 ***
stateNV               -0.8671943  0.2324066  -3.731 0.000190 ***
stateNY               -0.6800690  0.3280795  -2.073 0.038184 *
stateOH               -0.9955641  0.3065133  -3.248 0.001162 **
stateOK               -0.6727124  0.2733801  -2.461 0.013866 *
stateOR               -0.9389953  0.2072242  -4.531 5.86e-06 ***
statePA               -0.7132513  0.3186434  -2.238 0.025195 *
stateRI               -0.5503868  0.4244313  -1.297 0.194713
stateSC               -0.6408924  0.3174825  -2.019 0.043522 *
stateSD               -1.0227977  0.2613820  -3.913 9.11e-05 ***
stateTN               -1.2242217  0.3033107  -4.036 5.43e-05 ***
stateTX               -1.3300120  0.2754967  -4.828 1.38e-06 ***
stateUT               -0.8697106  0.2400270  -3.623 0.000291 ***
stateVA               -0.1724348  0.3215771  -0.536 0.591809
stateVT               -2.6023972  0.3445474  -7.553 4.25e-14 ***
stateWA               -0.9289237  0.2074602  -4.478 7.55e-06 ***
stateWI               -0.4945236  0.2850833  -1.735 0.082800 .
stateWV               -1.2764774  0.3160847  -4.038 5.38e-05 ***
stateWY               -0.9145767  0.2402796  -3.806 0.000141 ***
lat                   -0.0897619  0.0569396  -1.576 0.114924
long                  -0.3436660  0.1478366  -2.325 0.020091 *
city_pop              -0.0295718  0.0063649  -4.646 3.38e-06 ***
merch_lat             -0.1001268  0.0510449  -1.962 0.049816 *
merch_long             0.3354368  0.1388415   2.416 0.015693 *
trans_hour1           -0.2734284  0.0278716  -9.810  < 2e-16 ***
trans_hour2           -0.2381742  0.0278893  -8.540  < 2e-16 ***
trans_hour3           -0.2766678  0.0282252  -9.802  < 2e-16 ***
trans_hour4           -2.4728747  0.0426571 -57.971  < 2e-16 ***
trans_hour5           -2.5297365  0.0417069 -60.655  < 2e-16 ***
trans_hour6           -2.4041724  0.0413054 -58.205  < 2e-16 ***
trans_hour7           -2.8329314  0.0462264 -61.284  < 2e-16 ***
trans_hour8           -2.7713897  0.0453867 -61.062  < 2e-16 ***
trans_hour9           -2.3507568  0.0407650 -57.666  < 2e-16 ***
trans_hour10          -2.9035352  0.0484795 -59.892  < 2e-16 ***
trans_hour11          -2.7990479  0.0479725 -58.347  < 2e-16 ***
trans_hour12          -2.5007704  0.0641788 -38.966  < 2e-16 ***
trans_hour13          -1.6465272  0.0491517 -33.499  < 2e-16 ***
trans_hour14          -1.8735435  0.0541720 -34.585  < 2e-16 ***
trans_hour15          -1.8058027  0.0526417 -34.304  < 2e-16 ***
trans_hour16          -1.7361304  0.0511258 -33.958  < 2e-16 ***
trans_hour17          -1.4373475  0.0475494 -30.228  < 2e-16 ***
trans_hour18          -2.0225491  0.0513629 -39.378  < 2e-16 ***
trans_hour19          -1.6932380  0.0514906 -32.884  < 2e-16 ***
trans_hour20          -1.9282875  0.0509188 -37.870  < 2e-16 ***
trans_hour21          -1.5513647  0.0465771 -33.307  < 2e-16 ***
trans_hour22           1.3788702  0.0336517  40.975  < 2e-16 ***
trans_hour23           1.3359367  0.0335251  39.849  < 2e-16 ***
age                    0.0026707  0.0003391   7.877 3.36e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 513592  on 370478  degrees of freedom
Residual deviance: 193636  on 370384  degrees of freedom
AIC: 193826

Number of Fisher Scoring iterations: 11
```

- Logistic regression model is fit on the sampled training data with ‘over and under-both’ sampling method
- The estimates, standard error and P values are given in the output table
- Low standard error and p value <0.05 indicates the significance of features
- Residual deviance (deviation of fitted model with saturated one) and AIC values (relative amount loss); lesser these values better is the model fit

1.2. Feature selection using AIC:

To check whether all the feature are relevant, we need to carry out a variable selection in logistic regression using the function StepAIC() under the library(Mass) in R. the function uses the lowest AIC value to select the best model by adjusting the number of predictors. Below is the output including all the relevant feature for best fit model: R code along with its output:

```
> ### Feature selection using AIC
> library(MASS)
> step=stepAIC(fit.lm)
Start:  AIC=193826.2
is_fraud ~ category + amt + gender + state + lat + long + city_pop +
    merch_lat + merch_long + trans_hour + age

             Df Deviance    AIC
<none>          193636 193826
- lat         1  193638 193826
- gender      1  193639 193827
- merch_lat   1  193640 193828
- long        1  193643 193831
- merch_long  1  193643 193831
- city_pop    1  193662 193850
- age         1  193694 193882
- state      50  196786 196876
- category   13  223678 223842
- trans_hour 23  261473 261617
- amt         1  303734 303922
```

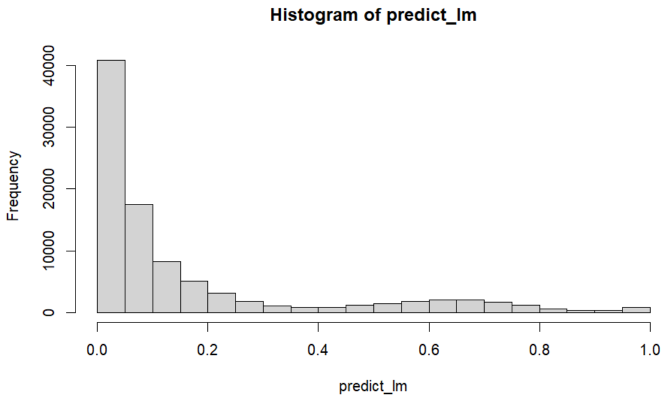
In the above StepAIC output, we can see that same features have been selected.

1.3. Predicting success probabilities using the LR model:

Trained model is used to predict the probability whether a loan would be approved or rejected. Code used is as follows:

```
> ### prediction using glm model on test data
> predict_lm <- predict(fit.lm, test_data, type = "response")
```

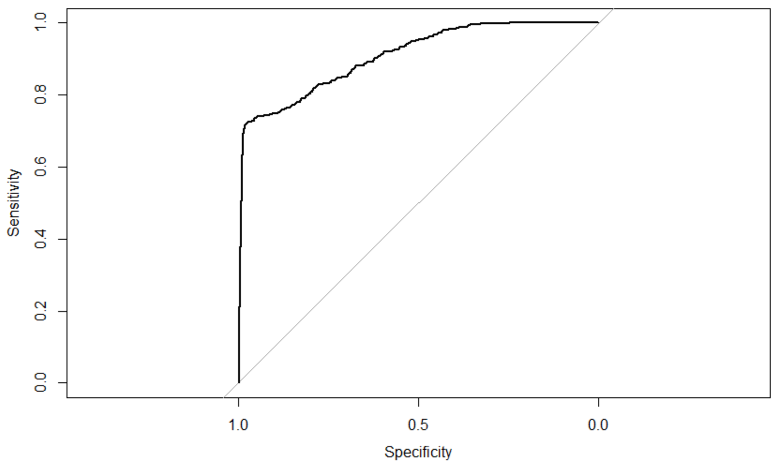
```
> hist(predict_lm)
>
```



1.4. ROC Curve and Threshold: AUC is 0.954

Plotting an ROC curve to set a threshold value, using that we can classify the observations into two categories of whether a transaction would be fraud or no fraud. R code along with output:

```
> roc1=roc(test_data[,10],predict_lm,plot=TRUE,legacy.axes=TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot(roc1)
> AUC_LM=roc1$auc
> AUC_LM
Area under the curve: 0.954
```



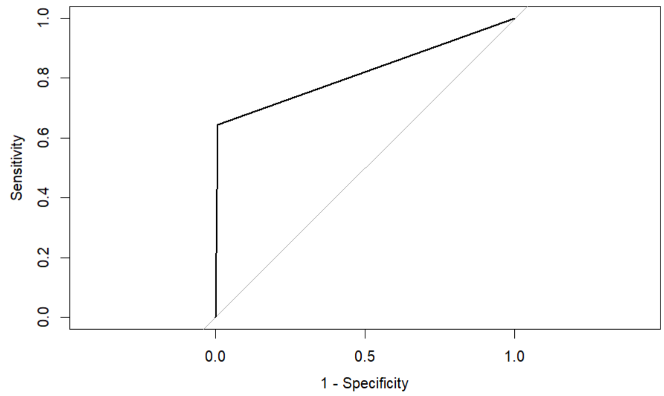
Findings:

- The ROC curve has an AUC value of .954, hence it is performing 40% better then the random classification model
- AUC greater should be greater than 0.8 for a good classification model and in our case, it is coming out to be .90 which is very good.

1.5 Evaluation of Logistic Regression Model

So far, the logistic regression model is built and in this segment its performance will be evaluated. Based on the standard threshold value of 0.50, let's compute the accuracy of the final model to evaluate its classification ability. The following output along with R code was obtained on evaluating the logistic regression model:

```
> pred_Y=ifelse(predict_lm > 0.9,1,0)

> roc=roc(test_data[,10],pred_Y,plot=TRUE,legacy.axes=TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```




```
> confusionMatrix(as.factor(test_data[,10]), as.factor(pred_Y))
```

Please refer below snippet for confusion matrix output:

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 91398   740
         1   171   311

               Accuracy : 0.9902
                 95% CI : (0.9895, 0.9908)
    No Information Rate : 0.9887
    P-Value [Acc > NIR] : 4.872e-06

                  Kappa : 0.4015

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9981
            Specificity : 0.2959
         Pos Pred Value : 0.9920
         Neg Pred Value : 0.6452
             Prevalence : 0.9887
         Detection Rate : 0.9868
   Detection Prevalence : 0.9948
      Balanced Accuracy : 0.6470

       'Positive' Class : 0
```

Inference:

- Accuracy is 99%, however it does not seem a good measure as the model is predicting majorly (740) fraud cases into no_fraud

2. Random Forest Fitting and Evaluation:

2.1 Random Forest Model building: R code along with output:

```
> rfboth<-randomForest(as.factor(is_fraud) ~ ., data=traindata_cc,ntree=500, mtry=6)
> rfboth

Call:
 randomForest(formula = as.factor(is_fraud) ~ ., data = traindata_cc,      ntree = 500,
y = 6)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 0.05%
Confusion matrix:
       0      1  class.error
0 185318    173 0.0009326598
1      0 184988 0.0000000000
```

2.2 Predicting probabilities on test data and Evaluation:

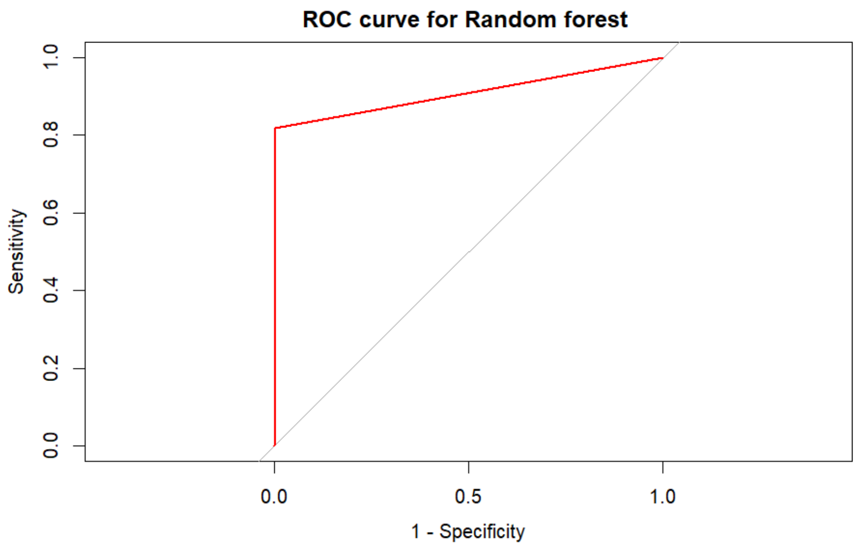
```
> library(caret)
> rf_predict <- predict(rfboth, test_data)
> rf_cm<- confusionMatrix(data = rf_predict, test_data$is_fraud)
> rf_cm
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 92003    87
         1   135   395

               Accuracy : 0.9976
                 95% CI : (0.9973, 0.9979)
    No Information Rate : 0.9948
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7794

 Mcnemar's Test P-Value : 0.001608

            Sensitivity : 0.9985
            Specificity : 0.8195
         Pos Pred Value : 0.9991
         Neg Pred Value : 0.7453
             Prevalence : 0.9948
         Detection Rate : 0.9933
   Detection Prevalence : 0.9943
      Balanced Accuracy : 0.9090

       'Positive' Class : 0
```

Inference:

- Accuracy is 99%, Sensitivity/Recall is 81.9%
- Seems Random Forest model is doing the great job in classifying or predicting the fraud and no_fraud
- Out of 482 cases of fraud, model has predicted correctly 395 which is a good ratio comparatively

2.3 ROC CURVE:

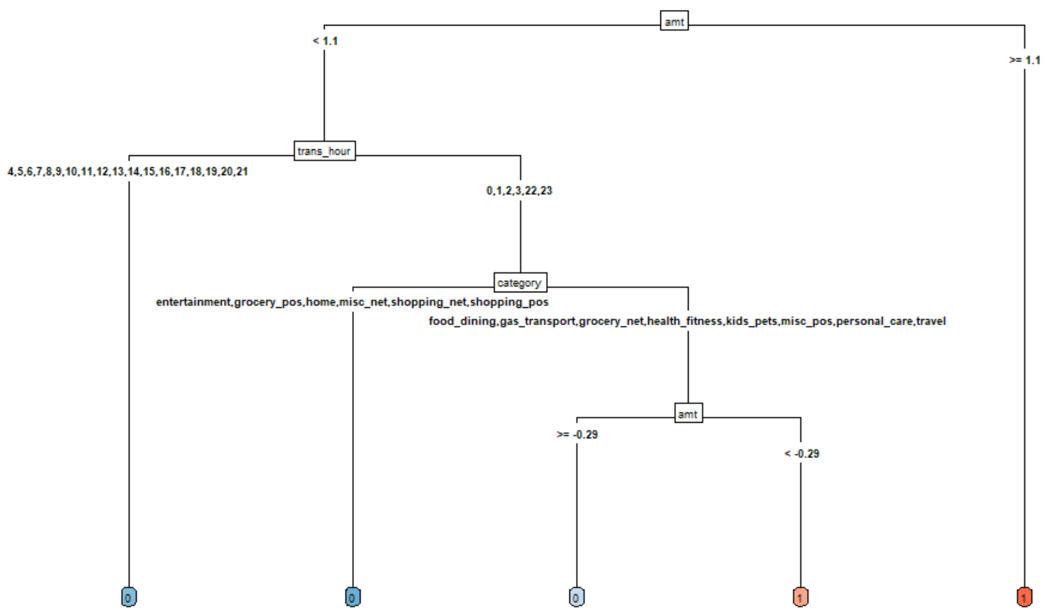
```
> roc_RF=roc((test_data[,10]),as.numeric(rf_predict),plot=TRUE,legacy.axes=TRUE,col='red'
ain = "ROC curve for Random forest ")
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> auc_RF=as.numeric(roc_RF$auc)
> auc_RF
[1] 0.9090184
```



3. Decision Tree Model Fitting and Evaluation:

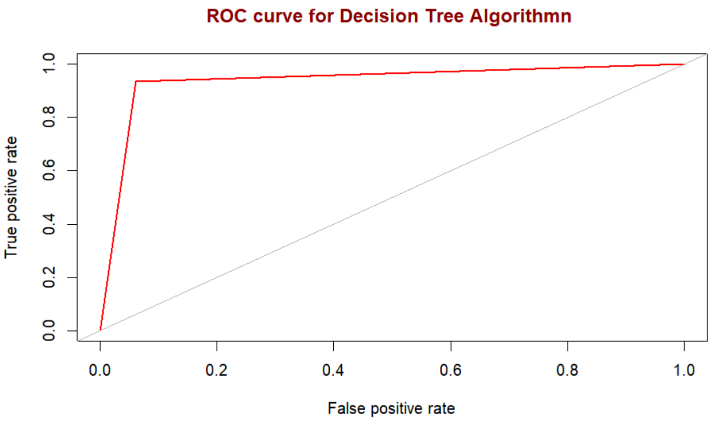
3.1. Model building: R script along with the output

```
> library(rpart)
> library(rpart.plot)
> mod_tree = rpart(is_fraud ~ .,data = traindata_cc,method = "class")
> rpart.plot(mod_tree,cex=0.45,extra = 0,type=5,box.palette = "BuRd")
```



3.2. Predicting probabilities on test data and ROC:

```
> pred_tree = predict(mod_tree,test_data,type="class")
> roc_tree=roc.curve(test_data$is_fraud,pred_tree,plotit = TRUE,
+                    col="red",main = "ROC curve for Decision Tree Algorithmn ",
+                    col.main="darkred")
> auc_tree=as.numeric(roc_tree$auc)
> auc_tree
[1] 0.9377299
```



3.3. Model Evaluation: below is the R code along with output

```
> confusionMatrix(pred_tree,test_data[,10])
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 86589    31
         1  5549   451

               Accuracy : 0.9398
                 95% CI : (0.9382, 0.9413)
    No Information Rate : 0.9948
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1308

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.93978
            Specificity : 0.93568
         Pos Pred Value : 0.99964
         Neg Pred Value : 0.07517
             Prevalence : 0.99480
         Detection Rate : 0.93488
   Detection Prevalence : 0.93522
      Balanced Accuracy : 0.93773

       'Positive' Class : 0
```

```
> precision_DT
[1] 0.07516667
> recall_DT
[1] 0.9356846
> f1_score_DT
[1] 0.1391546
> accuracy_DT
[1] 0.9397538
>
```

Summary: key insights

1. **EDA**
   - The fraud transactions occur more between 9 PM to 2 AM at night.
   - A large proportion of frauds occur in small amount transactions.

2. **Dealing with Imbalance problem:** in this project, the dataset is very imbalanced with 99.47% of cases being non-fraudulent transactions. So, to combat the imbalance problem, ROSE (Both- Under and Over sampling) method was used and the distribution came out to be normal. Project has demonstrated the importance of sampling effectively, modelling and predicting data with an imbalanced dataset

3. **Different ML Algorithms and Result Interpretation:**

   3.1. The modelling methods used are Logistic Regression, Random Forest and Decision Tree algorithms.

   3.2. Model Evaluation: all 3 models discussed above performed well with the sampled data including the prediction done with test and evaluated in distinct metrics. An appropriate measure of model performance here would be AUC (Area Under the Precision-Recall Curve).

   3.3. The Best Model or ML Algorithm:

   - The Logistic Regression model has high accuracy of 99% and obtained good AUC score of 95%, however, the model is still biased in predicting non fraudulent transactions than fraudulent ones.
   - While Decision Tree model has highest recall value of 93% but accuracy dropped to 93% because it categorised the 25% non-fraudulent transactions as fraudulent.
   - And, Random Forest algorithm has the highest recall value of 81%, accuracy 99% and AUC is 90%.

   - **Conclusion:** Random Forest is giving better prediction in the view of fraud and non-fraud transactions as out of 482 cases of fraud, model is able to predict 395 cases correctly as fraud and which is a good ratio comparative to other models we tried. Also, it is suitable in decreasing the loss occurred in fraudulent transactions (for more details, refer below CBA output).

4. **Cost & Benefit Analysis:** so, if we deploy Random Forest model to detect fraud transactions then the overall cost reduces to 96%, below table shows cost incurred before and after model on monthly basis:

Cost/Model deployment status	Before Model	After model
Monthly Cost Incurred	54508	2002