bank_fraud

Group Project

2023-11-09

1. Introducton

This project aims to identify fraudulent transactions with Credit Cards. Our objective is to build a Fraud detection system using Machine learning techniques. Each transaction is labelled either fraudulent or not fraudulent. Note that prevalence of fraudulent transactions is very low in the dataset, hence the dataset will be balanced using re-sampling techniques.

2. Data Understanding

The dataset contain 555719 transactions with 23 attributes that will be analyzed and then used to build a machine model for making predictions.

Dataset Breakdown: 23 attributes (22 predictive attributes and 1 goal field)

```
#importing library

suppressMessages({
library(caret)
library(tidyverse)
library(randomForest)
library(corrplot)
library(rpart)
library(kernlab)
library(pROC)
library(ROSE) #library for resampling
library(lubridate)
library(class)})

## Warning: package 'ROSE' was built under R version 4.3.2

#Load dataset
fraud_csv <- read.csv("fraud_dataset.csv")
head(fraud_csv)

##   X trans_date_trans_time       cc_num
merchant
## 1 0   2020-06-21 12:14:25 2.291164e+15                fraud_Kirlin and
Sons
## 2 1   2020-06-21 12:14:33 3.573030e+15                 fraud_Sporer-
Keebler
## 3 2   2020-06-21 12:14:53 3.598215e+15 fraud_Swaniawski, Nitzsche and
```

```
Welch
## 4 3   2020-06-21 12:15:15 3.591920e+15                   fraud_Haley
Group
## 5 4   2020-06-21 12:15:17 3.526826e+15                fraud_Johnston-
Casper
## 6 5   2020-06-21 12:15:37 3.040768e+13                  fraud_Daugherty
LLC
##         category   amt    first     last gender
street
## 1  personal_care  2.86     Jeff  Elliott      M           351 Darlene
Green
## 2  personal_care 29.84   Joanne Williams      F            3638 Marsh
Union
## 3 health_fitness 41.28   Ashley    Lopez      F        9333 Valentine
Point
## 4       misc_pos 60.05    Brian Williams      M 32941 Krystal Mill Apt.
552
## 5         travel  3.19   Nathan   Massey      M    5783 Evan Roads Apt.
465
## 6      kids_pets 19.55 Danielle    Evans      F  76752 David Lodge Apt.
064
##         city state   zip     lat      long city_pop                    job
## 1   Columbia    SC 29209 33.9659  -80.9355   333497    Mechanical engineer
## 2    Altonah    UT 84002 40.3207 -110.4360      302 Sales professional, IT
## 3   Bellmore    NY 11710 40.6729  -73.5365    34496      Librarian, public
## 4 Titusville    FL 32780 28.5697  -80.8191    54767           Set designer
## 5   Falmouth    MI 49632 44.2529  -85.0170     1126     Furniture designer
## 6  Breesport    NY 14816 42.1939  -76.7361      520        Psychotherapist
##         dob                        trans_num  unix_time merch_lat
merch_long
## 1 1968-03-19 2da90c7d74bd46a0caf3777415b3ebd3 1371816865  33.98639  -
81.20071
## 2 1990-01-17 324cc204407e99f51b0d6ca0055005e7 1371816873  39.45050 -
109.96043
## 3 1970-10-21 c81755dbbbea9d5c77f094348a7579be 1371816893  40.49581  -
74.19611
## 4 1987-07-25 2159175b9efe66dc301f149d3d5abf8c 1371816915  28.81240  -
80.88306
## 5 1955-07-06 57ff021bd3f328f8738bb535c302a31b 1371816917  44.95915  -
85.88473
## 6 1991-10-13 798db04aaceb4febd084f1a7c404da93 1371816937  41.74716  -
77.58420
##   is_fraud
## 1        0
## 2        0
## 3        0
## 4        0
## 5        0
## 6        0
```

```r
#checking initial structure and datatypes in dataset
str(fraud_csv)
```

```
## 'data.frame':    555719 obs. of  23 variables:
##  $ X                    : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ trans_date_trans_time: chr  "2020-06-21 12:14:25" "2020-06-21 12:14:33"
"2020-06-21 12:14:53" "2020-06-21 12:15:15" ...
##  $ cc_num               : num  2.29e+15 3.57e+15 3.60e+15 3.59e+15
3.53e+15 ...
##  $ merchant             : chr  "fraud_Kirlin and Sons" "fraud_Sporer-
Keebler" "fraud_Swaniawski, Nitzsche and Welch" "fraud_Haley Group" ...
##  $ category             : chr  "personal_care" "personal_care"
"health_fitness" "misc_pos" ...
##  $ amt                  : num  2.86 29.84 41.28 60.05 3.19 ...
##  $ first                : chr  "Jeff" "Joanne" "Ashley" "Brian" ...
##  $ last                 : chr  "Elliott" "Williams" "Lopez" "Williams" ...
##  $ gender               : chr  "M" "F" "F" "M" ...
##  $ street               : chr  "351 Darlene Green" "3638 Marsh Union"
"9333 Valentine Point" "32941 Krystal Mill Apt. 552" ...
##  $ city                 : chr  "Columbia" "Altonah" "Bellmore"
"Titusville" ...
##  $ state                : chr  "SC" "UT" "NY" "FL" ...
##  $ zip                  : int  29209 84002 11710 32780 49632 14816 95528
57374 16858 76678 ...
##  $ lat                  : num  34 40.3 40.7 28.6 44.3 ...
##  $ long                 : num  -80.9 -110.4 -73.5 -80.8 -85 ...
##  $ city_pop             : int  333497 302 34496 54767 1126 520 1139 343
3688 263 ...
##  $ job                  : chr  "Mechanical engineer" "Sales professional,
IT" "Librarian, public" "Set designer" ...
##  $ dob                  : chr  "1968-03-19" "1990-01-17" "1970-10-21"
"1987-07-25" ...
##  $ trans_num            : chr  "2da90c7d74bd46a0caf3777415b3ebd3"
"324cc204407e99f51b0d6ca0055005e7" "c81755dbbbea9d5c77f094348a7579be"
"2159175b9efe66dc301f149d3d5abf8c" ...
##  $ unix_time            : int  1371816865 1371816873 1371816893 1371816915
1371816917 1371816937 1371816944 1371816950 1371816970 1371816971 ...
##  $ merch_lat            : num  34 39.5 40.5 28.8 45 ...
##  $ merch_long           : num  -81.2 -110 -74.2 -80.9 -85.9 ...
##  $ is_fraud             : int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
#checking for cardinality of columns and also for null values
fraud_csv %>% summarise_all(n_distinct)
```

```
##        X trans_date_trans_time cc_num merchant category   amt first last
gender
## 1 555719                544760    924      693       14 37256   341  471
2
##   street city state zip lat long city_pop job dob trans_num unix_time
merch_lat
```

```
## 1    924  849    50 912 910  910      835 478 910    555719    544760
546490
##   merch_long is_fraud
## 1     551770        2
```

```r
sum(is.na.data.frame(fraud_csv))
```

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

Converting features to appropriate datatypes

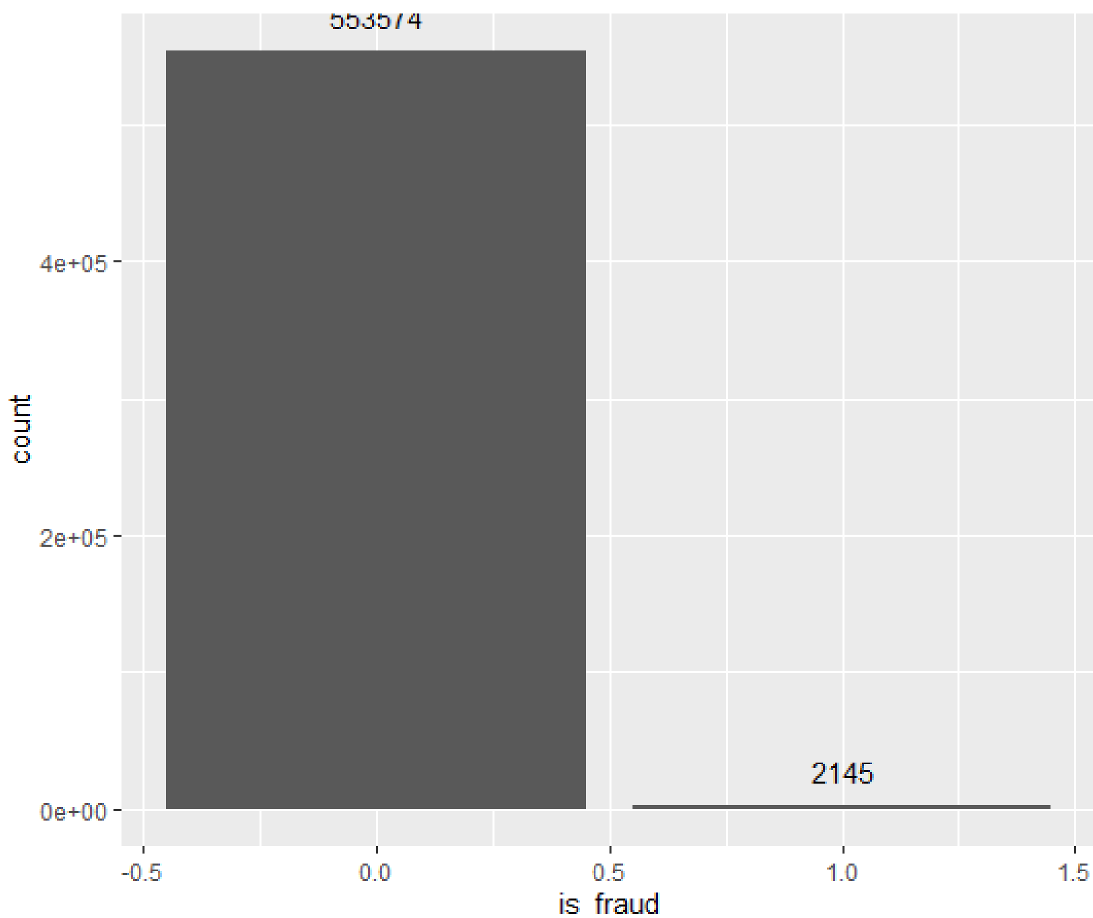
```r
#convert time to posix
fraud_csv$datetime <- as.POSIXct(fraud_csv$trans_date_trans_time, format="%Y-
%m-%d %H:%M:%S")

#extract dates
fraud_csv$date <- as.Date(fraud_csv$datetime)
#extract age from dob
fraud_csv$dob <- as.Date(fraud_csv$dob)
fraud_csv$age_2022 <- round(as.numeric(difftime(as.Date("2022-01-01"),
fraud_csv$dob, units = "days")) / 365.25, 0)

#convert dates +times into day, month and hour
fraud_csv$time <- format(fraud_csv$datetime, format="%H:%M:%S")
fraud_csv$hour <- as.numeric(hour(fraud_csv$datetime))
fraud_csv$weekday <- weekdays(as.Date(fraud_csv$datetime))
fraud_csv$weekday2 <-  as.numeric(format(fraud_csv$datetime, format = "%u"))
fraud_csv$month<- as.numeric(month(fraud_csv$datetime))

#convert gender to int
fraud_csv$gender <- ifelse(fraud_csv$gender=="M", 1, 0)

#concatenation of name
fraud_csv$full_name <- paste(fraud_csv$first, fraud_csv$last, sep = " ")

#checking new structure of dataset with new features created
str(fraud_csv)
```

```
## 'data.frame':    555719 obs. of  32 variables:
##  $ X                    : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ trans_date_trans_time: chr  "2020-06-21 12:14:25" "2020-06-21 12:14:33"
"2020-06-21 12:14:53" "2020-06-21 12:15:15" ...
##  $ cc_num               : num  2.29e+15 3.57e+15 3.60e+15 3.59e+15
3.53e+15 ...
##  $ merchant             : chr  "fraud_Kirlin and Sons" "fraud_Sporer-
Keebler" "fraud_Swaniawski, Nitzsche and Welch" "fraud_Haley Group" ...
##  $ category             : chr  "personal_care" "personal_care"
"health_fitness" "misc_pos" ...
##  $ amt                  : num  2.86 29.84 41.28 60.05 3.19 ...
##  $ first                : chr  "Jeff" "Joanne" "Ashley" "Brian" ...
##  $ last                 : chr  "Elliott" "Williams" "Lopez" "Williams" ...
```

```
##  $ gender                 : num  1 0 0 1 1 0 0 0 1 0 ...
##  $ street                 : chr  "351 Darlene Green" "3638 Marsh Union"
"9333 Valentine Point" "32941 Krystal Mill Apt. 552" ...
##  $ city                   : chr  "Columbia" "Altonah" "Bellmore"
"Titusville" ...
##  $ state                  : chr  "SC" "UT" "NY" "FL" ...
##  $ zip                    : int  29209 84002 11710 32780 49632 14816 95528
57374 16858 76678 ...
##  $ lat                    : num  34 40.3 40.7 28.6 44.3 ...
##  $ long                   : num  -80.9 -110.4 -73.5 -80.8 -85 ...
##  $ city_pop               : int  333497 302 34496 54767 1126 520 1139 343
3688 263 ...
##  $ job                    : chr  "Mechanical engineer" "Sales professional,
IT" "Librarian, public" "Set designer" ...
##  $ dob                    : Date, format: "1968-03-19" "1990-01-17" ...
##  $ trans_num              : chr  "2da90c7d74bd46a0caf3777415b3ebd3"
"324cc204407e99f51b0d6ca0055005e7" "c81755dbbbea9d5c77f094348a7579be"
"2159175b9efe66dc301f149d3d5abf8c" ...
##  $ unix_time              : int  1371816865 1371816873 1371816893 1371816915
1371816917 1371816937 1371816944 1371816950 1371816970 1371816971 ...
##  $ merch_lat              : num  34 39.5 40.5 28.8 45 ...
##  $ merch_long             : num  -81.2 -110 -74.2 -80.9 -85.9 ...
##  $ is_fraud               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ datetime               : POSIXct, format: "2020-06-21 12:14:25" "2020-06-
21 12:14:33" ...
##  $ date                   : Date, format: "2020-06-21" "2020-06-21" ...
##  $ age_2022               : num  54 32 51 34 66 30 71 50 49 66 ...
##  $ time                   : chr  "12:14:25" "12:14:33" "12:14:53" "12:15:15"
...
##  $ hour                   : num  12 12 12 12 12 12 12 12 12 12 ...
##  $ weekday                : chr  "Sunday" "Sunday" "Sunday" "Sunday" ...
##  $ weekday2               : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ month                  : num  6 6 6 6 6 6 6 6 6 6 ...
##  $ full_name              : chr  "Jeff Elliott" "Joanne Williams" "Ashley
Lopez" "Brian Williams" ...

write.csv(fraud_csv, file = "fraud_csv_new.csv")

#counting number of occurences of fraud(1) and no_fraud(0)
table(fraud_csv$is_fraud)

## 
##      0      1 
## 553574   2145

prop.table(table(fraud_csv$is_fraud))

## 
##           0           1 
## 0.996140136 0.003859864
```

```r
# visualizing data imbalance
ggplot(fraud_csv, aes(x = is_fraud)) +
geom_bar() +
geom_text(stat='count', aes(label=..count..), vjust=-1)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2
3.4.0.
## ℹ Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Extrapolating cost of Fraud (incidents and monetary value) for the whole year

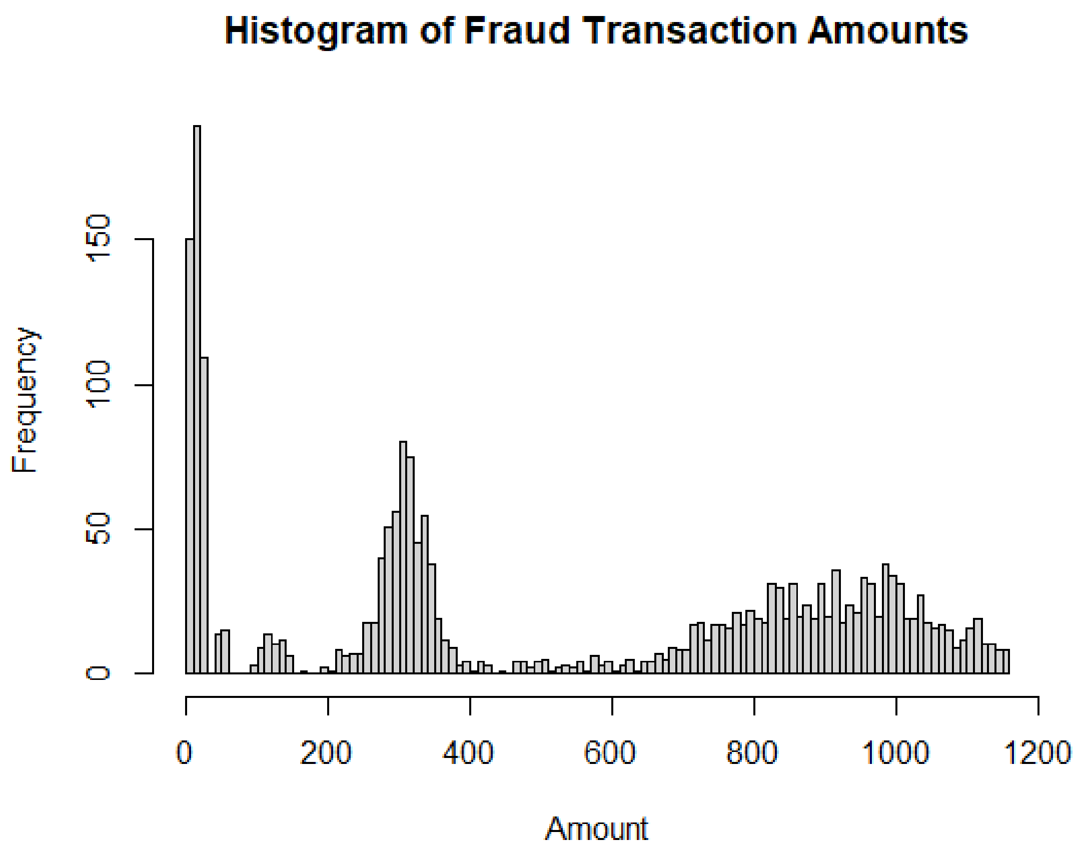
```r
#days_in_dataset: 194

days_in_dataset <- 194

# Calculate expected number of fraud cases per year
fraud_per_day <- sum(fraud_csv$is_fraud) / days_in_dataset
fraud_per_year <- fraud_per_day * 365
```

```
fraud_per_year_rounded <- round(fraud_per_year, 0)
fraud_per_year_rounded

## [1] 4036

# Calculate average non-fraudulent amount per day
nf_amount_df <- subset(fraud_csv, is_fraud == 0)
nf_amount_per_day <- sum(nf_amount_df$amt) / days_in_dataset
nf_amount_per_day_rounded <- round(nf_amount_per_day, 2)
paste(nf_amount_per_day_rounded, "$")

## [1] "192935.97 $"

# Calculate average fraudulent amount per day
fraud_amount_df <- subset(fraud_csv, is_fraud == 1)
fraud_amount_per_day <- sum(fraud_amount_df$amt) / days_in_dataset
fraud_amount_per_day_rounded <- round(fraud_amount_per_day, 2)
paste(fraud_amount_per_day_rounded, "$")

## [1] "5841.88 $"

# Estimate total fraudulent amount per year
total_fraud_yearly <- round(fraud_amount_per_day * 365, 2)
paste(total_fraud_yearly, "$")

## [1] "2132286.12 $"

# Estimate total non-fraudulent amount per year
total_non_fraud_yearly <- round(nf_amount_per_day * 365, 2)
paste(total_non_fraud_yearly, "$")

## [1] "70421629.52 $"
```

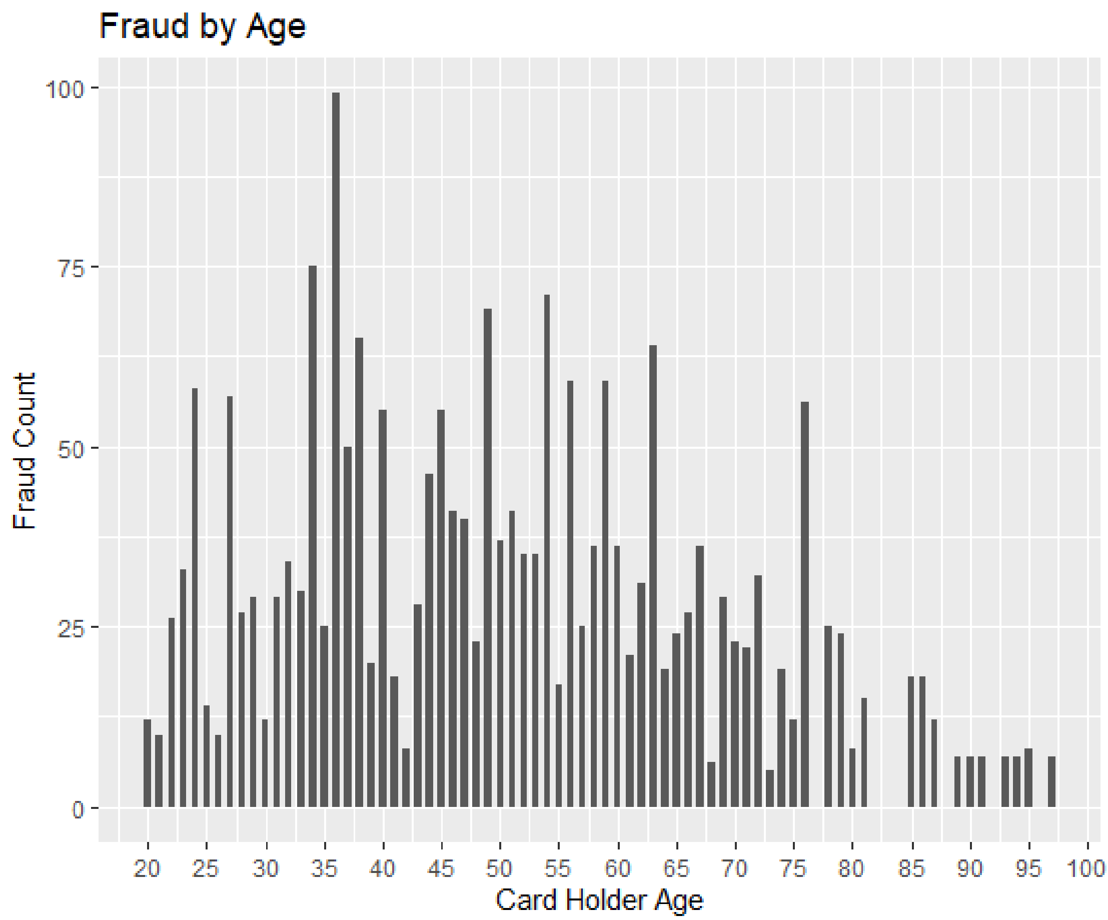
3. Exploratory Data Analysis

Before running machine learning models, features will be explored to see if there is any trends to point to prevelance of fraudulent transactions.

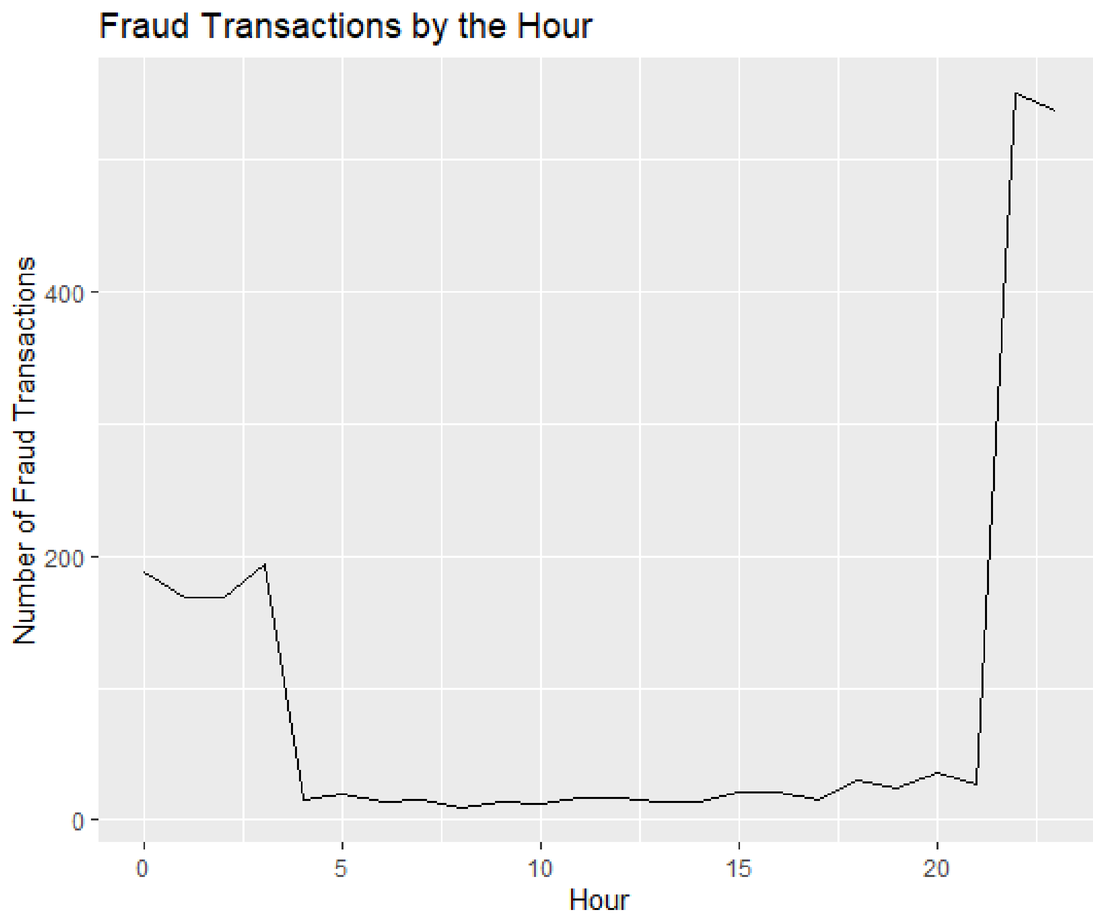
```
#sub-setting dataset for just fraudulent transactions
fraud_txns <- fraud_csv[fraud_csv$is_fraud == 1,]
dim(fraud_txns)

## [1] 2145   32

#checking distribution of fraudulent transaction amount
df_filtered <- fraud_txns[fraud_txns$amt < quantile(fraud_txns$amt, 0.99),]
hist(df_filtered$amt, breaks=100, main='Histogram of Fraud Transaction
Amounts', xlab='Amount')
```




```r
fraud_by_age <- fraud_txns %>%
  group_by(age_2022) %>%
  summarise(count_fraud = n())

ggplot(fraud_by_age, aes(age_2022, count_fraud, width =0.6)) +
  geom_bar(stat = "identity") +
  labs(title = "Fraud by Age", x = "Card Holder Age", y = "Fraud Count") +
scale_x_continuous(n.breaks=20)
```

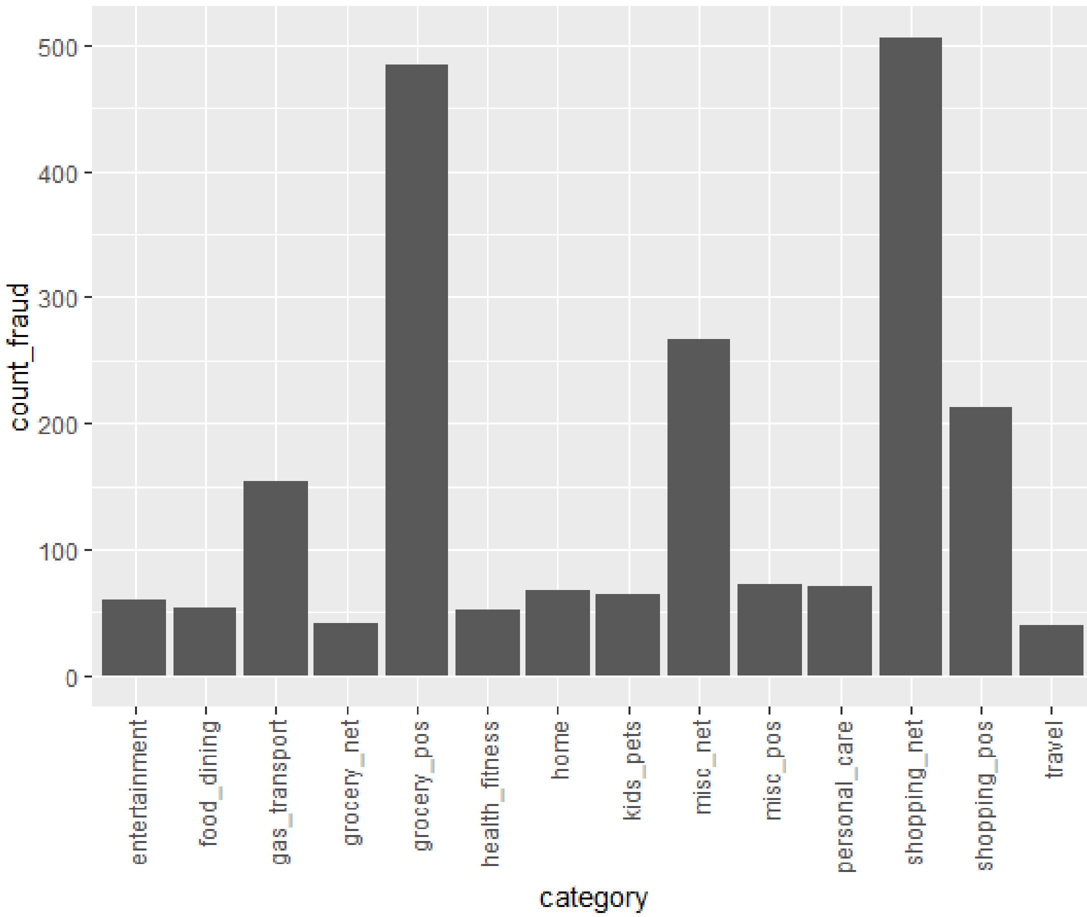



```r
fraud_by_hour <- fraud_txns %>%
  group_by(hour) %>%
  summarise(count_fraud = n())

#visualizing fraud distribution by hour
ggplot(fraud_by_hour, aes(hour, count_fraud)) +
  geom_line() +
  labs(title = "Fraud Transactions by the Hour",
       x = "Hour",
       y = "Number of Fraud Transactions")
```



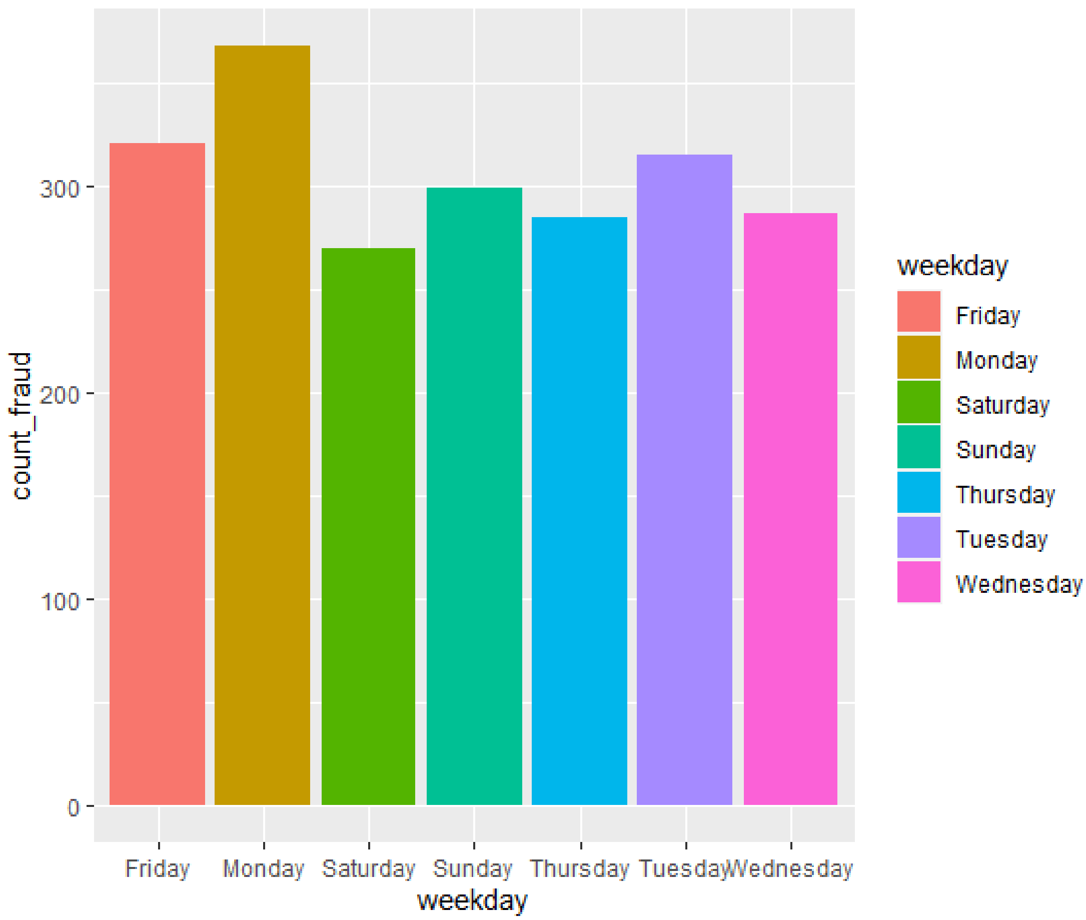
There is a clear trend of fraudulent transactions in the first 3 hours of the day and also last 2 hours.

```
fraud_by_category <- fraud_txns %>%
  group_by(category) %>%
  summarise(count_fraud = n())


ggplot(data= fraud_by_category, aes(x=category, y= count_fraud))+
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

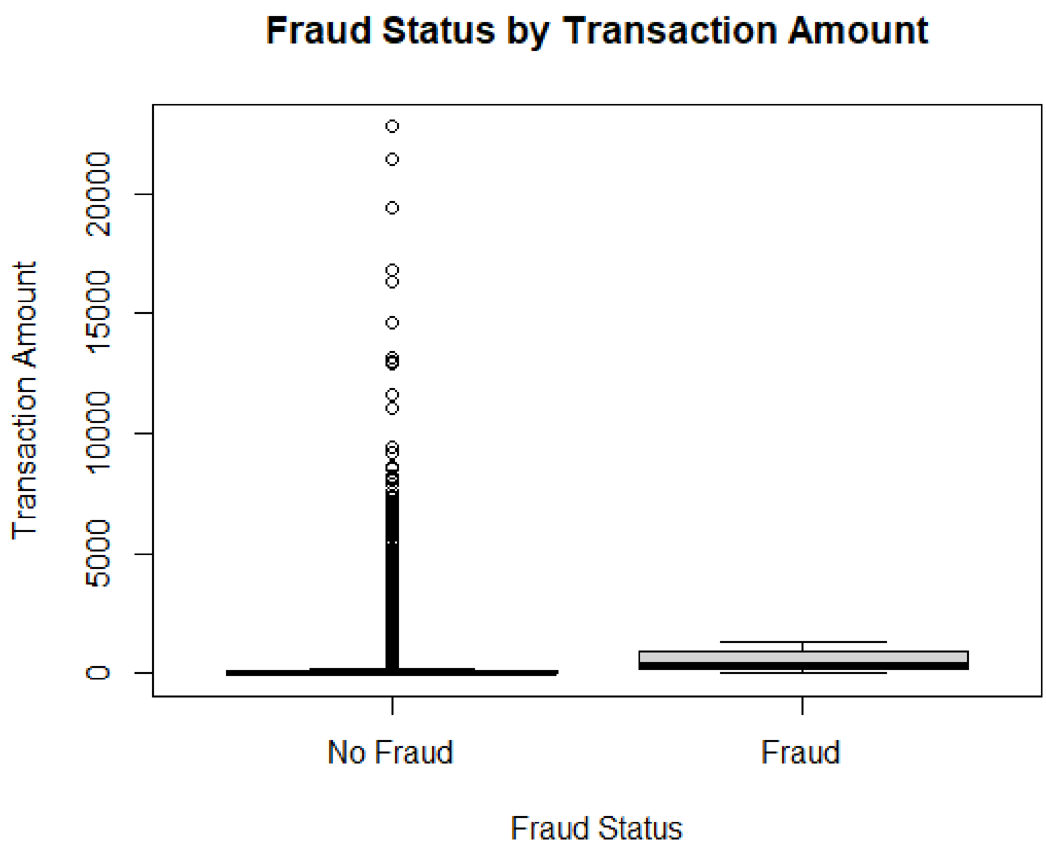


Higher fraud rates in grocery_pos and shopping_net transactions

```
fraud_by_weekday <- fraud_txns %>%
  group_by(weekday) %>%
  summarise(count_fraud = n())


ggplot(data= fraud_by_weekday, aes(x=weekday, y= count_fraud,fill=weekday))+
  geom_bar(stat="identity")
```




```
boxplot(amt ~ is_fraud, data = fraud_csv,
        main = "Fraud Status by Transaction Amount",
        xlab = "Fraud Status",
        ylab = "Transaction Amount",
        names = c("No Fraud", "Fraud"))
```



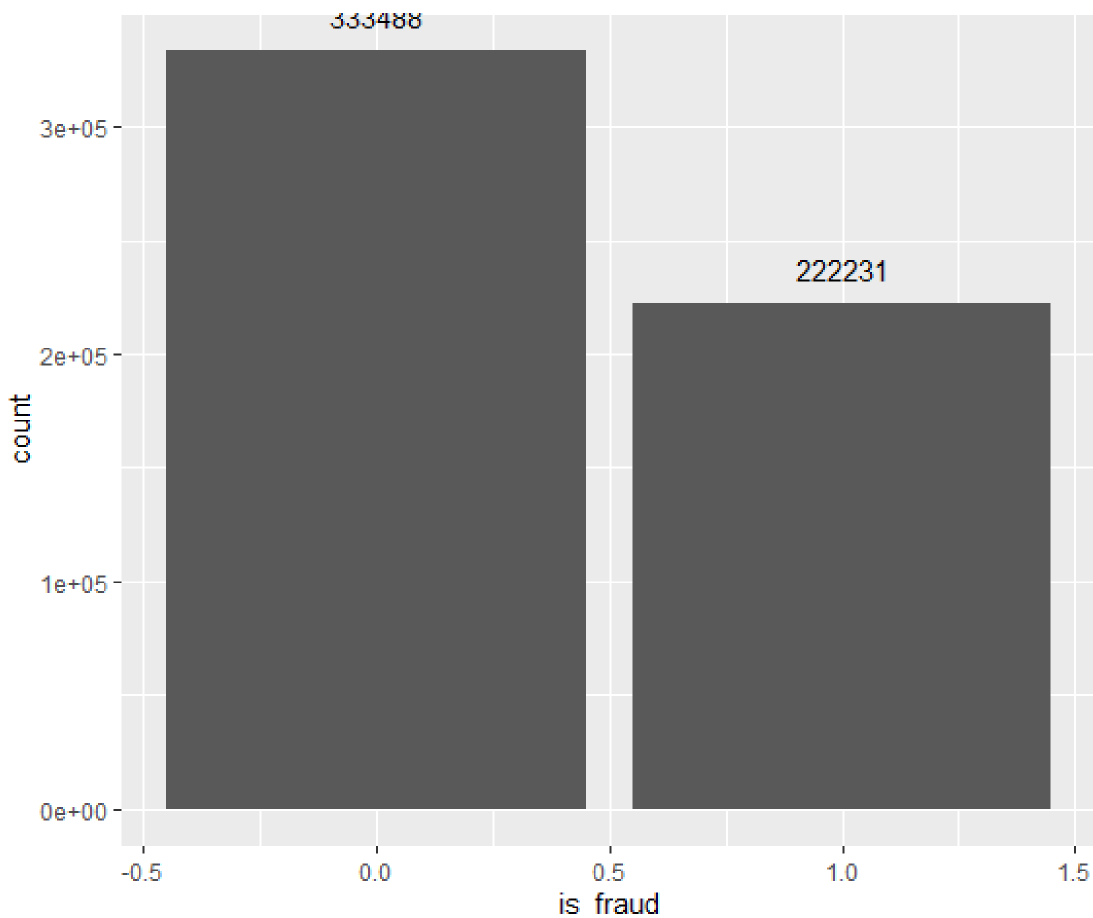
4 Balancing Dataset and Feature Selection

Due to the imbalance that exists in the dataset, a new dataset will be created with a 60:40 balance for non-fraud(0) and fraud(1) cases. ML models will be trained on both the balanced and unbalanced dataset with testing done on the unbalanced (original) dataset.

4.1 Balancing Dataset

```
#setting the no of fraud as non_fraud cases as well as desired percentage of
fraud in new dataset
r0 <- 0.40
n0 <- nrow(fraud_csv)

sampling_result <- ovun.sample(is_fraud ~ ., data = fraud_csv, method =
"both", N = n0,
                               p = r0, seed = 2018)

balanced_data <- sampling_result$data
table(balanced_data$is_fraud)

##
##      0      1
## 333488 222231
```

```
prop.table(table(balanced_data$is_fraud))

##
##         0         1
## 0.6001019 0.3998981

ggplot(balanced_data, aes(x = is_fraud)) +
geom_bar() +
geom_text(stat='count', aes(label=..count..), vjust=-1)
```



4.3 Feature Selection

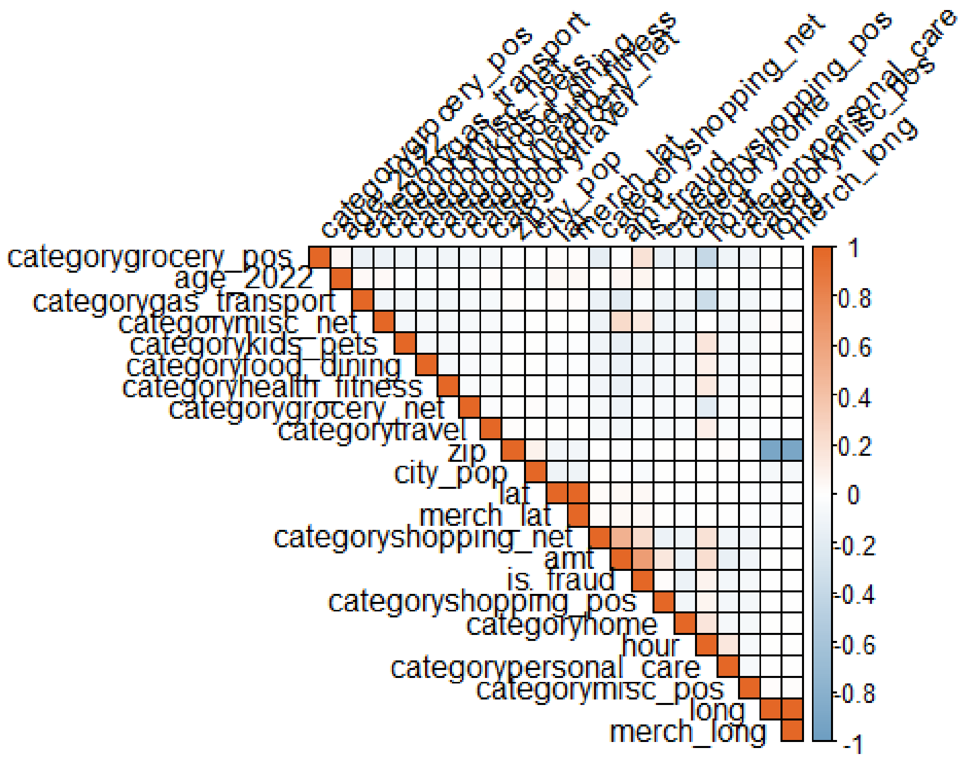
Feature Set 1 - All variables that do not contain relevant information and also modified variables will be dropped.

Balanced Data

```
# Remove unnecessary features using subset
balanced_data2 <- subset(balanced_data, select = -c(X, trans_date_trans_time,
cc_num,merchant, first, last,
                                                   street,
city,dob,trans_num,unix_time, datetime, date,
                                                   time,  weekday, job,
state, full_name))
```

```r
str(balanced_data2)

## 'data.frame':    555719 obs. of  14 variables:
##  $ category  : chr  "food_dining" "gas_transport" "personal_care"
"grocery_net" ...
##  $ amt       : num  65.2 55.9 73.6 47.2 23.8 ...
##  $ gender    : num  1 0 1 1 0 0 1 1 1 1 ...
##  $ zip       : int  74633 97014 16362 54559 29438 31563 29209 95827 11964
74633 ...
##  $ lat       : num  36.7 45.7 41.5 46.5 32.5 ...
##  $ long      : num  -96.8 -121.9 -79.9 -90.4 -80.3 ...
##  $ city_pop  : int  471 1288 1102 795 2408 1324 333497 757530 1858 471 ...
##  $ merch_lat : num  37.4 45.2 41.2 45.5 32.9 ...
##  $ merch_long: num  -96.9 -121.6 -79.1 -90.8 -81 ...
##  $ is_fraud  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age_2022  : num  81 86 32 36 24 95 54 43 38 81 ...
##  $ hour      : num  22 7 15 8 23 14 9 8 19 14 ...
##  $ weekday2  : num  2 2 4 7 7 2 6 5 2 5 ...
##  $ month     : num  10 7 11 11 11 6 11 11 12 7 ...

#encoding categorical variables
dmy <- dummyVars(" ~ .", data = balanced_data2, fullRank = T)
balanced_data3 <- data.frame(predict(dmy, newdata = balanced_data2))

#convert numerical data that are categories to factors
balanced_data3$gender <- factor(balanced_data3$gender)
balanced_data3$month <- factor(balanced_data3$month)
balanced_data3$weekday2 <- factor(balanced_data3$weekday2)


str(balanced_data3)

## 'data.frame':    555719 obs. of  26 variables:
##  $ categoryfood_dining   : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ categorygas_transport : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ categorygrocery_net   : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ categorygrocery_pos   : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ categoryhealth_fitness: num  0 0 0 0 0 1 0 0 0 0 ...
##  $ categoryhome          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorykids_pets     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorymisc_net      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorymisc_pos      : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ categorypersonal_care : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ categoryshopping_net  : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ categoryshopping_pos  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorytravel        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ amt                   : num  65.2 55.9 73.6 47.2 23.8 ...
##  $ gender                : Factor w/ 2 levels "0","1": 2 1 2 2 1 1 2 2 2 2
...
##  $ zip                   : num  74633 97014 16362 54559 29438 ...
```

```
##  $ lat                     : num  36.7 45.7 41.5 46.5 32.5 ...
##  $ long                    : num  -96.8 -121.9 -79.9 -90.4 -80.3 ...
##  $ city_pop                : num  471 1288 1102 795 2408 ...
##  $ merch_lat               : num  37.4 45.2 41.2 45.5 32.9 ...
##  $ merch_long              : num  -96.9 -121.6 -79.1 -90.8 -81 ...
##  $ is_fraud                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ age_2022                : num  81 86 32 36 24 95 54 43 38 81 ...
##  $ hour                    : num  22 7 15 8 23 14 9 8 19 14 ...
##  $ weekday2                : Factor w/ 7 levels "1","2","3","4",..: 2 2 4 7
7 2 6 5 2 5 ...
##  $ month                   : Factor w/ 7 levels "6","7","8","9",..: 5 2 6 6
6 1 6 6 7 2 ...

#Checking for multicollinearity among variables using correlation matrix

cor_matrix <- cor(balanced_data3[, sapply(balanced_data3, is.numeric)])

corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45,
         # Add grid and color
         addgrid.col = "black", col = colorRampPalette(c("#6D9EC1", "white",
"#E46726"))(200))
```



###

Unbalanced data

```
# Remove unnecessary features using subset
unbalanced_data <- subset(fraud_csv, select = -c(X, trans_date_trans_time,
cc_num,merchant, first, last,
                                                   street,
city,dob,trans_num,unix_time, datetime, date,
                                                   time,  weekday, job,
state, full_name))

#encoding categorical variables
dmy <- dummyVars(" ~ .", data = unbalanced_data, fullRank = T)
unbalanced_data2 <- data.frame(predict(dmy, newdata = unbalanced_data))

#convert numerical data that are categories to factors
unbalanced_data2$gender <- factor(unbalanced_data2$gender)
unbalanced_data2$month <- factor(unbalanced_data2$month)
unbalanced_data2$weekday2 <- factor(unbalanced_data2$weekday2)



str(unbalanced_data2)
```

```
## 'data.frame':    555719 obs. of  26 variables:
##  $ categoryfood_dining   : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ categorygas_transport : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorygrocery_net   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorygrocery_pos   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categoryhealth_fitness: num  0 0 1 0 0 0 1 0 0 0 ...
##  $ categoryhome          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorykids_pets     : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ categorymisc_net      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categorymisc_pos      : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ categorypersonal_care : num  1 1 0 0 0 0 0 1 0 0 ...
##  $ categoryshopping_net  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ categoryshopping_pos  : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ categorytravel        : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ amt                   : num  2.86 29.84 41.28 60.05 3.19 ...
##  $ gender                : Factor w/ 2 levels "0","1": 2 1 1 2 2 1 1 1 2 1
...
##  $ zip                   : num  29209 84002 11710 32780 49632 ...
##  $ lat                   : num  34 40.3 40.7 28.6 44.3 ...
##  $ long                  : num  -80.9 -110.4 -73.5 -80.8 -85 ...
##  $ city_pop              : num  333497 302 34496 54767 1126 ...
##  $ merch_lat             : num  34 39.5 40.5 28.8 45 ...
##  $ merch_long            : num  -81.2 -110 -74.2 -80.9 -85.9 ...
##  $ is_fraud              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ age_2022              : num  54 32 51 34 66 30 71 50 49 66 ...
##  $ hour                  : num  12 12 12 12 12 12 12 12 12 12 ...
##  $ weekday2              : Factor w/ 7 levels "1","2","3","4",..: 7 7 7 7
7 7 7 7 7 7 ...
##  $ month                 : Factor w/ 7 levels "6","7","8","9",..: 1 1 1 1
1 1 1 1 1 1 ...
```

5 Machine Learning Modelling

4 Models will be used to evaluate this classification task on both the balanced and unbalanced datasets. They are: i: Logistic Regression ii: Decision Tree iii: KNN iv: Random Forests

The models will be compared using classification ML metrics then the best one selected.

First the dataset will be split into Training and Test set

Setting set for Balanced dataset

```
#To achieve reproducible model; set the random seed number
set.seed(100)

# Data is split into training and test set in a 80:20 ratio
TrainingIndex1 <- createDataPartition(balanced_data3$is_fraud, p=0.75, list =
FALSE)
```

```r
TrainingSet1 <- balanced_data3[TrainingIndex1,]# Training Set
TestingSet1 <- balanced_data3[-TrainingIndex1,]# Test Set

#To achieve reproducible model; set the random seed number
set.seed(1000)

# Data is split into training and test set in a 80:20 ratio
TrainingIndex2 <- createDataPartition(unbalanced_data2$is_fraud, p=0.75, list
= FALSE)

TrainingSet2 <- unbalanced_data2[TrainingIndex2,]# Training Set
TestingSet2 <- unbalanced_data2[-TrainingIndex2,]# Test Set
```

5.1 Logistic Regression

5.1.1 LR Balanced Dataset

```r
cls <- glm(is_fraud~., family='binomial',data=TrainingSet1)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

cut=0.5

#Calculate the training error
yhat = (predict(cls,TrainingSet1,type="response")>cut)
tr.err = mean(TrainingSet1$is_fraud != yhat)
tr.err

## [1] 0.1402217

#Calculate the testing error
yhat_test=(predict(cls,TestingSet2,type="response")>cut)
te_err=mean(TestingSet2$is_fraud!=yhat_test)
te_err

## [1] 0.05466101

# Prediction on TestingSet using Logistic Regression
cls_prediction <- predict(cls, TestingSet2, type ="response")
head(cls_prediction)

##          2          8         13         14         27         30
## 0.20767808 0.18632325 0.05975286 0.08778102 0.09441230 0.10792785

#Assigning probabilities - If prediction exceeds threshold of 0.5, 1 else 0
cls_prediction <- ifelse(cls_prediction >0.5,1,0)
head(cls_prediction)

##  2  8 13 14 27 30
##  0  0  0  0  0  0
```

```r
#Computing confusion matrix values
confusionMatrix(factor(TestingSet2$is_fraud),factor(cls_prediction), mode
='everything', positive ="0")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 130926   7470
##          1    124    409
##
##                Accuracy : 0.9453
##                  95% CI : (0.9441, 0.9465)
##     No Information Rate : 0.9433
##     P-Value [Acc > NIR] : 0.0004544
##
##                   Kappa : 0.0907
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.99905
##             Specificity : 0.05191
##          Pos Pred Value : 0.94602
##          Neg Pred Value : 0.76735
##               Precision : 0.94602
##                  Recall : 0.99905
##                      F1 : 0.97182
##              Prevalence : 0.94329
##          Detection Rate : 0.94240
##    Detection Prevalence : 0.99616
##       Balanced Accuracy : 0.52548
##
##        'Positive' Class : 0
##
```

5.1.2 Unbalanced Dataset

```r
cls2 <- glm(is_fraud~., family='binomial',data=TrainingSet2)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
cut=0.5

#Calculate the training error
yhat = (predict(cls2,TrainingSet2,type="response")>cut)
tr.err = mean(TrainingSet2$is_fraud != yhat)
tr.err
```

```
## [1] 0.004109983
```

```r
#Calculate the testing error
yhat_test=(predict(cls2,TestingSet2,type="response")>cut)
```

```
te_err=mean(TestingSet2$is_fraud!=yhat_test)
te_err

## [1] 0.004081221

# Prediction on TestingSet using Logistic Regression
cls_prediction2 <- predict(cls2, TestingSet2, type ="response")
head(cls_prediction2)

##            2            8           13           14           27
30
## 0.0005146160 0.0007817741 0.0004053561 0.0018323282 0.0011362903
0.0006171092

#Assigning probabilities - If prediction exceeds threshold of 0.5, 1 else 0
cls_prediction2 <- ifelse(cls_prediction2 >0.5,1,0)
head(cls_prediction2)

##  2  8 13 14 27 30
##  0  0  0  0  0  0

#Computing confusion matrix values
confusionMatrix(factor(TestingSet2$is_fraud),factor(cls_prediction2), mode
='everything', positive ="0")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 138362     34
##          1    533      0
##
##                Accuracy : 0.9959
##                  95% CI : (0.9956, 0.9962)
##     No Information Rate : 0.9998
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -5e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9962
##             Specificity : 0.0000
##          Pos Pred Value : 0.9998
##          Neg Pred Value : 0.0000
##               Precision : 0.9998
##                  Recall : 0.9962
##                      F1 : 0.9980
##              Prevalence : 0.9998
##          Detection Rate : 0.9959
##    Detection Prevalence : 0.9962
```

```
##       Balanced Accuracy : 0.4981
##
##        'Positive' Class : 0
##
```

5.2 Decision Trees

5.2.1 DT Balanced Dataset

```
tree1 <- rpart(is_fraud ~., method = 'class', data = TrainingSet1, control =
rpart.control(cp = 0.0001))

# Predict using the decision tree model on the test data
predicted_values <- predict(tree1, newdata = TestingSet2, type = "class")

# Assuming 'is_fraud' is the actual target variable in the test data
actual_values <- TestingSet2$is_fraud
actual_values <- as.factor(actual_values)
levels(actual_values) <- c("0", "1")

tree_predictions1 <-predict(tree1, TestingSet2, type = 'class')
head(tree_predictions1)

##  2  8 13 14 27 30
##  0  0  0  0  0  0
## Levels: 0 1

# Create confusion matrix
conf_matrix1 <- confusionMatrix(predicted_values, actual_values)

print(conf_matrix1)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 137125      0
##          1   1271    533
##
##                Accuracy : 0.9909
##                  95% CI : (0.9903, 0.9913)
##     No Information Rate : 0.9962
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4529
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9908
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.2955
```

```
##               Prevalence : 0.9962
##           Detection Rate : 0.9870
##     Detection Prevalence : 0.9870
##        Balanced Accuracy : 0.9954
##
##         'Positive' Class : 0
##
```

5.2.2 DT Unbalanced Dataset

```
tree2 <- rpart(is_fraud ~., method = 'class', data = TrainingSet2, control =
rpart.control(cp = 0.0001))

# Predict using the decision tree model on the test data
predicted_values2 <- predict(tree2, newdata = TestingSet2, type = "class")

# Assuming 'is_fraud' is the actual target variable in the test data
actual_values2 <- TestingSet2$is_fraud
actual_values2 <- as.factor(actual_values)
levels(actual_values2) <- c("0", "1")

tree_predictions2 <-predict(tree2, TestingSet2, type = 'class')
head(tree_predictions2)

##  2  8 13 14 27 30
##  0  0  0  0  0  0
## Levels: 0 1

# Create confusion matrix
conf_matrix2 <- confusionMatrix(predicted_values2, actual_values2)

print(conf_matrix2)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction      0      1
##          0 138338    154
##          1     58    379
##
##                Accuracy : 0.9985
##                  95% CI : (0.9983, 0.9987)
##     No Information Rate : 0.9962
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7807
##
##  Mcnemar's Test P-Value : 6.817e-11
##
##             Sensitivity : 0.9996
##             Specificity : 0.7111
##          Pos Pred Value : 0.9989
```

```
##          Neg Pred Value : 0.8673
##              Prevalence : 0.9962
##          Detection Rate : 0.9957
##    Detection Prevalence : 0.9969
##       Balanced Accuracy : 0.8553
##
##        'Positive' Class : 0
##
```

5.3 KNN

5.3.1 KNN Balanced

```
fraudtrain1 <-TrainingSet1$is_fraud

#k selection with sqrt of total obs
print(sqrt(nrow(fraudtrain1)/2))

## numeric(0)

k <- 455

knnmodel1  <- knn(train = TrainingSet1, test = TestingSet2, cl = fraudtrain1,
k = k)

accuracy <- 100 * sum(TestingSet2$is_fraud == knnmodel1) / nrow(TestingSet2)
cat("k =", k, "Accuracy =", accuracy, "\n")

## k = 455 Accuracy = 89.69258

# Get predicted labels from the KNN model
predicted_labels1 <- as.numeric(knnmodel1)

# Calculate confusion matrix
conf_matrix1 <- table(Actual = TestingSet2$is_fraud, Predicted =
predicted_labels1)

# Calculate True Positives (TP), False Positives (FP), True Negatives (TN),
False Negatives (FN)
TP <- conf_matrix1[2, 2]
FP <- conf_matrix1[1, 2]
TN <- conf_matrix1[1, 1]
FN <- conf_matrix1[2, 1]

# Calculate True Positive Rate (TPR) and False Positive Rate (FPR)
TPR <- TP / (TP + FN)
TNR <- TN / (TN+FP)
FPR <- FP / (FP + TN)



TPR
```

```
## [1] 0.9043152
```

TNR

```
## [1] 0.8968973
```

FPR

```
## [1] 0.1031027
```

5.3.2 KNN Unbalanced Datset

```
fraudtrain2 <-TrainingSet2$is_fraud

#k selection with sqrt of total obs
print(sqrt(nrow(fraudtrain2)/2))

## numeric(0)

k <- 455

knnmodel2  <- knn(train = TrainingSet2, test = TestingSet2, cl = fraudtrain2,
k = k)

accuracy <- 100 * sum(TestingSet2$is_fraud == knnmodel2) / nrow(TestingSet2)
cat("k =", k, "Accuracy =", accuracy, "\n")

## k = 455 Accuracy = 99.61635

# Get predicted labels from the KNN model
predicted_labels2 <- as.numeric(knnmodel2)

# Calculate confusion matrix
conf_matrix2 <- table(Actual = TestingSet2$is_fraud, Predicted =
predicted_labels2)
```

5.4 Random Forest

5.4.1 RF Balanced Dataset

```
#First step in running rf is converting target variable to factor
TrainingSet1$is_fraud <- as.factor(TrainingSet1$is_fraud)

# converting TestingSet$popular to factor
is_fraud_factor1 <-  as.factor(TestingSet2$is_fraud)

# Assuming your data frame is called 'df' and the target variable is 'target'
rf_model1 <- randomForest(is_fraud~ ., data = TrainingSet1, ntree = 100)
rf_model1

##
## Call:
##  randomForest(formula = is_fraud ~ ., data = TrainingSet1, ntree = 100)
##                Type of random forest: classification
```

```
##                      Number of trees: 100
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.02%
## Confusion matrix:
##        0      1  class.error
## 0 249791     77 0.0003081627
## 1      0 166922 0.0000000000
```

```
rf_predictions1 <- predict(rf_model1, TestingSet2)
head(rf_predictions1)
```

```
##  2  8 13 14 27 30
##  0  0  0  0  0  0
## Levels: 0 1
```

```
cf_rf1 <- confusionMatrix(rf_predictions1, is_fraud_factor1)
cf_rf1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 138360      0
##          1     36    533
##
##                Accuracy : 0.9997
##                  95% CI : (0.9996, 0.9998)
##     No Information Rate : 0.9962
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9672
##
##  Mcnemar's Test P-Value : 5.433e-09
##
##             Sensitivity : 0.9997
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9367
##              Prevalence : 0.9962
##          Detection Rate : 0.9959
##    Detection Prevalence : 0.9959
##       Balanced Accuracy : 0.9999
##
##        'Positive' Class : 0
##
```

5.4.2 RF Unbalanced Dataset

```
#First step in running rf is converting target variable to factor
TrainingSet2$is_fraud <- as.factor(TrainingSet2$is_fraud)
```

```r
# converting TestingSet$popular to factor
is_fraud_factor2 <-  as.factor(TestingSet2$is_fraud)

# Assuming your data frame is called 'df' and the target variable is 'target'
rf_model2 <- randomForest(is_fraud~ ., data = TrainingSet2, ntree = 100)
rf_model2

## 
## Call:
##  randomForest(formula = is_fraud ~ ., data = TrainingSet2, ntree = 100) 
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 5
## 
##         OOB estimate of  error rate: 0.15%
## Confusion matrix:
##        0    1  class.error
## 0 415155   23 5.539793e-05
## 1    584 1028 3.622829e-01

rf_predictions2 <- predict(rf_model2, TestingSet2)
head(rf_predictions2)

##  2  8 13 14 27 30 
##  0  0  0  0  0  0 
## Levels: 0 1

cf_rf2 <- confusionMatrix(rf_predictions2, is_fraud_factor2)
cf_rf2

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction      0      1
##          0 138390    193
##          1      6    340
##                                           
##                Accuracy : 0.9986          
##                  95% CI : (0.9984, 0.9988)
##     No Information Rate : 0.9962          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.7729          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 1.0000          
##             Specificity : 0.6379          
##          Pos Pred Value : 0.9986          
##          Neg Pred Value : 0.9827          
##              Prevalence : 0.9962          
```

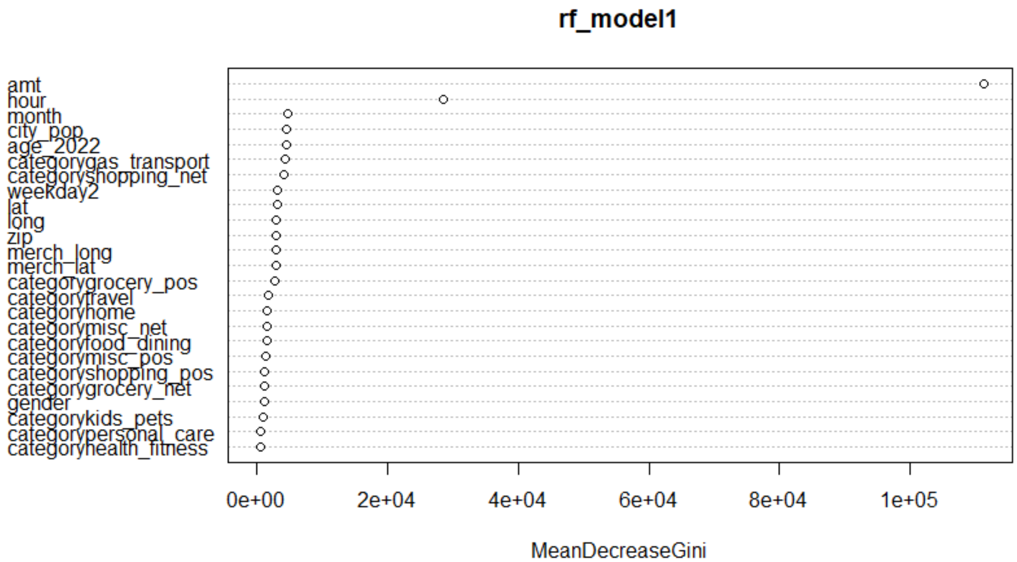
```
##          Detection Rate : 0.9961
##    Detection Prevalence : 0.9975
##       Balanced Accuracy : 0.8189
##
##        'Positive' Class : 0
##
```

Calculating Variable Importance

```
importance_values <- importance(rf_model1)
importance_values
```

```
##                          MeanDecreaseGini
## categoryfood_dining             1503.8373
## categorygas_transport           4351.5447
## categorygrocery_net             1207.6752
## categorygrocery_pos             2821.2147
## categoryhealth_fitness           509.4334
## categoryhome                    1605.8688
## categorykids_pets                896.9427
## categorymisc_net                1530.7656
## categorymisc_pos                1360.5457
## categorypersonal_care            631.2534
## categoryshopping_net            4091.5001
## categoryshopping_pos            1232.9741
## categorytravel                  1668.5266
## amt                           111257.4079
## gender                          1205.6115
## zip                             2990.3055
## lat                             3160.2742
## long                            3018.1267
## city_pop                        4486.7403
## merch_lat                       2884.2972
## merch_long                      2889.8851
## age_2022                        4468.6060
## hour                           28536.1991
## weekday2                        3192.7751
## month                           4737.7620
```

```
varImpPlot(rf_model1)
```



6 Model Evaluation

Models are evaluated using accuracy from confusion matrix, testing error and also AUC score.

6.1 Plotting ROC Curves

```
#converting prediction scores data type before plotting curves
cls_prediction2 <- as.numeric(cls_prediction2)
tree_curve2 <- as.numeric(tree_predictions2)
rf_predictions2 <- as.numeric(rf_predictions2)
```

```
#calculating AUC curves of models# Calculate ROC and AUC using pROC
cls_score2 <- print(paste('cls roc_roc_curve score is',auc(cls_roc_curve2)))
```

```
## [1] "cls roc_roc_curve score is 0.499877154315858"
```

```
tree_score2 <- print(paste('tree_roc_curve score is',auc(tree_roc_curve2)))
```

```
## [1] "tree_roc_curve score is 0.995440257253315"
```

```
rf_score2 <- print(paste('rf_roc_curve score is', auc(rf_roc_curve2)))
```

```
## [1] "rf_roc_curve score is 0.805114540848315"
```

```r
#creating the ROC function
cls_roc_curve2 <- roc(TestingSet2$is_fraud, cls_prediction2)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

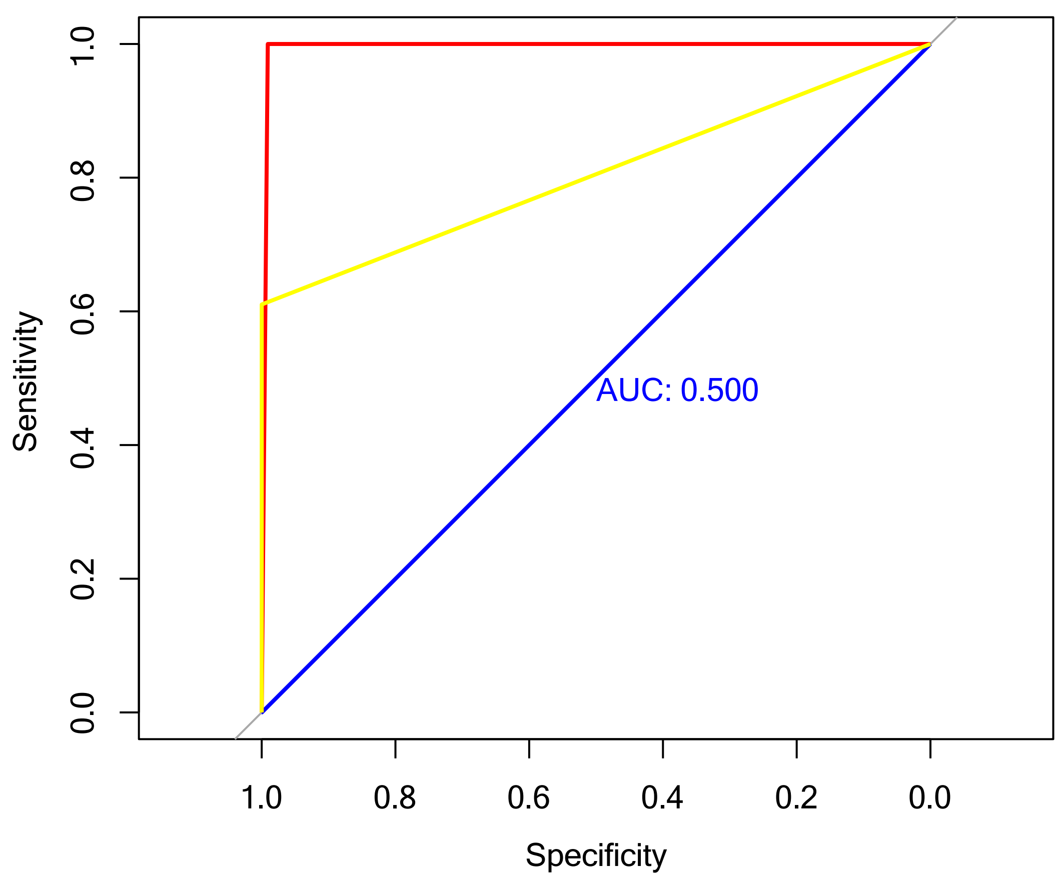
```r
tree_roc_curve2 <- roc(TestingSet2$is_fraud, tree_curve2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
rf_roc_curve2 <- roc(TestingSet2$is_fraud, rf_predictions2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Plotting ROC Curves
plot(cls_roc_curve2, col = "blue", print.auc = TRUE)
plot(tree_roc_curve2, col = "red", add = TRUE)
plot(rf_roc_curve2, col = "yellow", add = TRUE)
```



6.2 Table of Results

```
# Create a new table with some sample data
Model_Comparison <- data.frame(
  Model = c("Logistic Regression", "Logistic Regression", "Decison Trees",
"Decision Trees", "Random Forests", "Random Forests", "KNN", "KNN"),
  Dataset = c("Balanced", "Unbalanced","Balanced",
"Unbalanced","Balanced","Unbalanced","Balanced", "Unbalanced"),
  Accuracy = c(0.861, 0.996, 0.994, 0.999, 0.999, 0.999,0.897, 0.996),
  TestingError = c(0.139, 0.004, 0.006, 0.001, 0.103,0.004, 0.001, 0.001),
  Sensitivity = c(0.999, 0.996, 0.991, 0.999, 0.904, 0.722, 0.999, 0.999),
  Specificity = c(0.05, 0.00, 1, 0.711, 0.897, 0.826, 1, 0.647))

# Display the new table
print(Model_Comparison)

##                 Model    Dataset Accuracy TestingError Sensitivity
Specificity
## 1 Logistic Regression   Balanced    0.861        0.139       0.999
0.050
## 2 Logistic Regression Unbalanced    0.996        0.004       0.996
0.000
## 3       Decison Trees   Balanced    0.994        0.006       0.991
1.000
## 4      Decision Trees Unbalanced    0.999        0.001       0.999
0.711
## 5      Random Forests   Balanced    0.999        0.103       0.904
0.897
## 6      Random Forests Unbalanced    0.999        0.004       0.722
0.826
## 7                 KNN   Balanced    0.897        0.001       0.999
1.000
## 8                 KNN Unbalanced    0.996        0.001       0.999
0.647
```