Financial Payment System and Fraud

Esther Roseline

1/9/2020

Introduction

This is a report for Capstone Project II of Harvardx Data Science Program. The purpose is to develop a machine learning model to detect fraud in a financial payment system. We are using the "Synthetic data from a financial payment system" dataset downloaded from kaggle.com. Here is the Github link (https://github.com/estherlab/Synthetic-Financial-Data-for-Fraud-Detection.git). The key steps we will perform are, as follows:

- Importing the Dataset and essential libraries
- Performing Data Exploration
- Preprocessing
- Splitting Dataset into training and test sets
- Testing different models
- Measuring the performance

Methods/Analysis

Dataset and Libraries

We are importing the dataset that contains aggregated transactional data of bank payments and the essential libraries

```
library(ranger)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(data.table)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.3
```

```
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::lift()      masks caret::lift()
## x purrr::transpose() masks data.table::transpose()
```

```
library(ggplot2)
library(dplyr)
banksim_data <- read.csv("./banksim1/bs140513_032310.csv")
```

Data Exploration

We will now explore the data contained in the banksim_data dataframe. We will proceed by displaying the banksim_data by dimension, summary, and printing the first and last six lines of the dataframe.

```
## [1] 594643     10
```

```
##       step                 customer           age          gender
##  Min.   :  0.00   'C1978250683':   265   '2'    :187310   'E':  1178
##  1st Qu.: 52.00   'C1275518867':   252   '3'    :147131   'F':324565
##  Median : 97.00   'C806399525' :   237   '4'    :109025   'M':268385
##  Mean   : 94.99   'C515668508' :   205   '5'    : 62642   'U':   515
##  3rd Qu.:139.00   'C1338396147':   195   '1'    : 58131
##  Max.   :179.00   'C1896850232':   192   '6'    : 26774
##                   (Other)      :593297   (Other):  3630
##    zipcodeOri              merchant        zipMerchant
##  '28007':594643   'M1823072687':299693   '28007':594643
##                   'M348934600' :205426
##                   'M85975013'  : 26254
##                   'M1053599405':  6821
##                   'M151143676' :  6373
##                   'M855959430' :  6098
##                   (Other)      : 43978
##                     category            amount              fraud
##  'es_transportation'    :505119   Min.   :   0.00   Min.   :0.00000
##  'es_food'              : 26254   1st Qu.:  13.74   1st Qu.:0.00000
##  'es_health'            : 16133   Median :  26.90   Median :0.00000
##  'es_wellnessandbeauty' : 15086   Mean   :  37.89   Mean   :0.01211
##  'es_fashion'           :  6454   3rd Qu.:  42.54   3rd Qu.:0.00000
##  'es_barsandrestaurants':  6373   Max.   :8329.96   Max.   :1.00000
##  (Other)                : 19224
```

```
##   step      customer age gender zipcodeOri      merchant zipMerchant
## 1    0 'C1093826151' '4'    'M'    '28007'  'M348934600'     '28007'
## 2    0  'C352968107' '2'    'M'    '28007'  'M348934600'     '28007'
## 3    0 'C2054744914' '4'    'F'    '28007' 'M1823072687'     '28007'
## 4    0 'C1760612790' '3'    'M'    '28007'  'M348934600'     '28007'
## 5    0  'C757503768' '5'    'M'    '28007'  'M348934600'     '28007'
## 6    0 'C1315400589' '3'    'F'    '28007'  'M348934600'     '28007'
```

```
##              category amount fraud
## 1 'es_transportation'   4.55     0
## 2 'es_transportation'  39.68     0
## 3 'es_transportation'  26.89     0
## 4 'es_transportation'  17.25     0
## 5 'es_transportation'  35.72     0
## 6 'es_transportation'  25.81     0
```

```
##        step      customer age gender zipcodeOri      merchant zipMerchant
## 594638  179  'C748358246' '2'    'M'    '28007' 'M1823072687'     '28007'
## 594639  179 'C1753498738' '3'    'F'    '28007' 'M1823072687'     '28007'
## 594640  179  'C650108285' '4'    'F'    '28007' 'M1823072687'     '28007'
## 594641  179  'C123623130' '2'    'F'    '28007'  'M349281107'     '28007'
## 594642  179 'C1499363341' '5'    'M'    '28007' 'M1823072687'     '28007'
## 594643  179  'C616528518' '4'    'F'    '28007' 'M1823072687'     '28007'
##                   category amount fraud
## 594638 'es_transportation'  51.17     0
## 594639 'es_transportation'  20.53     0
## 594640 'es_transportation'  50.73     0
## 594641        'es_fashion'  22.44     0
## 594642 'es_transportation'  14.46     0
## 594643 'es_transportation'  26.93     0
```

The number of transactions that have been observed as fraud vs normal is, as follows: (0 means normal, 1 means fraud)

```
##
##      0      1
## 587443   7200
```

Hence the percentage of fraud vs. normal transactions is:

```
percent_fraud
```

```
## [1] 1.21
```

```
percent_normal
```

```
## [1] 98.79
```

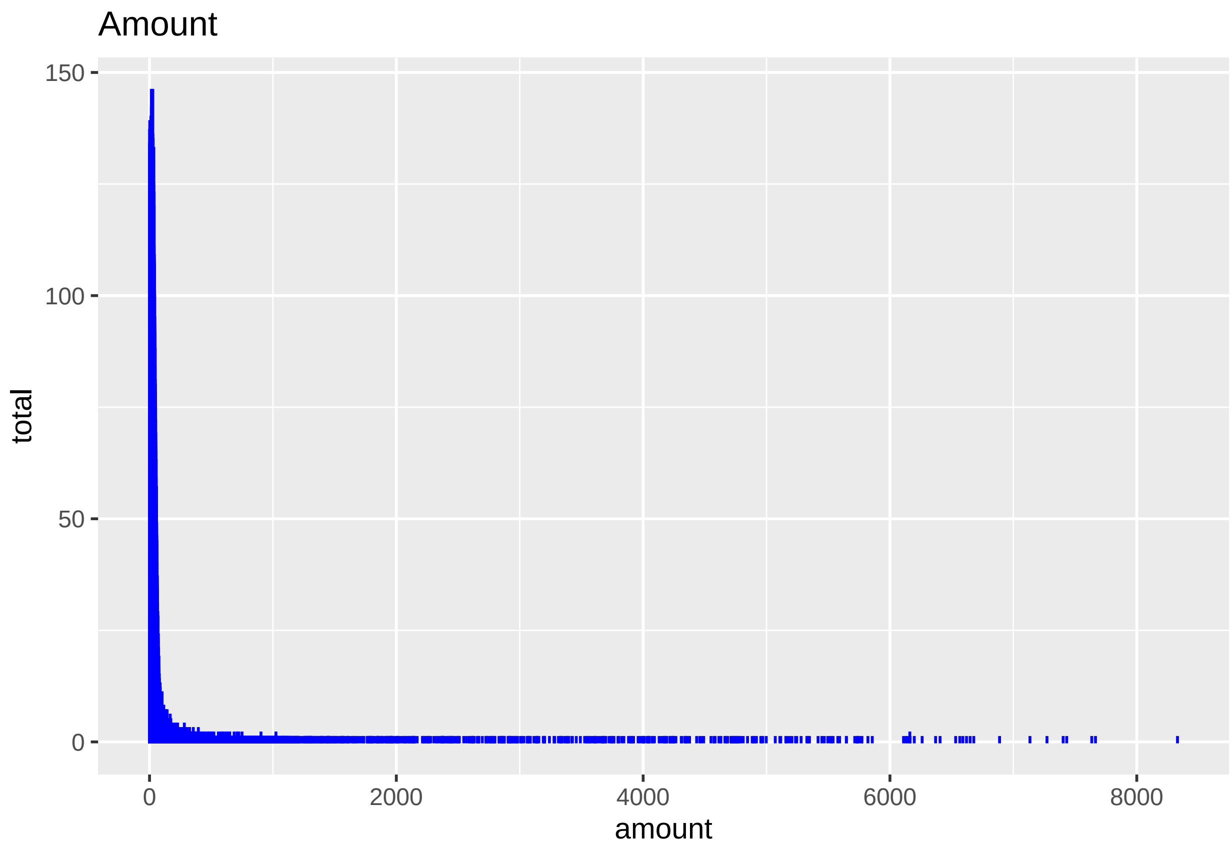
Now we know we have a very unbalanced data set here. The fraud sample is only about 1% of the total dataset. So we must be very careful in looking at the accuracy of our model later.

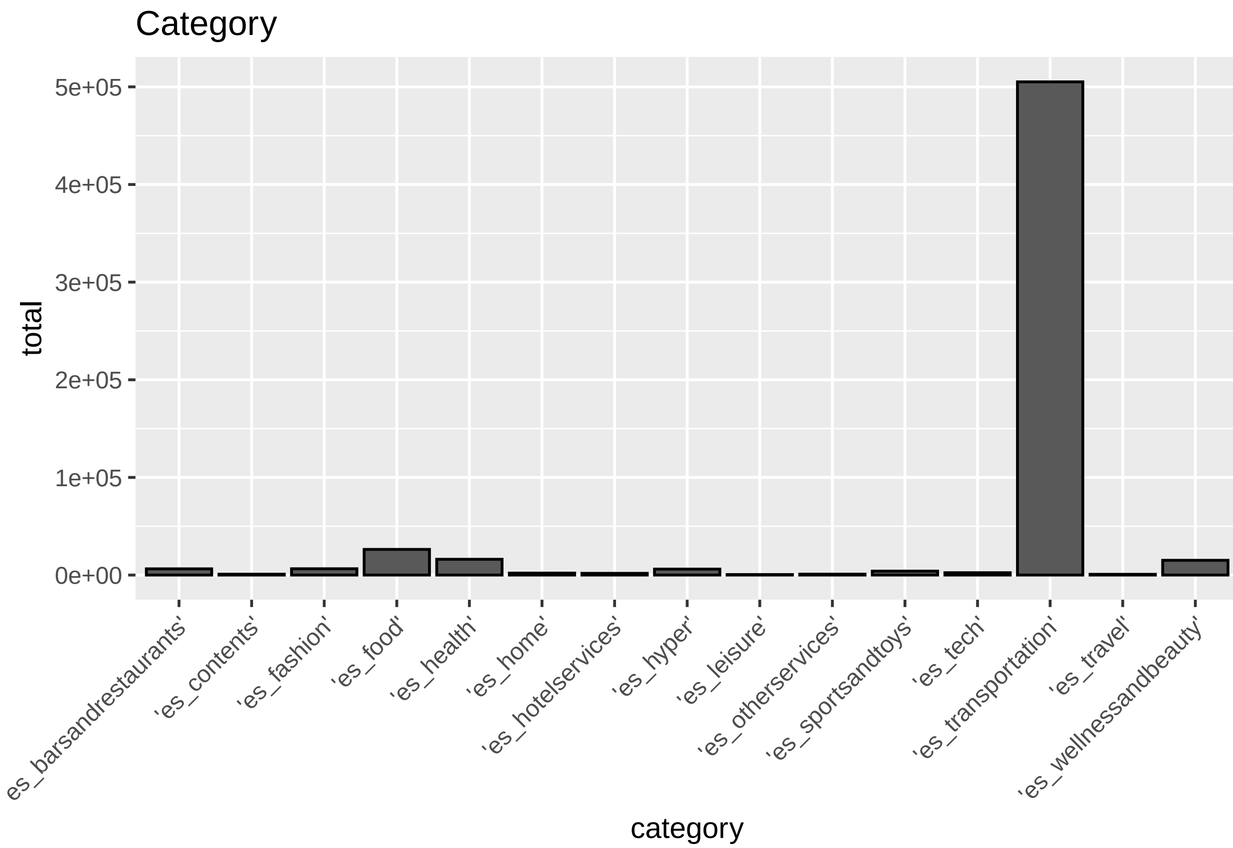
Now we will look into the variables at the dataset. There are 10 variables

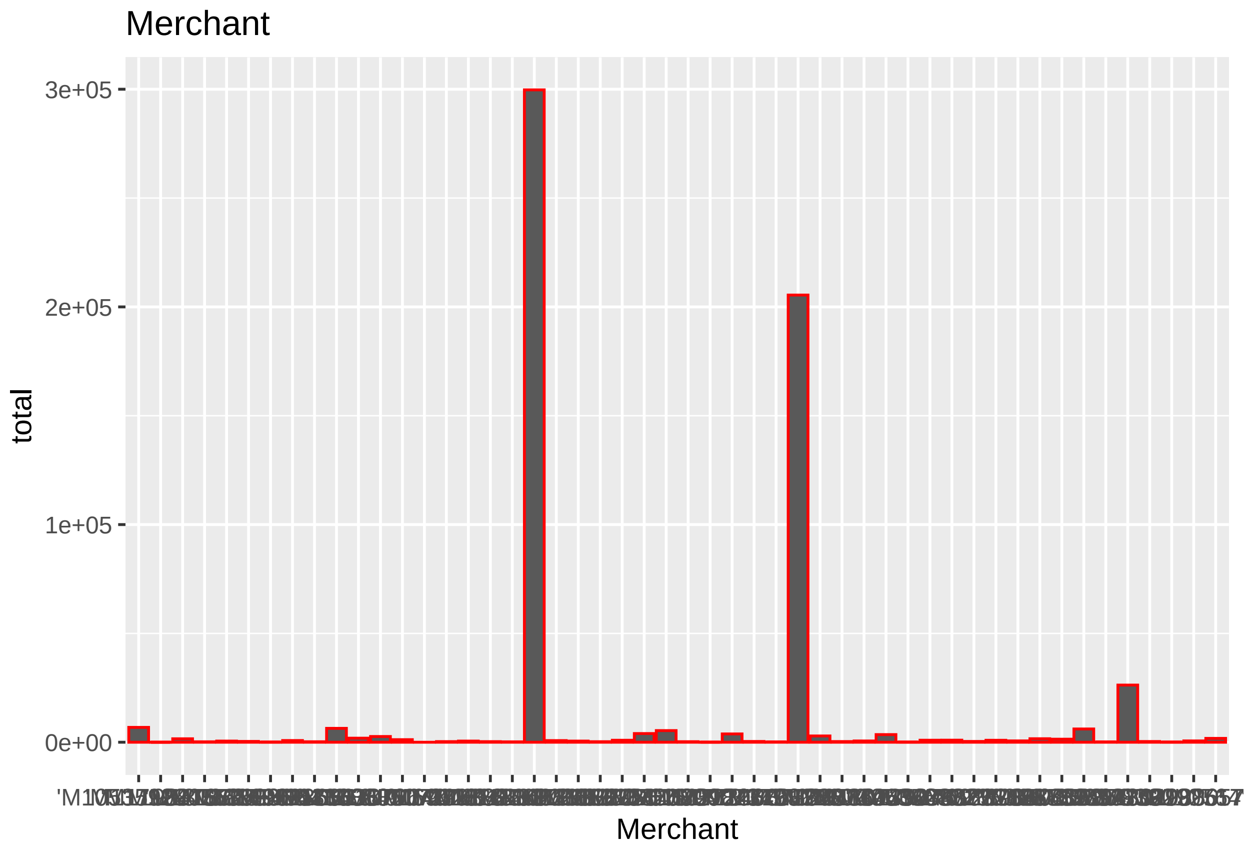
```
##  [1] "step"        "customer"    "age"         "gender"      "zipcodeOri"
##  [6] "merchant"    "zipMerchant" "category"    "amount"      "fraud"
```

In this case, we notice that the first 9 variables (step, customer, age, gender, zipcodeOri, merchant, zipMerchant, category, amount) are features, while the 10th variable is our target (fraud)

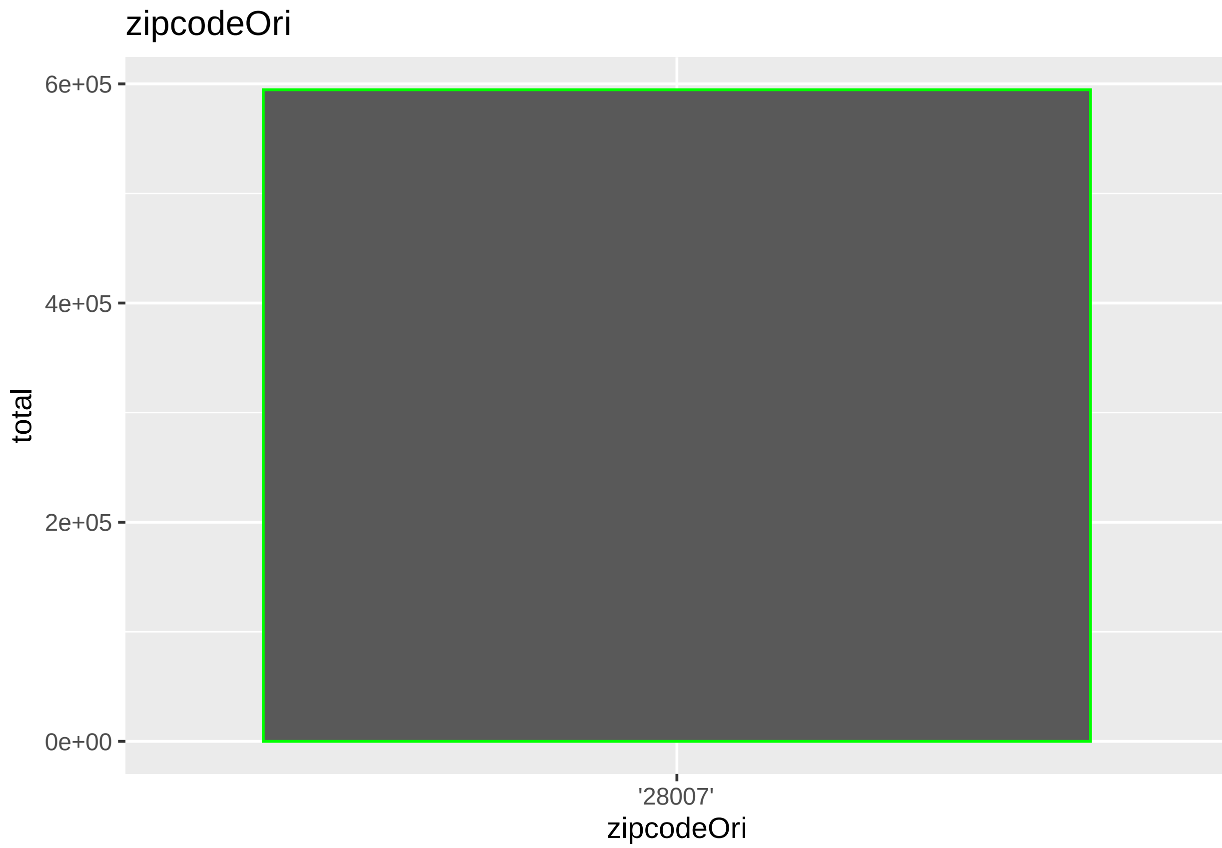
We will start looking into the variables by visualizing the distribution of each of the features to see how their variance is.

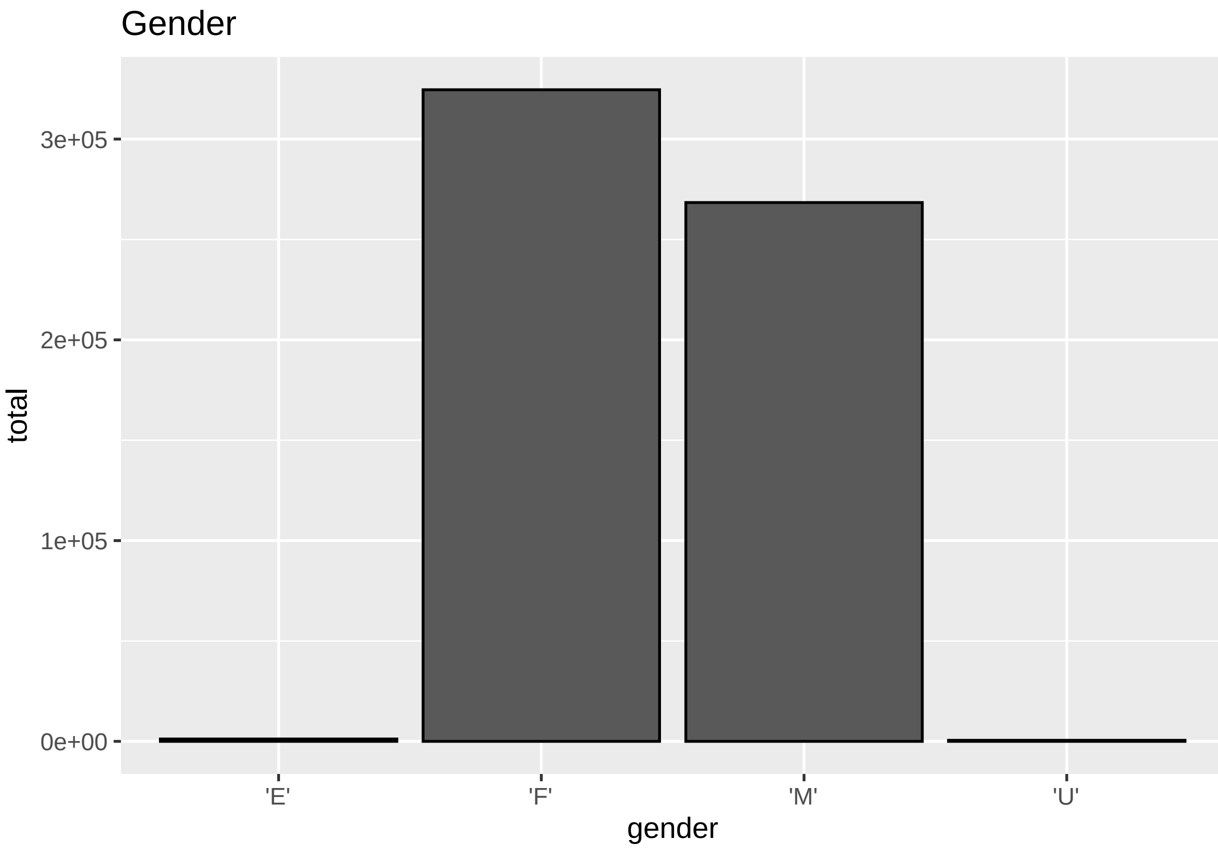


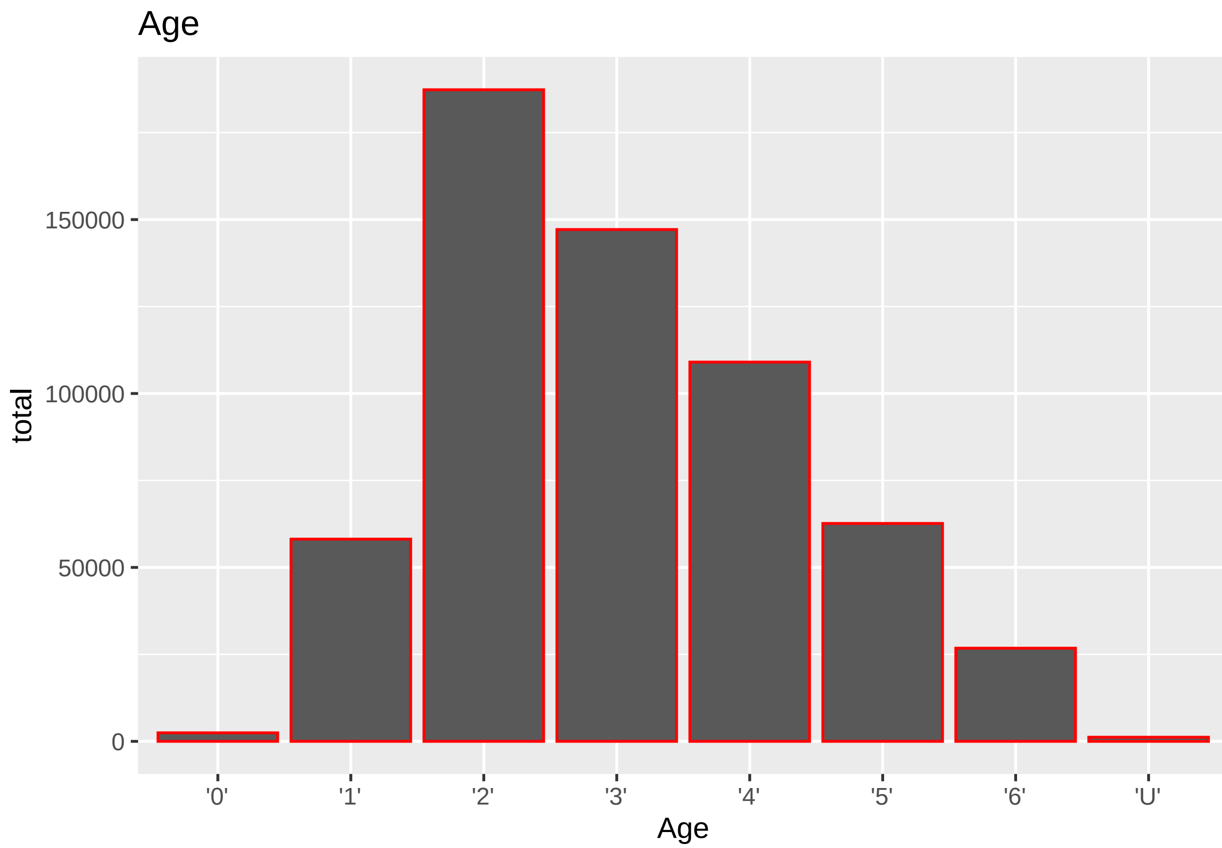

Category
total
5e+05
4e+05
3e+05
2e+05
1e+05
0e+00
es_barsandrestaurants'
'es_contents'
'es_fashion'
'es_food'
'es_health'
'es_home'
'es_hotelservices'
'es_hyper'
'es_leisure'
'es_otherservices'
'es_sportsandtoys'
'es_tech'
'es_transportation'
'es_travel'
'es_wellnessandbeauty'
category


Merchant
3e+05
2e+05
1e+05
0e+00
total
Merchant

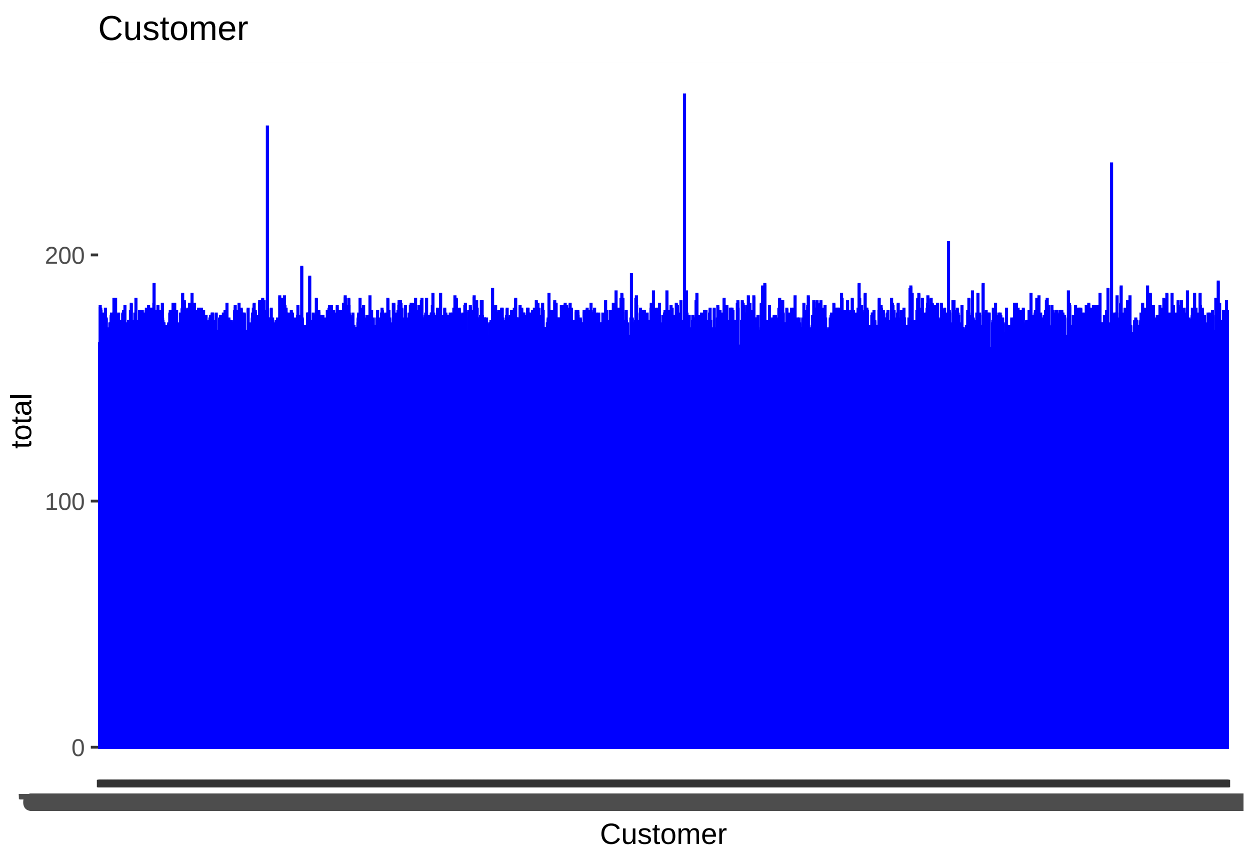

ZipMerchant
6e+05
4e+05
2e+05
0e+00
total
'28007'
zipMerchant

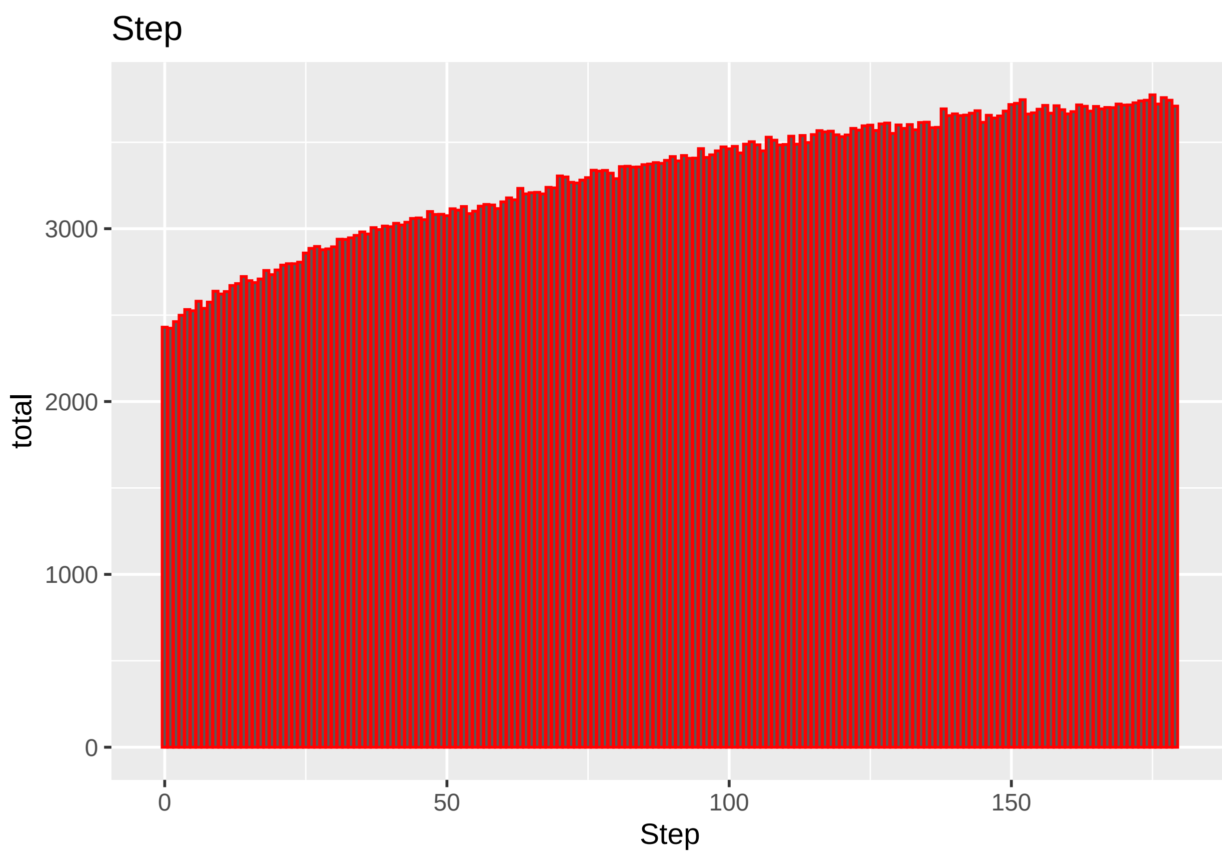

zipcodeOri
6e+05
4e+05
2e+05
0e+00
total
'28007'
zipcodeOri


Gender
3e+05
2e+05
1e+05
0e+00
total
'E'
'F'
'M'
'U'
gender



Age
150000
100000
50000
0
total
'0'
'1'
'2'
'3'
'4'
'5'
'6'
'U'
Age


Customer
200
100
0
total
Customer



From the visualizations, we can safely infer that there are features that won't affect our classification process/prediction so we can take out/ignore in our models:

- zipcodeOri and zipMerchant, because they are constant/only have 1 unique entry.
- Customer, because practically anybody can have any random customer IDs, and it is better that we just account for the relevant customers' characteristics by the other features (such as gender and age). We will factor in the customers' characteristics in our model.

Preprocessing

We will standardize the amount data in our banksim_data, and simulatenously create a new dataset that has had zipcodeOri, zipMerchant, and Customer, and step data removed

```
banksim_data$amount <- scale(banksim_data$amount)
Standardized_Data <- banksim_data[,-c(1)]
NewData_banksim <- Standardized_Data [,-c(1,4,6)]
```

Our new data now looks like as follows:

```
##   age gender      merchant            category      amount fraud
## 1 '4'    'M'  'M348934600' 'es_transportation' -0.29927548     0
## 2 '2'    'M'  'M348934600' 'es_transportation'  0.01606660     0
## 3 '4'    'F' 'M1823072687' 'es_transportation' -0.09874197     0
## 4 '3'    'M'  'M348934600' 'es_transportation' -0.18527478     0
## 5 '5'    'M'  'M348934600' 'es_transportation' -0.01948007     0
## 6 '3'    'F'  'M348934600' 'es_transportation' -0.10843652     0
```

Splitting Datasets

We will now split our datasets into training and test sets. The test set will be 20% of the whole data set. The 80% will go to the train set.

```
set.seed(1)
test_index <- createDataPartition(y = NewData_banksim$fraud, times = 1, p = 0.2, list = FALSE)
train_data <- NewData_banksim[-test_index,]
test_data <- NewData_banksim[test_index,]
```

```
##       age         gender              merchant
##  '2'    :37778   'E':  219   'M1823072687':59892
##  '3'    :29369   'F':64940   'M348934600' :41268
##  '4'    :21683   'M':53647   'M85975013'  : 5272
##  '5'    :12494   'U':  123   'M1053599405': 1364
##  '1'    :11566               'M151143676' : 1225
##  '6'    : 5355               'M855959430' : 1214
##  (Other):  684               (Other)      : 8694
##                      category          amount.V1          fraud
##  'es_transportation'    :101160   Min.   :-0.34012   Min.   :0.00000
##  'es_food'              :  5272   1st Qu.:-0.21633   1st Qu.:0.00000
##  'es_health'            :  3159   Median :-0.09784   Median :0.00000
##  'es_wellnessandbeauty' :  3014   Mean   : 0.00170   Mean   :0.01197
##  'es_fashion'           :  1284   3rd Qu.: 0.04344   3rd Qu.:0.00000
##  'es_barsandrestaurants':  1225   Max.   :74.43321   Max.   :1.00000
##  (Other)                :  3815
```

```
##       age          gender               merchant
##  '2'    :149532   'E':   959   'M1823072687':239801
##  '3'    :117762   'F':259625   'M348934600' :164158
##  '4'    : 87342   'M':214738   'M85975013'  : 20982
##  '5'    : 50148   'U':   392   'M1053599405':  5457
##  '1'    : 46565                'M151143676' :  5148
##  '6'    : 21419                'M855959430' :  4884
##  (Other):  2946                (Other)      : 35284
##                      category          amount.V1          fraud
##  'es_transportation'    :403959   Min.   :-0.34012   Min.   :0.00000
##  'es_food'              : 20982   1st Qu.:-0.21687   1st Qu.:0.00000
##  'es_health'            : 12974   Median :-0.09883   Median :0.00000
##  'es_wellnessandbeauty' : 12072   Mean   :-0.00043   Mean   :0.01214
##  'es_fashion'           :  5170   3rd Qu.: 0.04138   3rd Qu.:0.00000
##  'es_barsandrestaurants':  5148   Max.   :68.46926   Max.   :1.00000
##  (Other)                : 15409
```

Testing different models

First Model: Decision Tree

The first model we will try is a Decision Tree model.

```
require(tree)
```

```
## Loading required package: tree

## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```r
library(rpart)
library(rpart.plot)
tree.banksim <- rpart(fraud~.,train_data, method='class')
```

However, plotting the classification tree produces a very hard-to-read figure Nonetheless, let us see the summary of our classification tree:

```
## Call:
## rpart(formula = fraud ~ ., data = train_data, method = "class")
##   n= 475714
##
##           CP nsplit rel error    xerror        xstd
## 1 0.47420360      0 1.0000000 1.0000000 0.013077771
## 2 0.08431440      1 0.5257964 0.5325485 0.009571000
## 3 0.03384695      2 0.4414820 0.4407895 0.008712378
## 4 0.01000000      4 0.3737881 0.3744806 0.008033624
##
## Variable importance
##   amount merchant category
##       59       31       10
##
## Node number 1: 475714 observations,    complexity param=0.4742036
##   predicted class=0  expected loss=0.01214175  P(node) =1
##     class counts: 469938  5776
##    probabilities: 0.988 0.012
##   left son=2 (471543 obs) right son=3 (4171 obs)
##   Primary splits:
##       amount   < 2.004032 to the left,  improve=5606.4060000, (0 missing)
##       merchant splits as  LLLRLLRLLLLLLLRRLLLLLRLLLRRLRLLLRLRRLLLLLRLLLRLLLLR, improve=5202.0930000,
##       category splits as  LLLLLLRLRRRLLRL, improve=2791.8600000, (0 missing)
##       gender   splits as  LRLL, improve=   7.7199790, (0 missing)
##       age      splits as  RRRRRLLL, improve=   0.5214234, (0 missing)
##   Surrogate splits:
##       merchant splits as  LLLRLLRLLLLLLLLLLLLLLRLLLLRLLLLLRLLRLLLLRLLLRLLLLL, agree=0.993, adj=0.151
##       category splits as  LLLLLLLLRLLLLRL, agree=0.992, adj=0.142, (0 split)
##
## Node number 2: 471543 observations,    complexity param=0.03384695
##   predicted class=0  expected loss=0.004922139  P(node) =0.9912321
##     class counts: 469222  2321
##    probabilities: 0.995 0.005
##   left son=4 (467795 obs) right son=5 (3748 obs)
##   Primary splits:
##       merchant splits as  LLLRLLRLLLLLLLRRLLLLLRLLLRLLRLLLRLRLLLLLLRLLLRLLLLR, improve=1103.8670000,
##       category splits as  LLLLLLLLRLRLLRL, improve= 490.3869000, (0 missing)
##       amount   < 1.030897 to the left,  improve= 355.0821000, (0 missing)
##       gender   splits as  LRLL, improve=   1.5723240, (0 missing)
##       age      splits as  RRRLRLLL, improve=   0.1525445, (0 missing)
```

```
##   Surrogate splits:
##       category splits as  LLLLLLLLRLRLLRL, agree=0.993, adj=0.081, (0 split)
##
## Node number 3: 4171 observations,    complexity param=0.0843144
##   predicted class=1  expected loss=0.1716615  P(node) =0.008767873
##     class counts:   716  3455
##    probabilities: 0.172 0.828
##   left son=6 (579 obs) right son=7 (3592 obs)
##   Primary splits:
##       merchant splits as  LLRR-LRLLRR-RLRRRL-LLRRL-RRRRR-LRLRLLR-LRRRRR-RR-R, improve=754.136400, (0
##       amount   < 2.61241  to the left,  improve=221.288100, (0 missing)
##       category splits as  R-R-LLRRRRRL-RR, improve=177.045700, (0 missing)
##       gender   splits as  LRLL, improve=  5.864769, (0 missing)
##       age      splits as  RRRRRRRL, improve=  1.765169, (0 missing)
##   Surrogate splits:
##       gender splits as  RRRL, agree=0.862, adj=0.003, (0 split)
##
## Node number 4: 467795 observations
##   predicted class=0  expected loss=0.001859789  P(node) =0.9833534
##     class counts: 466925   870
##    probabilities: 0.998 0.002
##
## Node number 5: 3748 observations,    complexity param=0.03384695
##   predicted class=0  expected loss=0.3871398  P(node) =0.007878683
##     class counts:  2297  1451
##    probabilities: 0.613 0.387
##   left son=10 (2745 obs) right son=11 (1003 obs)
##   Primary splits:
##       merchant splits as  ---R--R-------LL-----R---L--L---R-L-----R---L----R, improve=259.451400, (0
##       category splits as  ----LLR-R-R--R-, improve=146.491900, (0 missing)
##       amount   < 1.054595 to the left,  improve=115.681500, (0 missing)
##       gender   splits as  LRL-, improve= 14.206760, (0 missing)
##       age      splits as  LLRLLRLL, improve=  6.053547, (0 missing)
##   Surrogate splits:
##       category splits as  ----LLL-R-R--R-, agree=0.840, adj=0.402, (0 split)
##       amount   < 1.966331 to the left,  agree=0.733, adj=0.001, (0 split)
##
## Node number 6: 579 observations
##   predicted class=0  expected loss=0.07944732  P(node) =0.001217118
##     class counts:   533    46
##    probabilities: 0.921 0.079
##
## Node number 7: 3592 observations
##   predicted class=1  expected loss=0.05094655  P(node) =0.007550755
##     class counts:   183  3409
##    probabilities: 0.051 0.949
##
## Node number 10: 2745 observations
##   predicted class=0  expected loss=0.2746812  P(node) =0.005770274
##     class counts:  1991   754
##    probabilities: 0.725 0.275
##
## Node number 11: 1003 observations
##   predicted class=1  expected loss=0.3050847  P(node) =0.00210841
```

```
##     class counts:   306   697
##    probabilities: 0.305 0.695
```

From the summary, however, we discover in Variable Importance that actually only "amount", "merchant", and "category" which are the most important features in our dataset for predicting fraud.

We will now review our prediction:

```r
tree.pred <- predict(tree.banksim, test_data, type="class")
with(test_data, table(tree.pred, fraud))
```

```
##          fraud
## tree.pred      0      1
##         0 117386    397
##         1    119   1027
```

Because we have an unbalanced data, we need to see the confusion matrix as follows:

```r
confusionMatrix(tree.pred, as.factor(test_data$fraud))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 117386    397
##          1    119   1027
##
##                Accuracy : 0.9957
##                  95% CI : (0.9953, 0.996)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7971
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9990
##             Specificity : 0.7212
##          Pos Pred Value : 0.9966
##          Neg Pred Value : 0.8962
##              Prevalence : 0.9880
##          Detection Rate : 0.9870
##    Detection Prevalence : 0.9904
##       Balanced Accuracy : 0.8601
##
##        'Positive' Class : 0
##
```

Here in this case, we can see that the balanced accuracy is about 0.86. And it can detect fraud at the rate of 0.72 (specificity). The model works pretty fine too on the unbalanced data since it produces high precision and high recall.

Logistic Regression Model

So now we will turn to Logistic Regression Model and see if it will perform better. As we have discovered that the most important variables in our dataset are merchant, amount, and category, we will now include these features in our model - that is, we are to predict fraud, given/based on the merchant, amount, and category:

```
fit_glm <- glm(fraud ~ ., train_data, family = "binomial")
```

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

```
p_hat_glm <- predict(fit_glm, test_data, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
y_hat_glm <- ifelse(p_hat_glm > 0.5, 1, 0) %>% factor()
```

The performance of our model will now be measured by Confusion Matrix:

```
confusionMatrix(data = y_hat_glm, reference = as.factor(test_data$fraud))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 117372    380
##          1    133   1044
##
##                Accuracy : 0.9957
##                  95% CI : (0.9953, 0.9961)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8006
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9989
##             Specificity : 0.7331
##          Pos Pred Value : 0.9968
##          Neg Pred Value : 0.8870
##              Prevalence : 0.9880
##          Detection Rate : 0.9869
##    Detection Prevalence : 0.9901
##       Balanced Accuracy : 0.8660
##
##        'Positive' Class : 0
##
```

The model also has high precision and recall so it works also fine on the unbalanced data. However, here in this case, we can see that the balanced accuracy slightly increases to about 0.87. And that it can predict fraud better (0.73). The number of fraud predicted as fraud increases its number to 1044, although on the contrary, its "non-fraud" predicted as "non-fraud" decreases (sensitivity). Which means, there is a risk that there are more non-fraudulent transactions will be predicted as fraud. Nevertheless, it is much better than having more real fraud slipped through being dismissed as non-fraud.

Results

Based on the experiments of different models and measuring their performance, it is suggested that the best prediction is produced using the **Logistic Regression Model** because it can work on unbalanced data (high recall and precision) and can better detect fraud than the decision tree, as well as has a higher balanced accuracy. First we notice that our data is very unbalanced and hence must be wary about it. Then we eliminate the irrelevant variables, and we take into account all the other variables until we found the most important variables. We then found that the Logistic Regression Model can work fine on the unbalanced data and produces a better prediction to detect fraud in our data. The final model can be demonstrated as it is being run on our train and test set, as follows:

```
fit_glm <- glm(fraud ~ merchant + amount + category, train_data, family = "binomial")
```

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

```
p_hat_glm <- predict(fit_glm, test_data, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
y_hat_glm <- ifelse(p_hat_glm > 0.5, 1, 0) %>% factor()
```

The prediction generated is:

```
head(y_hat_glm)
```

```
##  1 10 13 14 15 17
##  0  0  0  0  0  0
## Levels: 0 1
```

With the balanced accuracy of 0.87

```
confusionMatrix(data = y_hat_glm, reference = as.factor(test_data$fraud))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 117368    390
##          1    137   1034
##
##                Accuracy : 0.9956
##                  95% CI : (0.9952, 0.9959)
```

```
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7947
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9988
##             Specificity : 0.7261
##          Pos Pred Value : 0.9967
##          Neg Pred Value : 0.8830
##              Prevalence : 0.9880
##          Detection Rate : 0.9869
##    Detection Prevalence : 0.9902
##       Balanced Accuracy : 0.8625
##
##        'Positive' Class : 0
##
```

Conclusion

The core of the whole process in making a model to detect fraud in financial payment system is that we must build an algorithm with the past data we have collected (with all variables related to the transaction concerned) in order to detect future possible occurences of fraud. We have reached the conclusion that in improving our models to minimize errors, there are at least two things that are most essential to be taken into account while producing detection system: we need to discover the relevant variables, and take into account the unbalanced data we possibly have.

There are, however, some limitations on the models we develop. Some of the major limitations are: first, not all past fraud cases are or have been detected, hence our input data used to train and test might be flawed. Second, the most important variables detected by our algorithm might also be mistaken, since in real life not everything can be captured by data – fraud, in reality, is a human behaviour problem.

In the future, these kinds of models must also be actively analyzed by fraud experts, investigators, and behavioural analysts in order to better support data scientists in developing their models.