“Fraud Detection”

Ladan Foroughi

2021-09-20

Contents

List of Figures

List of Tables

Introduction

The Fraud Detection is a popular subject in different industry and organization as banking, insurance and etc. The current project utilizes simulated data generated by PaySim data simulator. This simulator uses financial logs of a mobile money service and then generates synthetic dataset of money transactions that resembles normal transactions. The simulator will then introduces fraudulent transactions for the purpose of evaluating fraud detection methods.

In this project, a 1/4 scaled-down subset of the original dataset is downloaded from Kaggle for data visualization purposes. For machine learning, this dataset is further downsized to allow the model to run within author's computer capabilities. The data were first inspected in order to understand the data pattern and dataset structure. Several plots have been created in order to visualize the relationship between account types, transaction time, amount transferred, difference balance with whether or not the transaction is marked as fraud.

Methodology

All the packages that we need during this project is installed from http://cran.us.r-project.org.

```
if(!require(pacman))install.packages("pacman")
pacman::p_load(
  tidyverse,
  dplyr,
  ggplot,
  caret,
  magnittr,
  pacman,
  GGally,
  knitr,
  parallel,
  rattel,
  tictoc,
  gridExtra,
  kableExtra,
  readr,
  purrr,
  randomForest,
  pROC,
  fastDummies,
  rpart.plot,
  data.table,
  reshape2,
  graphics,
  corrplot,
  latexpdf,
  ReporteRs,
  tinytex,
  latexdiffr,
  latex2exp
)
```

The dataset that is used in this project is downloaded from https://www.kaggle.com/ealaxi/paysim1.

```
temp <- tempfile()
url <- "https://www.kaggle.com/ealaxi/paysim1"
download.file(url,"temp")
rawdata <- fread("PS_20174392719_1491204439457_log.csv", header=TRUE)
unlink(temp)
Fraud <- rename(rawdata)
rm(rawdata,temp,url)
```

The specification of the Fraud dataset are summarized in below. The five first rows of Fraud dataset is shown in Table below. These dataset is consist of 1048575 data, and 11 variable.

```
kable(t(head(Fraud,5)),
      "pandoc",
      caption = "The Fraud Dataset",
      align = "c",
      font_size = 5)
```

Table 1: The Fraud Dataset

step	1	1	1	1	1
type	PAYMENT	PAYMENT	TRANSFER	CASH_OUT	PAYMENT
amount	9839.64	1864.28	181.00	181.00	11668.14
nameOrig	C1231006815	C1666544295	C1305486145	C840083671	C2048537720
oldbalanceOrg	170136	21249	181	181	41554
newbalanceOrig	160296.36	19384.72	0.00	0.00	29885.86
nameDest	M1979787155	M2044282225	C553264065	C38997010	M1230701703
oldbalanceDest	0	0	0	21182	0
newbalanceDest	0	0	0	0	0
isFraud	0	0	1	1	0
isFlaggedFraud	0	0	0	0	0

```
dim(Fraud)
```

```
## [1] 1048575      11
```

The name of variables are step, step , type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, isFlaggedFraud (Table below).

```
kable(names(Fraud),
      "pandoc",
      caption = "Name of variable of Fraud Dataset",
      align = "c",
      font_size = 5)
```

Table 2: Name of variable of Fraud Dataset

x
step
type
amount
nameOrig
oldbalanceOrg
newbalanceOrig
nameDest
oldbalanceDest
newbalanceDest
isFraud
isFlaggedFraud

Each variable is explained as below:

- Step:Maps a unit of time in the real world (step 1 is equal to 1 hour of time).
- type:CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
- amount:Amount of the transaction in local currency.
- nameOrig:Customer who started the transaction
- oldbalanceOrg:Initial balance before the transaction
- newbalanceOrig:New balance after the transaction
- nameDest:Customer who is the recipient of the transaction
- oldbalanceDest:Initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).
- newbalanceDest:New balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).
- isFraud: This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.
- isFlaggedFraud:The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

The summary of Fraud dataset shows there is no missing data, also there is no NA value as well.

```
kable(t(summary(Fraud)),
      "pandoc",
      caption = "Summary of Fraud Dataset",
      align = "c",
      font_size = 3)
```

Table 3: Summary of Fraud Dataset

step	Min. : 1.00	1st Qu.:15.00	Median :20.00	Mean :26.97	3rd Qu.:39.00	
type	Length:1048575	Class :character	Mode :character	NA	NA	
amount	Min. : 0	1st Qu.: 12149	Median : 76343	Mean : 158667	3rd Qu.: 213762	Ma
nameOrig	Length:1048575	Class :character	Mode :character	NA	NA	
oldbalanceOrg	Min. : 0	1st Qu.: 0	Median : 16002	Mean : 874010	3rd Qu.: 136642	Ma
newbalanceOrig	Min. : 0	1st Qu.: 0	Median : 0	Mean : 893809	3rd Qu.: 174600	Ma
nameDest	Length:1048575	Class :character	Mode :character	NA	NA	
oldbalanceDest	Min. : 0	1st Qu.: 0	Median : 126377	Mean : 978160	3rd Qu.: 915923	Ma
newbalanceDest	Min. : 0	1st Qu.: 0	Median : 218260	Mean : 1114198	3rd Qu.: 1149808	Ma
isFraud	Min. :0.000000	1st Qu.:0.000000	Median :0.000000	Mean :0.001089	3rd Qu.:0.000000	M
isFlaggedFraud	Min. :0	1st Qu.:0	Median :0	Mean :0	3rd Qu.:0	

```
any(is.na(Fraud))
```

```
## [1] FALSE
```

Analysis

We need to do some modification on Fraud dataset to better analysis. The step variable has to change to hour because it is easy to analysis, also the defined two variable as diff-balance (D.B) and diff-balance-recipient (D.B.R) that are difference between oldbalance and newbalance of original and recipient transaction.

```
Fraud <- Fraud %>%
  mutate(hour = step %% 24,
         diff_balance = oldbalanceOrg - newbalanceOrig,
         diff_balance_recipient = oldbalanceDest - newbalanceDest)
```

In this dataset the number of Fraud that reported is around 1142 (Table below).

```
number_of_Fraud <- Fraud %>% group_by(isFraud) %>% summarise(n = n())
kable(number_of_Fraud,
      "pandoc",
      caption = "Number of Fraud",
      align = "c",
      font_size = 5)
```

Table 4: Number of Fraud

isFraud	n
0	1047433
1	1142

Also there is no illegal attempt business accrued in these dataset report.

```
Number_of_illegal<- Fraud %>% group_by(isFlaggedFraud) %>% summarise(n = n())
kable(Number_of_illegal,"pandoc", caption = "Number of Illegal Business",
      align = "c")
```

Table 5: Number of Illegal Business

isFlaggedFraud	n
0	1048575

Effect of Type of Payment

The number of each type of Transaction are shown in Table below.The highest number of transactions is related to Cash_out type of payment (373641) and lowest number of transaction is for debit type of payment (7178).

```
Number_of_each_transaction <- Fraud %>%
                              group_by(type) %>%
                              summarise(n = n())
kable(Number_of_each_transaction,
      "pandoc",
      caption = "Number of Each Type of Transaction",
      align = "c")
```

Table 6: Number of Each Type of Transaction

type	n
CASH_IN	227130
CASH_OUT	373641
DEBIT	7178
PAYMENT	353873
TRANSFER	86753

The Figure below shows the logarithm of transformed number of each type of transaction based on occurred Fraud. It is shown that Fraud is occurred in two types of transaction as CASH-OUT and TRANSFER.

```
Fraud %>% group_by(type) %>%
  ggplot(aes(type, fill = factor(isFraud))) +
  geom_bar(stat = "count") + scale_y_log10() +
  scale_fill_manual(values = c("pink", "blue"))+
  xlab("Type of Transaction") +
  ylab("Log Transformed Number of Transaction") +
  theme(axis.text.x = element_text(angle = 90 ,hjust = 0.5))+
  guides(fill= guide_legend(title= "IsFraud"))
```

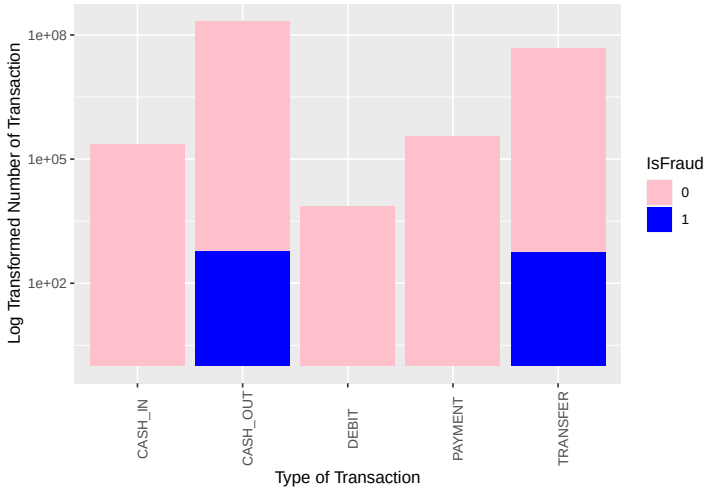

Figure 1: Number of Transaction for each Type

The number of Fraud in Cash_out and Transfer type of transaction are around 578 and 564 respectively (less than 1% of total transaction).

```
Number_of_Fraud_in_cashout_Transfer_type <- Fraud %>% filter(isFraud == 1) %>% group_by(type) %>%
  summarise(n = n())
kable(Number_of_Fraud_in_cashout_Transfer_type,
      "pandoc",
      caption = "Number of Fraud in Cash_out and Transfer type of Transaction",
      align = "c")
```

Table 7: Number of Fraud in Cash_out and Transfer type of Transaction

type	n
CASH_OUT	578
TRANSFER	564

Effect of Hour

Figure below shows that the number of transaction is low at initial hour (hour < 7), over time this number increased.

```
Fraud %>% group_by(hour) %>% filter(hour != 0) %>%
  summarise(n = n()) %>%
  ggplot(aes(hour, n)) +
  geom_col(fill= "blue" , col = "pink") +
  xlab("Hour") +
  ylab("Number of Transaction")
```

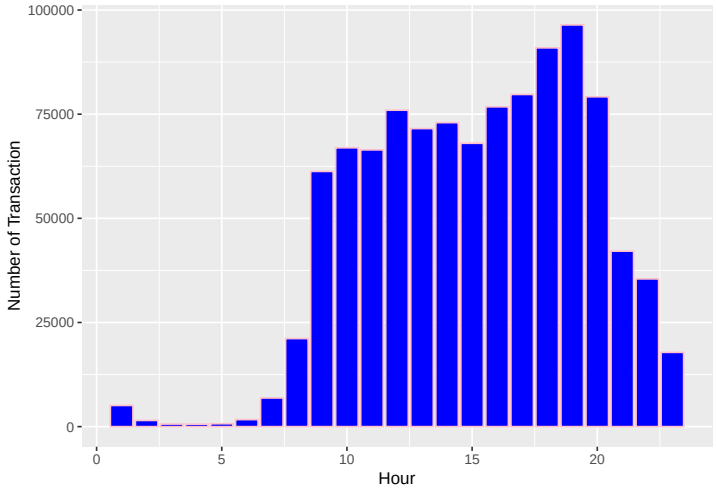

Figure 2: Number of Transaction versus of Hour

Since the fraud occurred only on two type of transaction. the effect of hours on these typs of transaction is studied in detail.

Figure below, shows that the number of fraud is approximate same during hours but the ratio between fraudulent on total number of transaction is more in hour between 2 to 6.

```
Fraud %>% group_by(hour) %>%
  filter(type %in% c("TRANSFER","CASH_OUT")) %>%
  ggplot(aes(hour, fill = factor(isFraud))) +
  scale_fill_manual(values = c("pink","blue"))+
  geom_bar(stat = "count")+ scale_y_log10() +
  facet_grid(type ~ .) +
  xlab("Hour") + ylab("Log Transformed Number of Transaction") +
  guides(fill = guide_legend(title = "isFraud"))
```

In Table below, the percentage of Fraud based on hour is shown. It is confirmed that between 2 to 6 hour, we have the most fraudulent around 12 to 26 %.

```
percentage_of_fraud_based_hour<- Fraud %>% mutate(hour = step %% 24) %>%
  filter(type %in% c("TRANSFER","CASH_OUT")) %>%
  group_by(hour) %>%
  summarise(Yes_Fraud = mean(isFraud == 1),
            No_Fraud = mean(isFraud == 0),
            percent_of_Number_of_Fraud = Yes_Fraud * 100/(Yes_Fraud+No_Fraud)) %>%
  arrange(desc(percent_of_Number_of_Fraud)) %>% select(hour, percent_of_Number_of_Fraud)
```

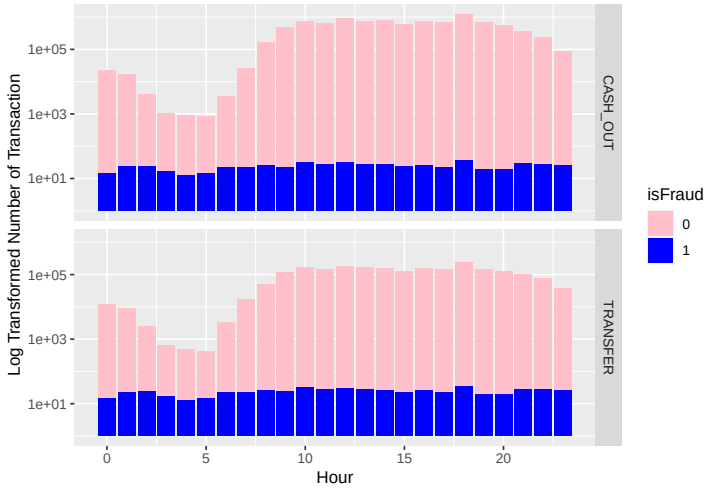

Figure 3: Number of Log Transformed of two type of Payment Transaction versus of Hour

```
kable(head(percentage_of_fraud_based_hour),
      "pandoc",
      caption = "percentage of fraud versus on hour of transaction in two method of payment")
```

Table 8: percentage of fraud versus on hour of transaction in two method of payment

hour	percent_of_Number_of_Fraud
5	26.08696
3	25.75758
4	19.25926
2	14.76923
6	12.29050
1	4.00000

The figure below is also confirmed these results.

```
percentage_of_fraud_based_hour %>%
  ggplot(aes(hour,percent_of_Number_of_Fraud )) +
  geom_point(size = 2, col = "blue") +
  xlab("Hour") + ylab("% of Fraudulent")
```

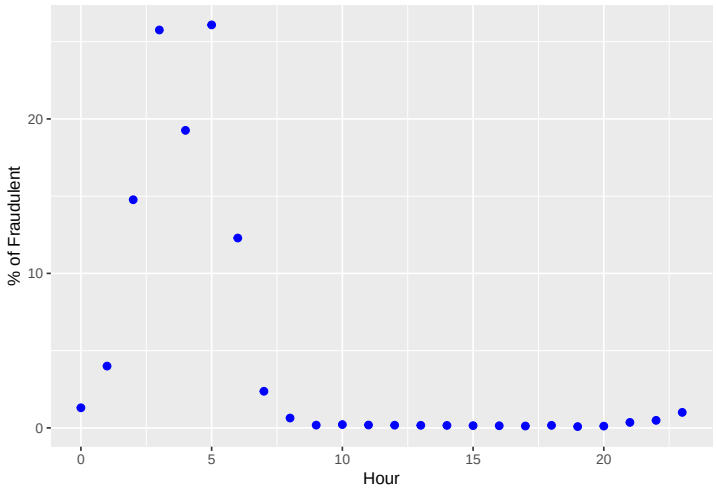

Figure 4: Variation of Fraudulent versus hour

Effect of Amount

The Density plot shows the number of transaction per amount of money taken out of the original account for fraudulent and none-fraudulent transactions. This plot shows that at low amounts the number of transaction is more and consequently the most fraudulent transactions occurred. Also, there is a group of fraudulent transactions that involve very large amounts.

```
Fraud %>% filter(type %in% c("CASH_OUT","TRANSFER")) %>%
  ggplot(aes(amount,fill= as.factor(isFraud))) +
  geom_density(alpha = 0.4)+
  scale_fill_manual(values = c("pink",'blue'))+
  xlab("Amount ($)") + ylab("Density")
```

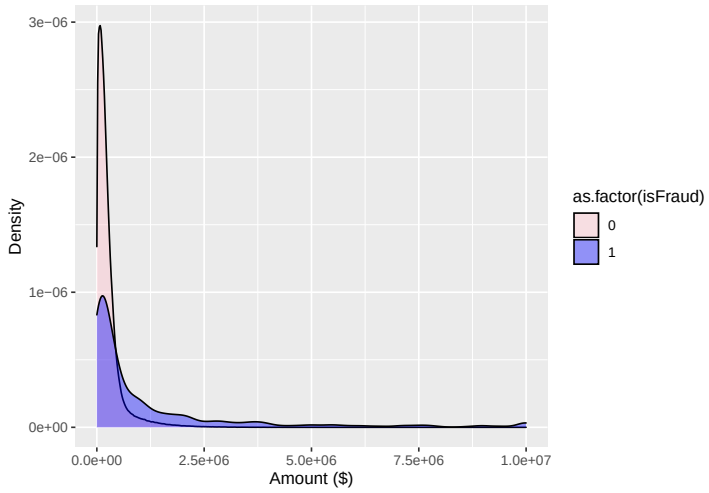

Figure 5: Distribution of Amount

The following boxplot shows that fraudulent transactions have higher median compared to none-fraudulent transactions.

```
Fraud %>% group_by(isFraud) %>%
  ggplot(aes(y = amount, fill = as.factor(isFraud))) +
  geom_boxplot() + scale_y_log10()+
  scale_fill_manual(values = c("pink","blue"))+
  guides(fill = guide_legend(title = "IsFraud"))+
  ylab("Amount") + xlab("IsFraud") +
  theme(axis.text.x = element_blank())+
  theme(plot.title = element_text(hjust = 0.5))
```

The amount of fraudulent is approximately constant during the hour for both of type of Transaction (Figure below).

```
Fraud %>%
  filter(type %in% c("CASH_OUT","TRANSFER")) %>%
  mutate(hour = step %% 24) %>%
  filter(isFraud == 1) %>%
  group_by(hour) %>%
  ggplot(aes(hour, amount, fill =type)) +
  geom_col() +
  scale_y_log10() +
```

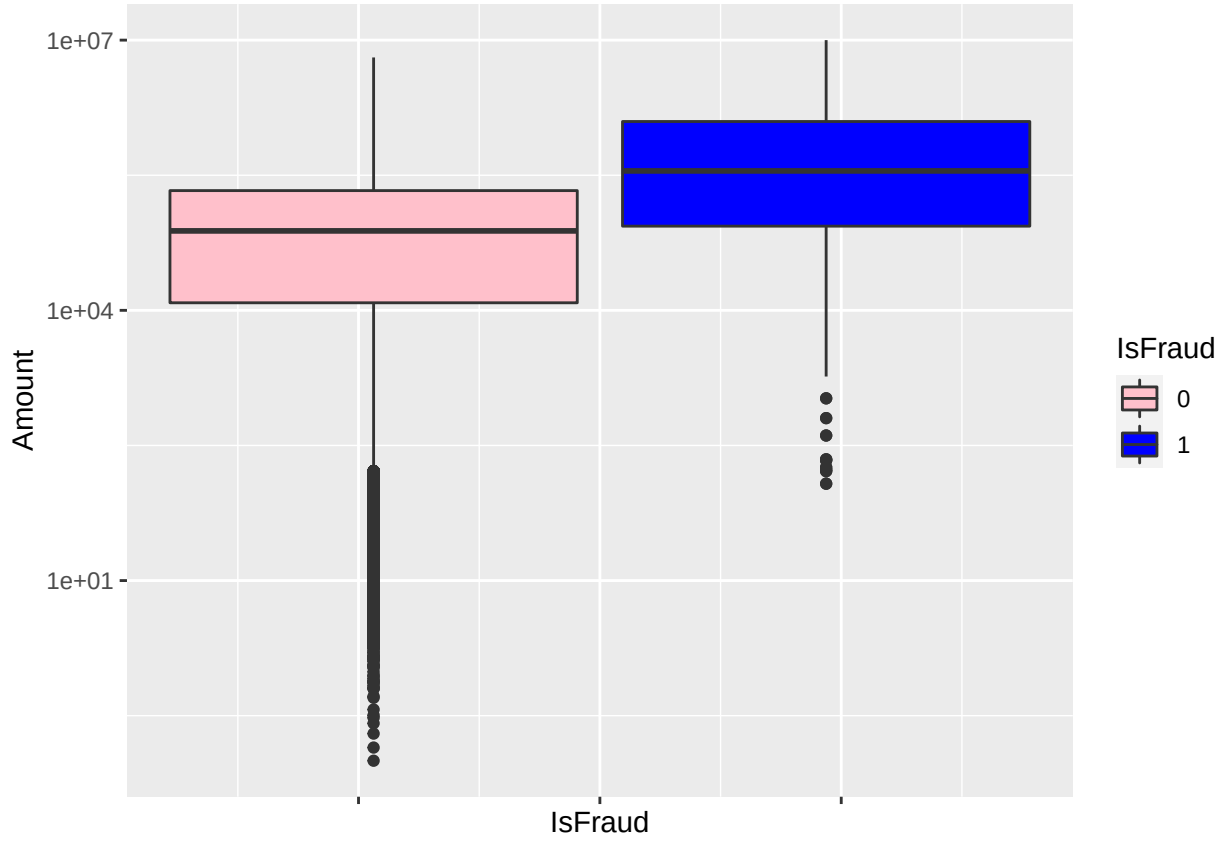

Figure 6: Amount of fraudulent vs non-fradulent Transaction

```
theme(legend.text = element_blank())+
facet_grid(type ~.)+
xlab("hour(hr)") + ylab("Log of Amount of Fraudulent")
```

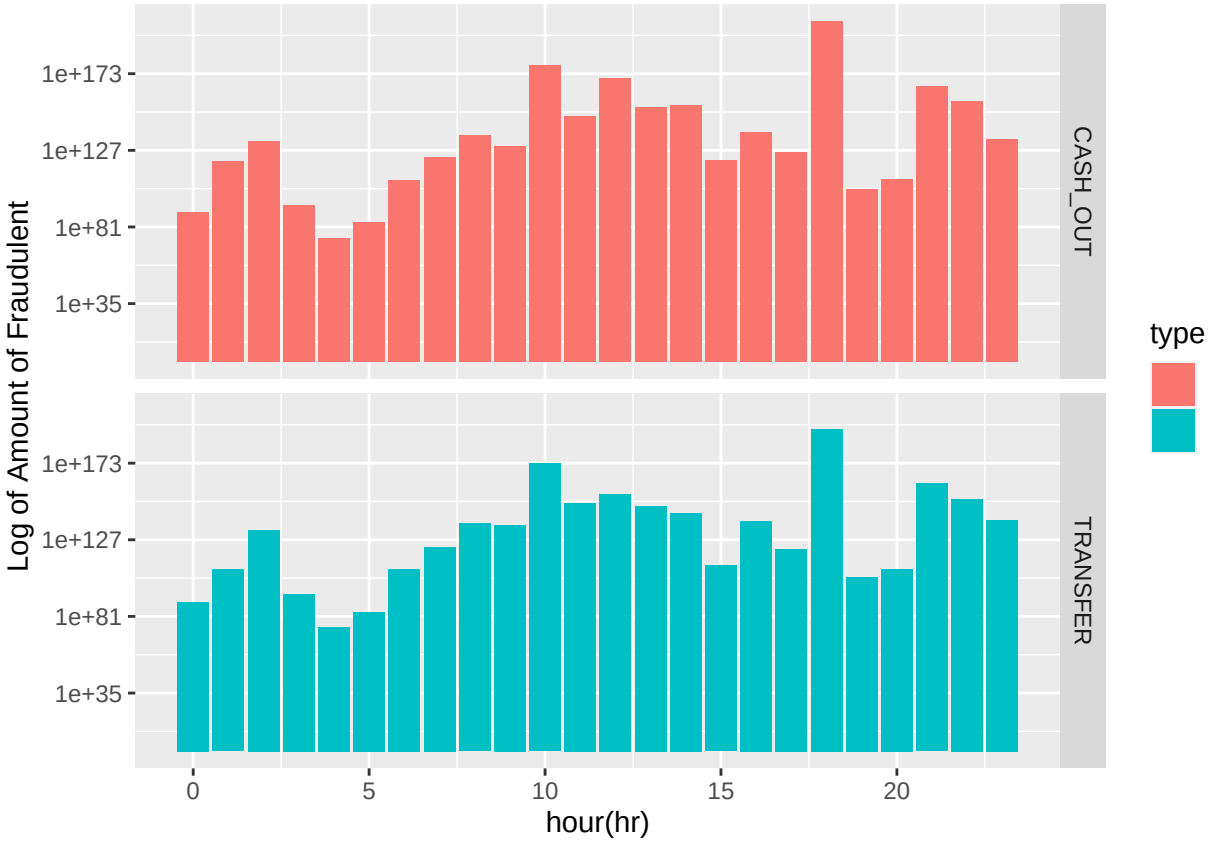

Figure 7: Amount of Fraudulent versus hours

Effect of Difference Balance and Differenc Balance Recipient

The distribution of Difference balance (D.B.) and Difference Balance Recipient (D.B.R.) for Fraudulent in two type of Transaction is shown in Figure below . It is Clear the Difference Balance of Fraudulent is only in Transfer Type, But the Difference Balance Recipient Fraudulent in both type of Transaction. Also at lower difference Balance and Difference Balance Recipient amount the intensity of Fraud is higher.

```
D_B_density <- Fraud %>%
  filter(type %in% c("CASH_OUT","TRANSFER") &
           diff_balance != 0) %>%
  filter(isFraud ==1) %>%
  ggplot(aes(diff_balance, fill = type)) +
  geom_density(alpha = 0.1) +
  scale_fill_manual(values = c("pink","blue")) +
  xlab("D.B.($)") +
  ylab("Density")

D_b_R_density <- Fraud %>%
  filter(type %in% c("CASH_OUT","TRANSFER"),
         diff_balance_recipient != 0) %>%
  filter(isFraud ==1) %>%
  ggplot(aes(diff_balance_recipient, fill = type)) +
  geom_density(alpha = 0.1)+
  scale_fill_manual(values = c("pink","blue")) +
  xlab("D.B.R. ($)") +
  ylab("Density")
```

```
grid.arrange(D_B_density,D_b_R_density)
```

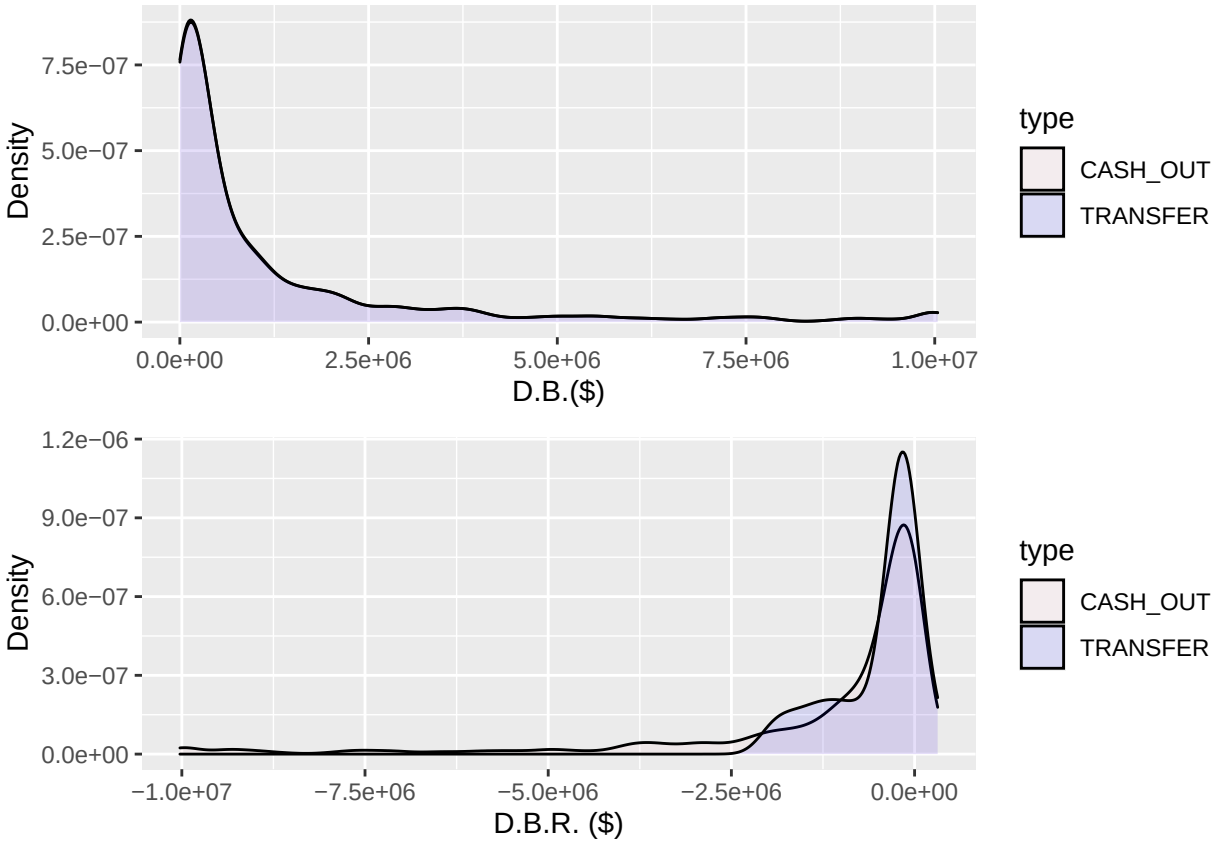

Figure 8: Distribution of Difference Balance(D.B.) and Difference Balance Recipient (D.B.R.)

The effect of hour on Difference Balance (D.B.) and Difference Balance recipient (D.B.R.) amount of Fraudulent of two types of Transaction is shown in Figure below. The results shows that hours did not effect on the D.B. and D.B.R. of two type of transaction.

```
D_B_vs_hour<- Fraud %>%
  filter(type %in% c("CASH_OUT","TRANSFER") &
         diff_balance != 0) %>%
  group_by(hour) %>%
  filter(isFraud == 1) %>%
  mutate(diff_balance_avg = mean(diff_balance)) %>%
  ggplot(aes(hour,diff_balance_avg)) +
  geom_col(fill = "pink")+
  facet_grid( .~ type) +
  xlab("hour(hr)") +
  ylab("D.B amount($)")

D_B_R_vs_hour <- Fraud %>%
  filter(type %in% c("CASH_OUT","TRANSFER") &
           diff_balance_recipient != 0) %>%
  group_by(hour) %>%
  filter(isFraud == 1) %>%
  mutate(diff_balance_recipient_avg = mean(diff_balance_recipient)) %>%
  ggplot(aes(hour,diff_balance_recipient_avg)) +
  geom_col(fill = "blue") +
  scale_y_reverse()+
  facet_grid( .~ type) +
  xlab("hour(hr)") +
  ylab("D.B.R amount($)")

grid.arrange(D_B_vs_hour, D_B_R_vs_hour)
```

In general, the correlation of parameters as hour, amount, diff-balance, diff_balance_recipient and isFraud is shown in Figure below.

```
Cor_for_Fraud <- Fraud %>%
  filter(type %in% c("CASH_OUT","TRANSFER")) %>%
            select(hour, amount, diff_balance,
                   diff_balance_recipient,isFraud)

r<- cor(Cor_for_Fraud, method = "pearson")
corrplot(r, tl.col = "black", tl.srt = 45, bg = "light blue",
         #title = "\n\n Correlation Plot Of Fraud Detection",
         type = "full")
```

As shown in analysis section, the variable as isFlaggedFraud, nameOrig and nameDest didnot have bearing in idetifing the fraudulent. Also the variables as step, oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest modified to some variables that is usefull in analysis. In addition, the two type of transaction as Cash-out and Transfer are associat in fraudulent transaction. Then the Fraud dataset that is used for modeling in next section is representing below.

```
Fraud <- Fraud %>% filter(type %in% c("CASH_OUT","TRANSFER")) %>%
  mutate(isFraud= as.factor(isFraud)) %>%
```

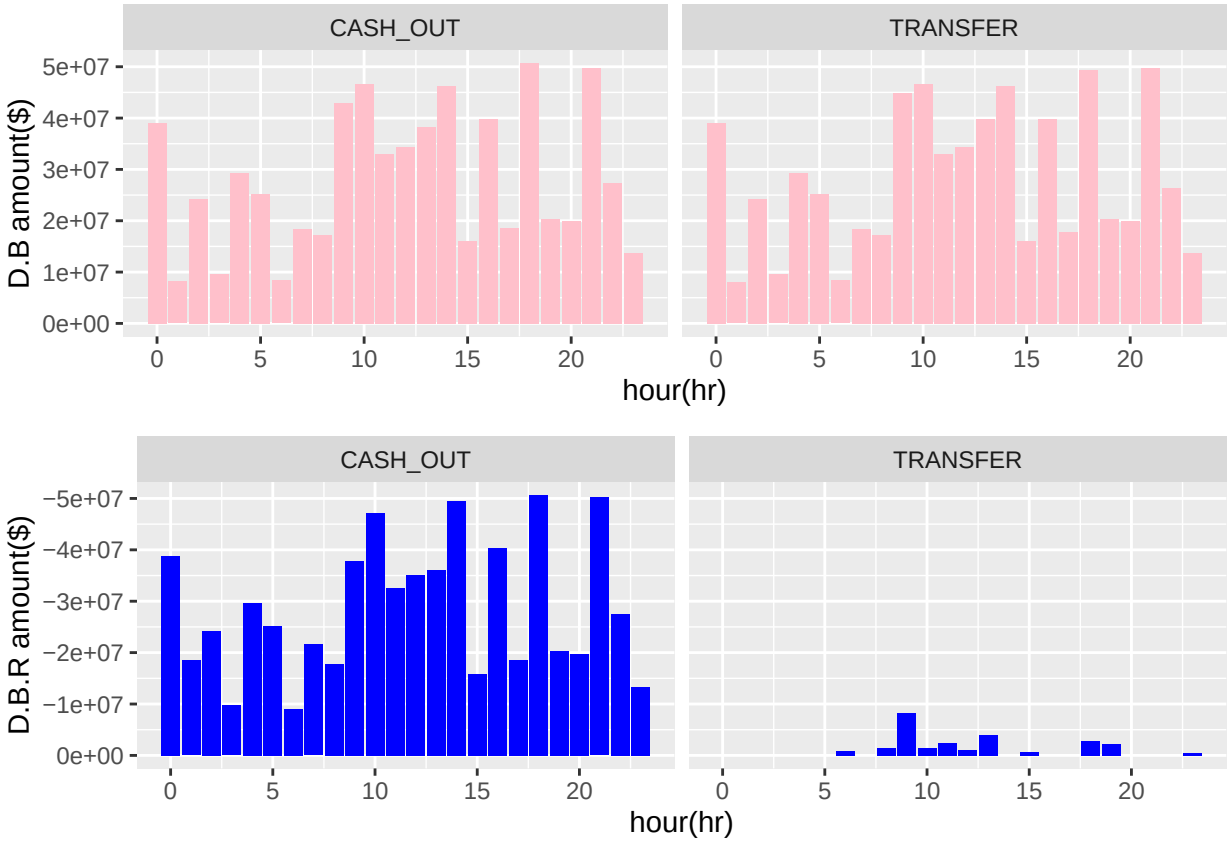

Figure 9: Difference Balance(D.B.) and Difference Balance Recipient (D.B.R.) versus hour

Figure 10: Correlation Plot Of Fraud Detection

```
  select(hour, type, amount, diff_balance, diff_balance_recipient,
         isFraud)
```

Results

The Fraud dataset is too large for the computer used by the author for the machine learning purposes. The machine learning algorithms take either too long or will not run at all indicating memory reached. Therefore, for these reason, only 40% of the original dataset is used. Note that for more powerful computers the following code can be removed.

```
set.seed(1)
dat <- sample_frac(Fraud, size = 0.4, replace = FALSE)
```

For starting the modeling of these dataset, the Fraud dataset is sliced to part with proportion of 80/20 percent, as train set and validation set. we assume that the outcomes of validation set is unknown.

```
index <- createDataPartition(dat$isFraud, times = 1, p = 0.2, list = FALSE)
train <- dat %>% slice(-index)
validation <- dat %>% slice(index)
```

The training dataset is further divided into training_set_split and test_set to avoid over training.

```
index_test <- createDataPartition(train$isFraud, times = 1, p= 0.2, list = FALSE)
train_set <- train %>% slice(-index_test)
test_set <- train %>% slice(index_test)
```

The number of each splitting datasets as train, validation, train_set, test_set are in Table below ,

```
train_row = nrow(train)
validation_row = nrow(validation)
train_set_row = nrow(train_set)
test_set_row = nrow(test_set)
kable(rbind(train_row,validation_row,train_set_row,test_set_row),
      "pandoc",
      caption = "The number of Rows of each Datasets",
      align = "c")
```

Table 9: The number of Rows of each Datasets

train_row	147326
validation_row	36832
train_set_row	117860
test_set_row	29466

Guessing method

As a base model, we will guess the outcome as follows:

```
set.seed(1, sample.kind = "Rounding")
y_hat_gussing <- sample(c("1","0"),
                        length(index_test),
                        replace = TRUE) %>% factor()
```

The overall accuracy, sensitivity, specificity and F1 are the overall proportion that is predicted correctly. This will be extracted from the confusion matrix and F-meas as follows:

```
y_hat_gussing <- sample(c("1","0"),
                        length(index_test),
                        replace = TRUE) %>% factor()
Accuracy_gussing <-confusionMatrix(y_hat_gussing,test_set$isFraud)$overall["Accuracy"]
sensitivity_gussing <-sensitivity(y_hat_gussing,test_set$isFraud)
specificity_gussing <-specificity(y_hat_gussing, test_set$isFraud)
F_1 <- F_meas(y_hat_gussing,test_set$isFraud)

Results <- tibble(method = "Guessing",
                  Accuracy = Accuracy_gussing,
                  sensitivity = sensitivity_gussing,
                  specificity = specificity_gussing,
                  F_1 = F_1)
kable((Results[1:1,]),
      "pandoc",
      caption = "Model Specification- Guessing",
      align = "c")
```

Table 10: Model Specification- Guessing

method	Accuracy	sensitivity	specificity	F_1
Guessing	0.5058712	0.5059704	0.4647887	0.6713763

The results show that the accuracy is not high, as well as sensitivity and specificity.

Effect of Hour on Accuracy

Based on results from analysis section, it is established that most fraudulent activities occur overnight. So, we might be able to predict better, if we incorporate time into our predictions as follows:

```
y_hat_timeeffect <- ifelse(test_set$hour>=2 & test_set$hour <= 6,"1","0") %>% factor()
Accuracy_timeeffect <- confusionMatrix(y_hat_timeeffect,test_set$isFraud)$overall["Accuracy"]
sensitivity_timeeffect <- sensitivity(y_hat_timeeffect,test_set$isFraud)
specificity_timeeffect <- specificity(y_hat_timeeffect,test_set$isFraud)
F_1_timeeffect <- F_meas(y_hat_timeeffect, test_set$isFraud)
Results <- add_row(Results,
                   method = "Timeeffect",
                   Accuracy = Accuracy_timeeffect,
                   sensitivity = sensitivity_timeeffect,
                   specificity = specificity_timeeffect,
                   F_1 = F_1_timeeffect)

kable((Results[1:2,]),
      "pandoc",
      caption = "Model Specification-Guessing, TimeEffect",
      align = "c")
```

Table 11: Model Specification-Guessing, TimeEffect

method	Accuracy	sensitivity	specificity	F_1
Guessing	0.5058712	0.5059704	0.4647887	0.6713763
Timeeffect	0.9961990	0.9981630	0.1830986	0.9980950

Table below, shows using very simple model can be improved the accuracy. Also, sensitivity, which is defined as the ability of an algorithm to predict a positive outcome when the actual outcome is positive, has also improved. That is this algorithm correctly identifies frauds as frauds.

The specificity of this model is very low. As there are a lot more non-fraudulent transactions compared to fraudulent transactions, this model mistakenly predicts non-fraudulent transactions as fraud.

Although the sensitivity is more favor than specificity in this project, because it is important to identify fraudulent transactions correctly in expense of having some legal transactions being blocked. Otherwise we try to improve both sensitivity and specificity.

Rpart Classification Model

In rpart algorithm, the dataset is recursive by spliting. This means that the resulted subsets is split until a criterion is reached. The dataset is split until possible reduction is reached for the dependent variable. The algorithm is tuned for the complexity parameter that maximizes the model accuracy.

```
rpart <- train(isFraud ~ .,
               method = "rpart",
               tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)),
               data = train_set)
```

The following plot shows the rpart decision tree.

```
rpart.plot(rpart$finalModel,
           type = 5)
```

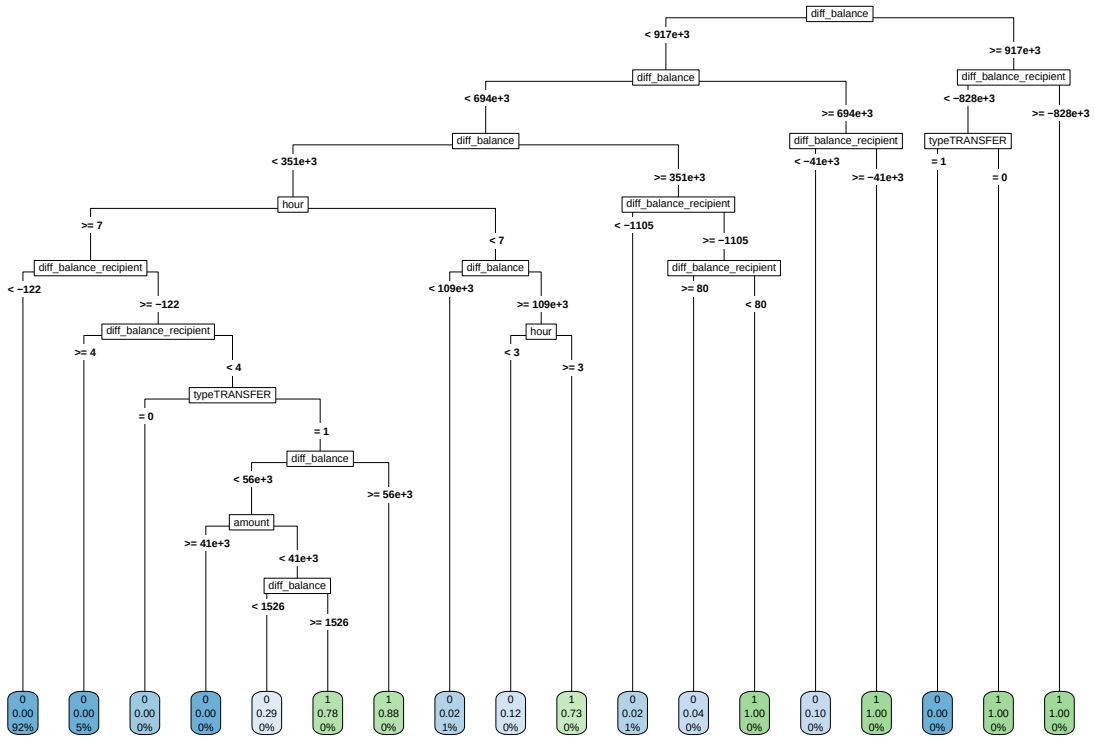

Figure 11: Decision Tree

Table below shows , Rpart model has significantly been improved on accuracy, sensitivity and specificity. This model is based on only one tree. Usually, a combination of a number of trees will help improve our predictions. Therefore, the Random Forest method is recommended.

```
y_hat_rpart <- predict(rpart, test_set)

Accuracy_rpart <-confusionMatrix(y_hat_rpart,test_set$isFraud)$overall["Accuracy"]
sensitivity_rpart <- sensitivity(y_hat_rpart,test_set$isFraud)
specificity_rpart <- specificity(y_hat_rpart,test_set$isFraud)
```

```
F_1_rpart <- F_meas(y_hat_rpart, test_set$isFraud)

Results <- add_row(Results,
                   method = "Rpart",
                   Accuracy = Accuracy_rpart,
                   sensitivity = sensitivity_rpart,
                   specificity = specificity_rpart,
                   F_1 = F_1_rpart)

kable((Results[1:3,]),
      "pandoc",
      caption = "Model Specification-Guessing, Timeeffect, Rpart",       align = "c")
```

Table 12: Model Specification-Guessing, Timeeffect, Rpart

method	Accuracy	sensitivity	specificity	F_1
Guessing	0.5058712	0.5059704	0.4647887	0.6713763
Timeeffect	0.9961990	0.9981630	0.1830986	0.9980950
Rpart	0.9987104	0.9998639	0.5211268	0.9993540

The most important parameters for Rpart predictions are plotted in Figure below. And model parameters are calculated (as shown in Figure below).

```
ggplot(varImp(rpart)) +
  theme(plot.title = element_text(hjust = 0.5))
```

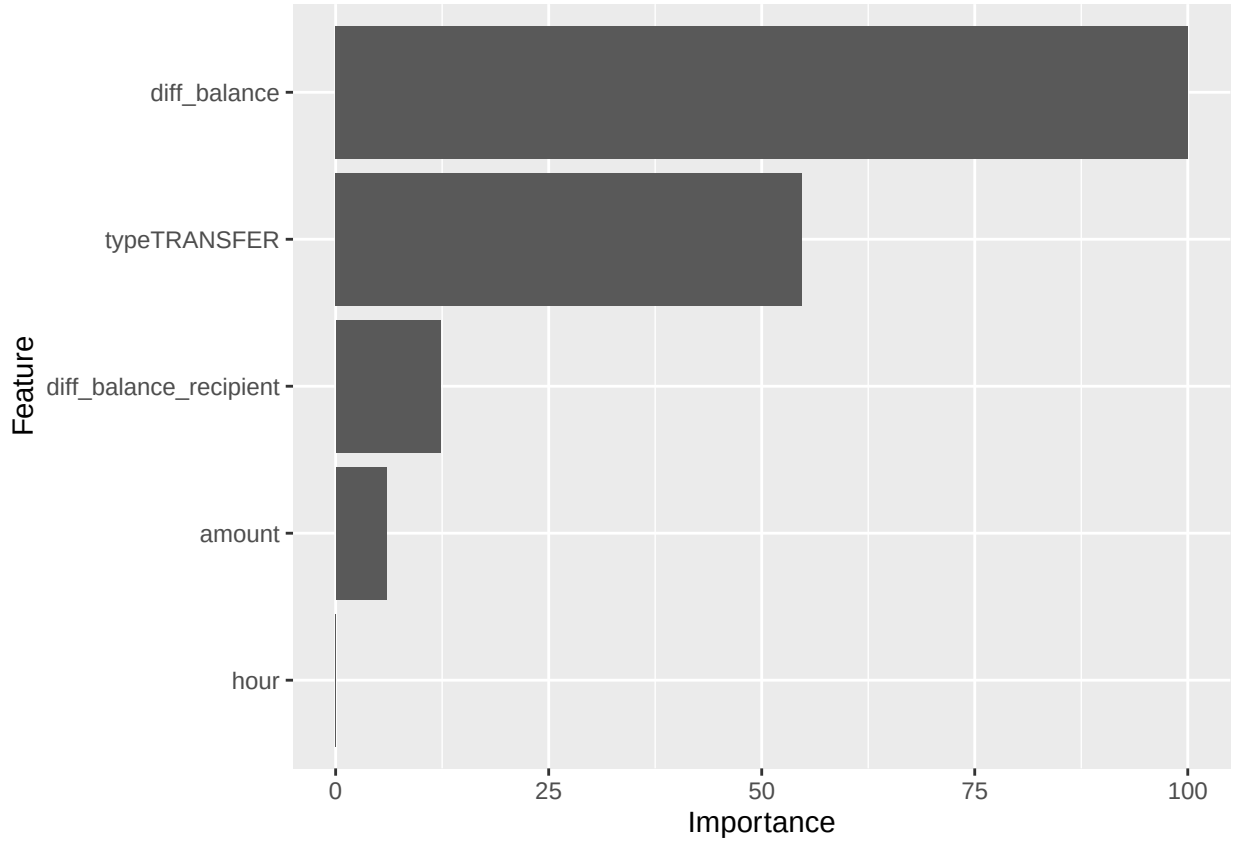

Figure 12: Predictive importance of variables in Rpart method

Random Forest Model

The random forest model uses multiple decision trees rather than using only one decision tree as in rpart model. The term “random” is used because training dataset is randomly sampled and the predictors are randomly selected. The model is trained for “mtry” tuning parameter.

```
set.seed(1)
Random_forest <- train(isFraud ~ .,
                       data = train_set,
                       trControl = trainControl(method = "cv", number = 5),
                       importance = TRUE,
                       method = "rf",
                       ntree = 25,
                       tuneGrid = data.frame(mtry = seq(1,6,1)))
```

The most important parameters for random forest predictions are plotted. And model parameters are calculated. Figure below shows that difference balance has the maximum effect, then type of Transfer and other parameters have less than 50 % importance.

```
ggplot(varImp(Random_forest)) +
  theme(plot.title = element_text(hjust = 0.5))
```

Accuracy, sensitivity, and specificity has improved even more compared to rpart decision tree. Therefore, this algorithm is selected for prediction.

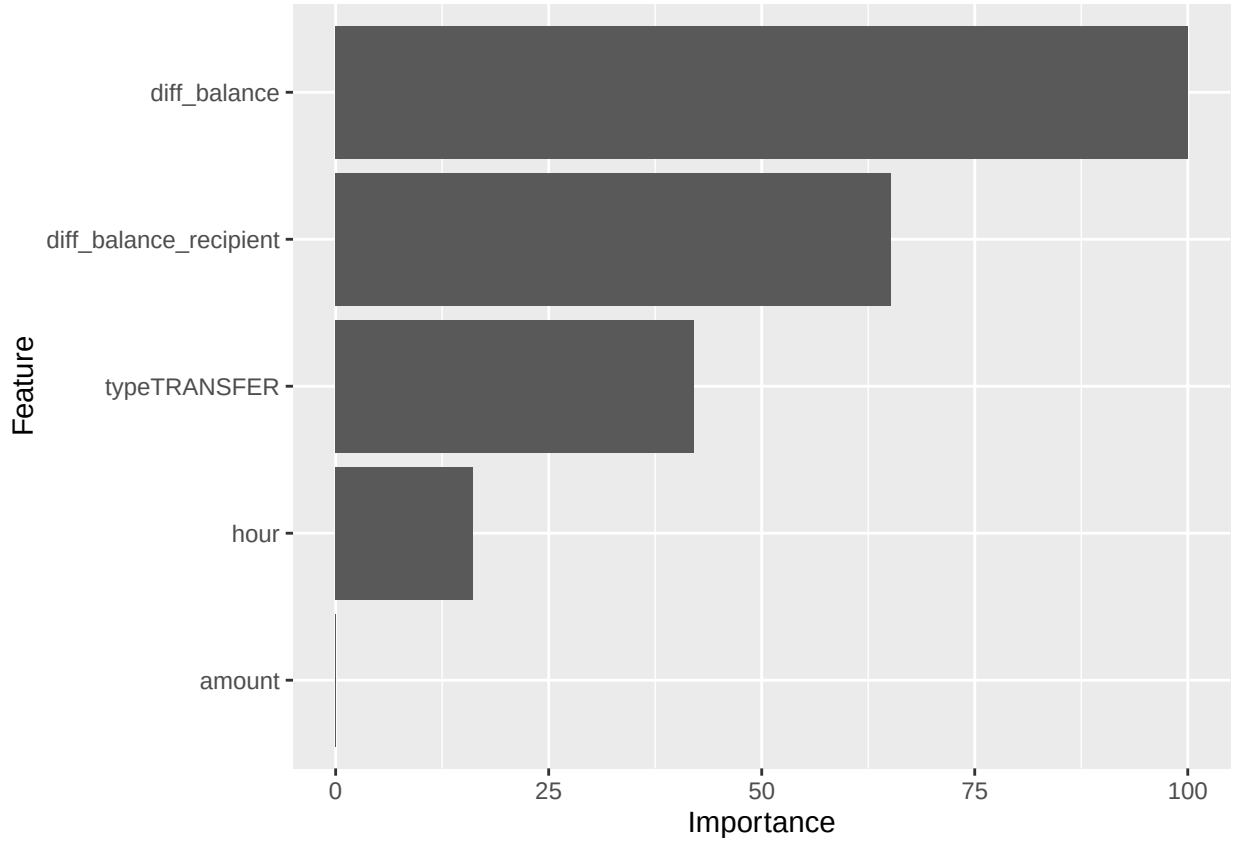

Figure 13: Predictive importance of variables in Random Forest

```
y_hat_rf <- predict(Random_forest, test_set)

Accuracy_rf <- confusionMatrix(y_hat_rf,test_set$isFraud)$overall["Accuracy"]
sensitivity_rf <- sensitivity(y_hat_rf,test_set$isFraud)
specificity_rf <- specificity(y_hat_rf,test_set$isFraud)
F_1_rf <- F_meas(y_hat_rf, test_set$isFraud)

Results <- add_row(Results,
                   method = "Random Forest",
                   Accuracy = Accuracy_rf,
                   sensitivity = sensitivity_rf,
                   specificity = specificity_rf,
                   F_1 = F_1_rf)
kable((Results[1:4,]),
      "pandoc",
      caption = "Model Specification-Guessing, Timeeffect, rpart,             RandomForest",
      align = "c")
```

Table 13: Model Specification-Guessing, Timeeffect, rpart, RandomForest

method	Accuracy	sensitivity	specificity	F_1
Guessing	0.5058712	0.5059704	0.4647887	0.6713763
Timeeffect	0.9961990	0.9981630	0.1830986	0.9980950

method	Accuracy	sensitivity	specificity	F_1
Rpart	0.9987104	0.9998639	0.5211268	0.9993540
Random Forest	0.9989819	0.9998639	0.6338028	0.9994899

Random Forest for Validation

After choosing the desired algorithm, the Rrandom Forest is tested on the validation set to predict a sample not used in the training. The accuracy, sensitivity, and specificity is then calculated for validation set through the code below:

```
set.seed(1, sample.kind = "Rounding")
y_hat_rf_validation <- predict(Random_forest, validation)

Accuracy_rf_validation <- confusionMatrix(y_hat_rf_validation, validation$isFraud)$overall["Accuracy"
sensitivity_rf_validation <- sensitivity(y_hat_rf_validation, validation$isFraud)
specificity_rf_validation <- specificity(y_hat_rf_validation, validation$isFraud)
F_1_rf_validation <- F_meas(y_hat_rf_validation, validation$isFraud)

Results <- add_row(Results,
                   method = "Random Forest-validation",
                   Accuracy = Accuracy_rf_validation,
                   sensitivity = sensitivity_rf_validation,
                   specificity = specificity_rf_validation,
                   F_1 = F_1_rf_validation)
kable((Results[1:5,]),
      "pandoc",
      caption = "Model Specification",
      align = "c")
```

Table 14: Model Specification

method	Accuracy	sensitivity	specificity	F_1
Guessing	0.5058712	0.5059704	0.4647887	0.6713763
Timeeffect	0.9961990	0.9981630	0.1830986	0.9980950
Rpart	0.9987104	0.9998639	0.5211268	0.9993540
Random Forest	0.9989819	0.9998639	0.6338028	0.9994899
Random Forest-validation	0.9989140	0.9997006	0.6704545	0.9994558

Conclusion

The Random Forest model is proved to be the best algorithm that can accurately predict fraudulent and non-fraudulent activities. Both sensitivity and specificity has much improved in this model compared to the other methods discussed in this project. The model specificity is not perfect and still is at 61% , however, it is better to block suspicious activities, rather than allowing fraudulent transactions to happen.

Acknowledgement

I would like to thank Dr. Rafael Irizarry and all colleagues for providing this absolutely valuable resource. I would also thank all the fellow students who take the time to read and grade this project.