HarvardX: PH125.9x Data Science Capstone - Fraud Detection

Golpira Elmi Assadzadeh, Ph.D.

17/04/2020

PROJECT OVERVIEW

The private nature of financial transactions has lead to insufficient data available for researchers specifically in fraud detection domain. As a result, 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 25% 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, original, and destination account balances with whether or not the transaction is marked as fraud.

The dataset were then modified to exclude variables that have no bearing on transactions being fraudulent or not (e.g., original account name). The modified dataset is then splitted into training and validation sets. In order to avoid overtraining, the training dataset is further splitted into "training set split" and "test set". Several different algorithms were trained using the "training set split" to predict the fraudulent activities. As the aim of this model is a binary classification (whether a tranaction is fraud or not), the accuracy, sensitivity, and specificity of the predictive algorithms are compared to one another in order to choose the algorithm of choice. The final algorithm was then tested on validation (unkown) set.

The following code will download the dataset:

```
# Dataset Source

# https://www.kaggle.com/ntnu-testimon/paysim1
# https://www.kaggle.com/ntnu-testimon/paysim1/download/WFkgBkx3T4g8fI1piSWE%2Fversions%2F

# Downloading dataset
temp <- tempfile()
url <- "https://drive.google.com/open?id=1cvul7cKtTObr-Ja-UvwO46rZ8z2IqQUf"
download.file(url, "temp")
rawdata <- fread("PS_20174392719_1491204439457_log.csv",
    header = TRUE)
unlink(temp)

# Number of digits to report
options(digits = 7)
```

METHOD AND ANALYSIS

Inspecting and visualizing the dataset

A few rows from edx dataset is printerd in order to identify its variables. The datsset contains 11 columns as follows:

```
kable(head(rawdata), "latex", longtable = T, booktabs = T,
    caption = "Fraud Prediction Dataset") %>% kable_styling(font_size = 7)
```

Table 1: Fraud Prediction Dataset

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	PAYMENT	9839.64	C1231006815	170136	160296.36	M1979787155	0	0	0	0
1	PAYMENT	1864.28	C1666544295	21249	19384.72	M2044282225	0	0	0	0
1	TRANSFER	181.00	C1305486145	181	0.00	C553264065	0	0	1	0
1	CASH_OUT	181.00	C840083671	181	0.00	C38997010	21182	0	1	0
1	PAYMENT	11668.14	C2048537720	41554	29885.86	M1230701703	0	0	0	0
1	PAYMENT	7817.71	C90045638	53860	46042.29	M573487274	0	0	0	0

The column descriptions has been provided by Kaggle as follows:

step - Unit of time in the real world. In this case 1 step is 1 hour of time.

type - Transaction type

amount - Amount of the transaction

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 of recipient before the transaction

newbalanceDest - New balance recipient after the transaction

isFraud - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behaviour 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 200000 in a single transaction.

The dataset has 6362620 rows with no missing values as shown in the summary table:

```
# Total rows in the database
total_transact <- rawdata %>% nrow()
total_transact
```

```
## [1] 6362620
```

```
# Any missing data
any(is.na(rawdata))
```

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

The difference between new balance and old balance is calculated for both origin accounts and recipient accounts. In addition, the variable "step" is converted to "hour" to show the time of transactions in hour.

```
rawdata <- rawdata %>% mutate(trans_orig = newbalanceOrig -
    oldbalanceOrg, trans_dest = newbalanceDest - oldbalanceDest,
    hour = step%%24)
```

A subset of data with only fraudulent transactions is also calculated as follows:

```
# Dataset of fraudulant transactions
fraud_data <- rawdata %>% filter(isFraud == 1)
```

Only 0.13% of all tranactions are fraudulent, as shown in the following code:

```
# Calculating fraud transaction percentage
number_of_fraud_transact <- fraud_data %>% nrow()
fraud_percentage <- (number_of_fraud_transact/total_transact) *
    100
fraud_percentage
```

```
## [1] 0.129082
```

Transaction Type Effect

The total number of transactions per type is calculated:

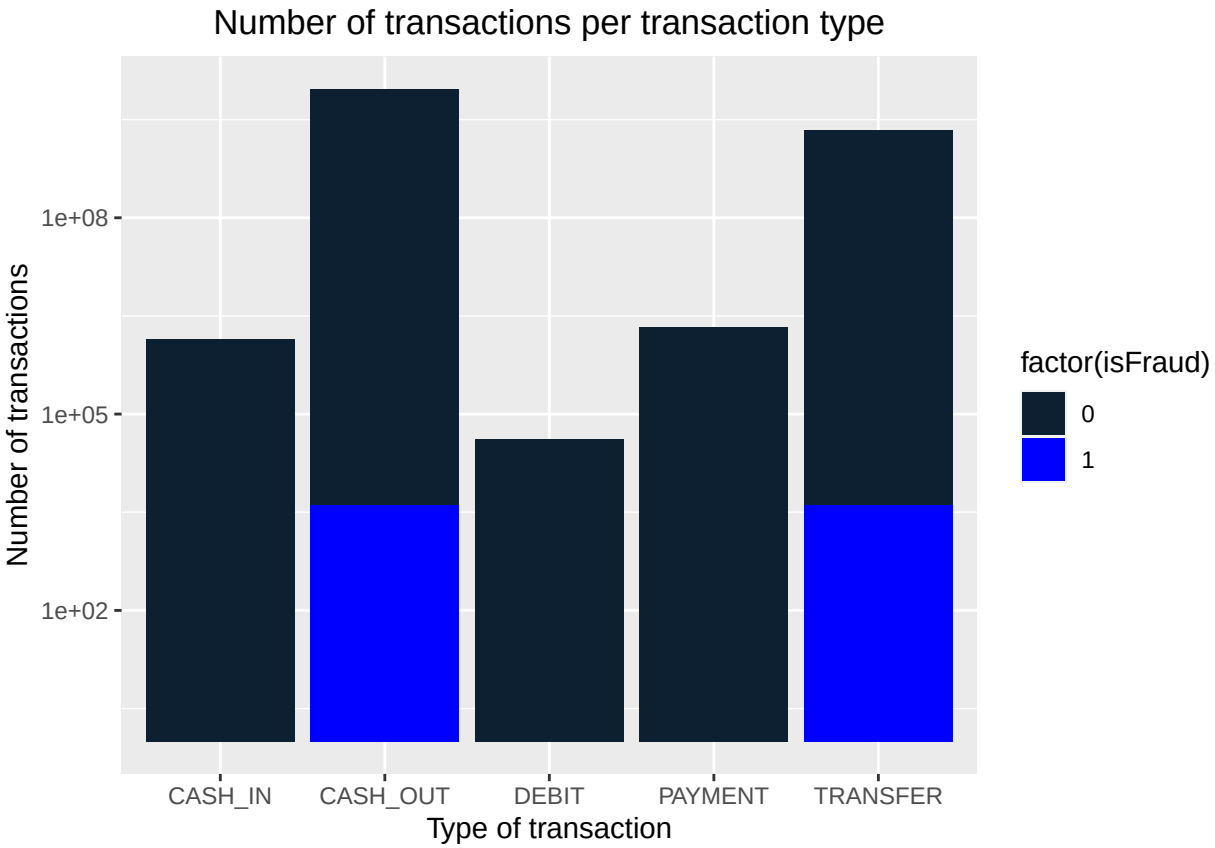
```
# Number of each type of transaction

number_transaction_per_type <- rawdata %>% group_by(type) %>%
    summarize(len = length(type))

kable(number_transaction_per_type, "latex", caption = "Total Number of transaction by type"
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 2: Total Number of transaction by type

type	len
CASH_IN	1399284
CASH_OUT	2237500
DEBIT	41432
PAYMENT	2151495
TRANSFER	532909

This code is plotting the number of transactions in each type:

```
# Number of transactions in each type
rawdata %>% ggplot(aes(x = type, fill = factor(isFraud))) +
    geom_bar() + scale_y_log10() + scale_fill_manual(values = c("#0c2032",
    "blue")) + xlab("Type of transaction") + ylab("Number of transactions") +
    ggtitle("Number of transactions per transaction type") +
    theme(plot.title = element_text(hjust = 0.5))
```



The number of fraudulent activities per transaction type is shown in the following table. This table indicates that fraudulent transactions only occured in “cash-out” and “transfer” transactions with almost equal proportion.

```
# Number of fradulent transaction by type
fraud_by_type <- fraud_data %>% group_by(type) %>%
    summarize(n = n())

kable(fraud_by_type, "latex", caption = "Number of fradulent transaction by type") %>%
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 3: Number of fradulent transaction by type

type	n
CASH_OUT	4116
TRANSFER	4097

The following code shows that there are only 16 transactions from total of 8213 fradulent transactions that were flagged as fraud (attempt to withdraw over 200000 dollars), suggesting that the marking transaction amounts of over 200000 as fraud is not an accurate criteria.

```
# Total number of fraudulent transactions
fraud_data %>% nrow()
```

```
## [1] 8213
```

```
# Number of flagged as fraud that are fraud
rawdata %>% filter(isFlaggedFraud == 1 & isFraud ==
```

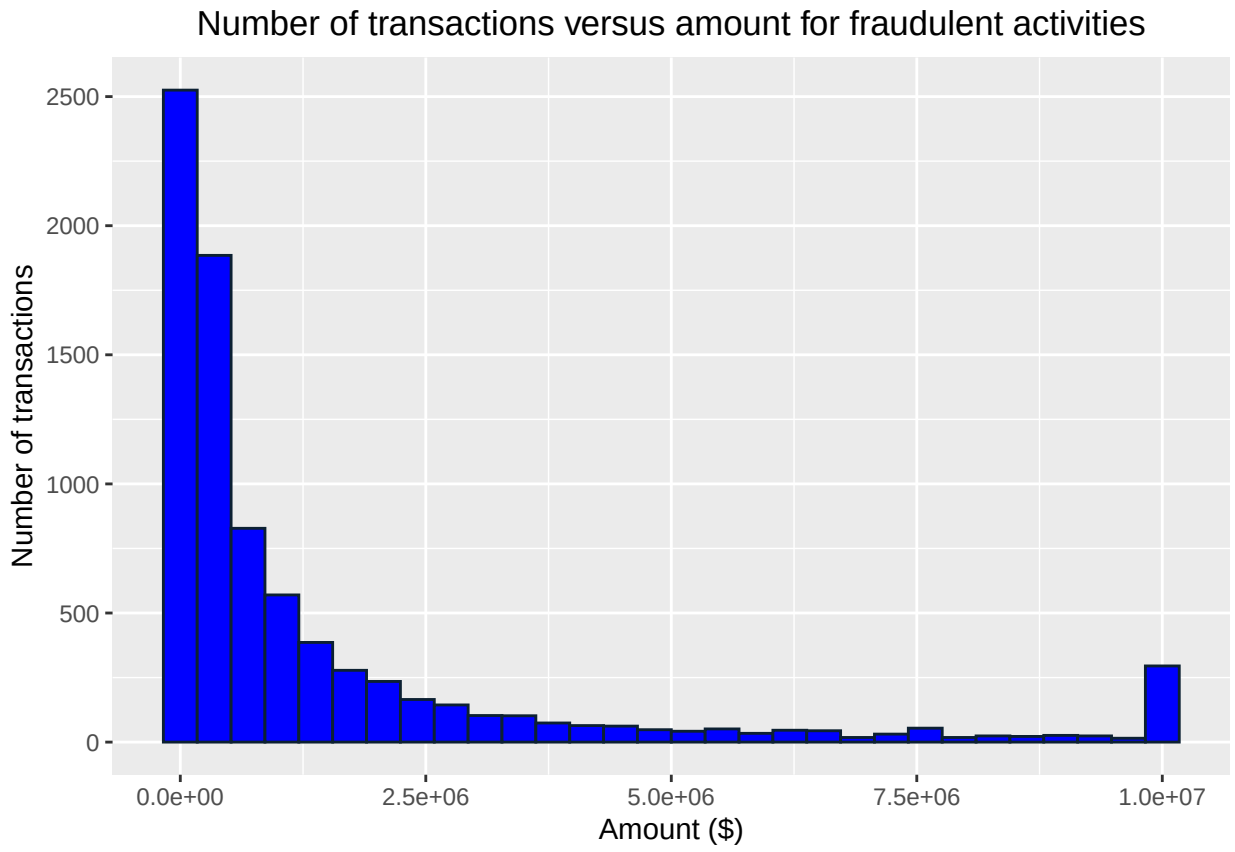
```
    1) %>% count()
```

Transaction Amount Effect

The following plot shows the number of transaction per amount of money taken out of the original account for fraudulent transactions. This plot shows that most fraudulent transactions involve low amounts. However, there is a group of fraudulent transactions that involve very large amounts.

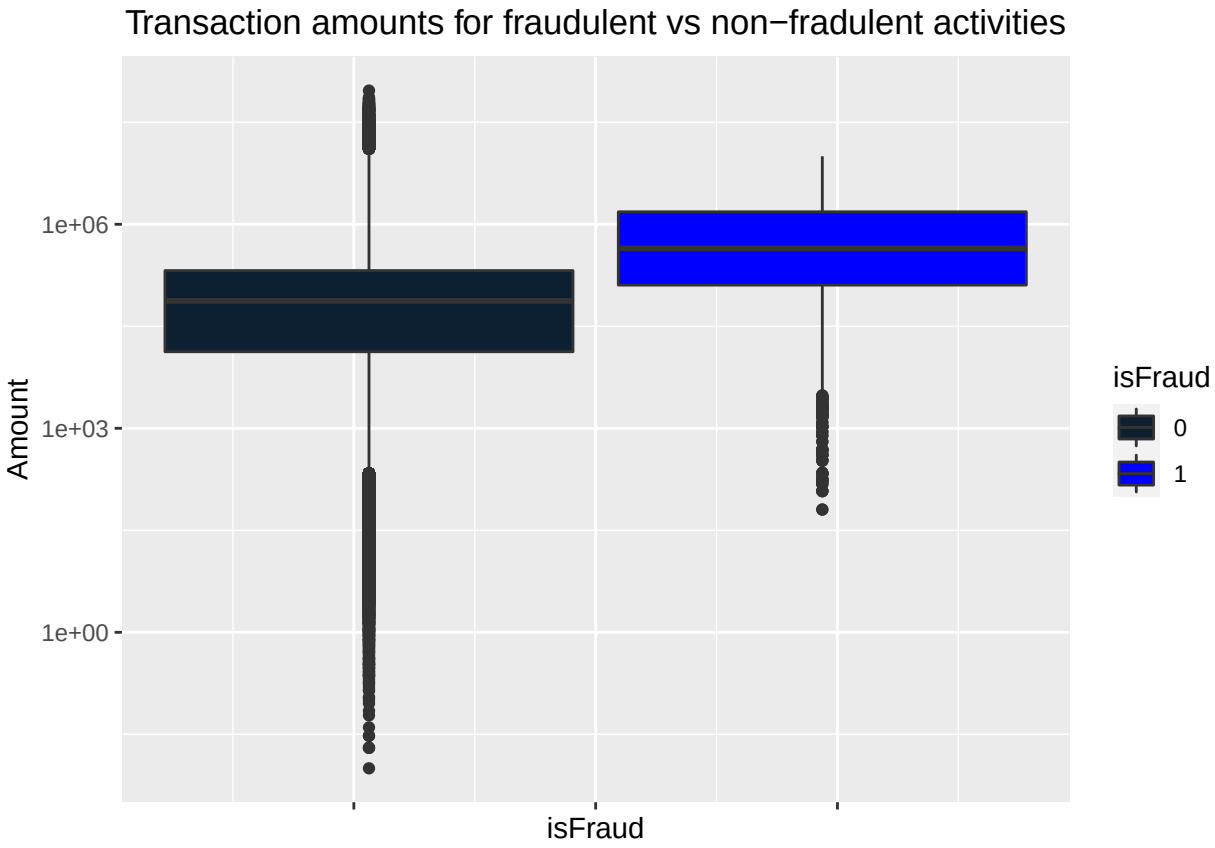
```
# Number of fraudulent transactions vs amount
fraud_data %>% ggplot(aes(x = amount)) + geom_histogram(fill = "blue",
    col = "#0c2032") + xlab("Amount ($)") + ylab("Number of transactions") +
    ggtitle("Number of transactions versus amount for fraudulent activities") +
    theme(plot.title = element_text(hjust = 0.5))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The following boxplot shows that fradulent transactions have higher median compared to non-fradulent transactions.

```
# transaction amounts for fraudulent vs
# non-fradulent transactions
rawdata %>% ggplot(aes(y = amount, fill = factor(isFraud))) +
    xlab("isFraud") + ylab("Amount") + geom_boxplot() +
    scale_y_log10() + scale_fill_manual(values = c("#0c2032",
    "blue")) + theme(axis.text.x = element_blank()) +
    labs(fill = "isFraud") + ggtitle("Transaction amounts for fraudulent vs non-fradulent a
    theme(plot.title = element_text(hjust = 0.5))
```



Destination Account

The following piece of code aims to understand if fraudulent activities are associated with any particular account name. That is, if a particular recipient is connected to more than one fraudulent activity. The code shows that only 44 recipient accounts associated with more than one fraudulent activity, and none are associated with more than two fraudulent activities. That is we cannot find theives by their account numbers.

```
# Number of fraudulent transactions in destination
# accounts
fraud_data %>% group_by(nameDest) %>% summarize(n = n()) %>%
    filter(n > 1) %>% nrow()
```

```
## [1] 44
```

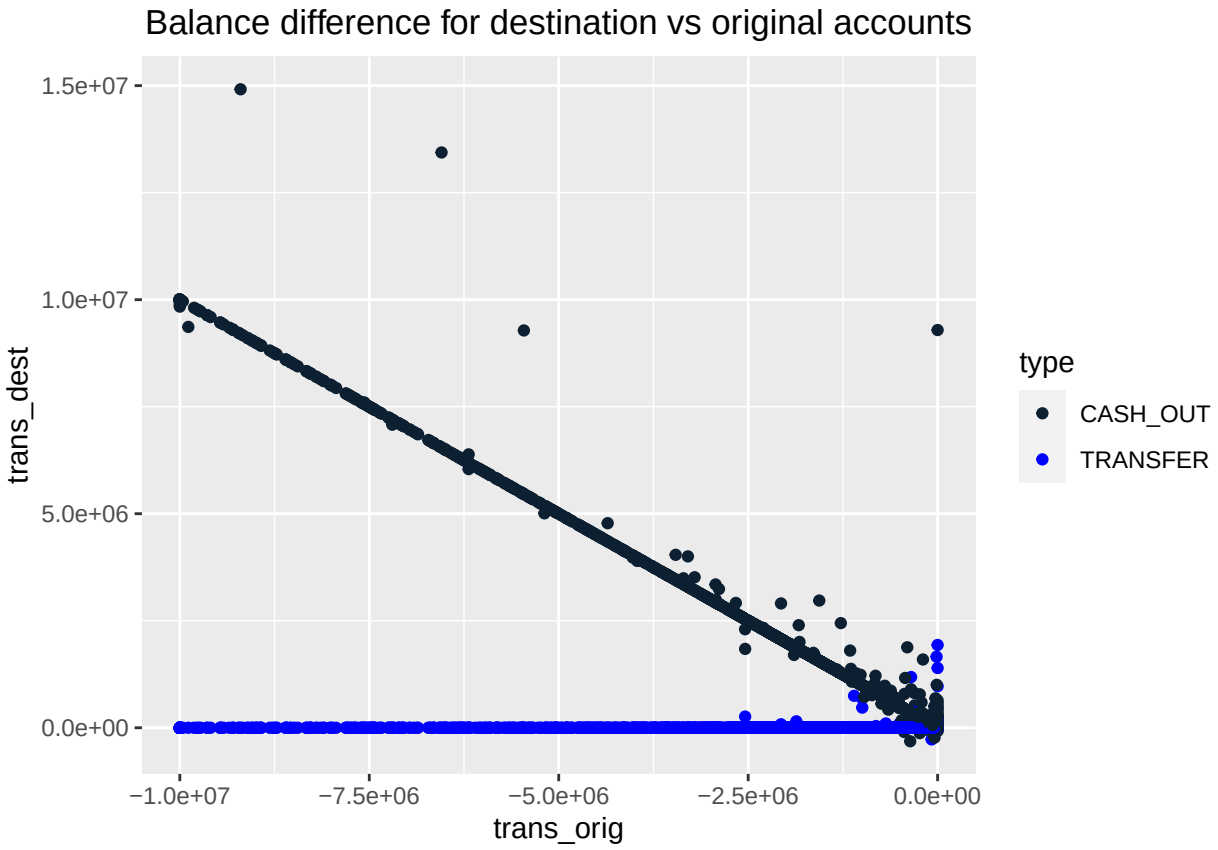
```
fraud_data %>% group_by(nameDest) %>% summarize(n = n()) %>%
    filter(n > 2) %>% nrow()
```

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

Balance Difference

The following plot shows the destination balance difference versus original balance difference.

```
fraud_data %>% ggplot(aes(x = trans_orig, y = trans_dest,
    col = type)) + geom_point() + scale_color_manual(values = c("#0c2032",
    "blue")) + ggtitle("Balance difference for destination vs original accounts") +
    theme(plot.title = element_text(hjust = 0.5))
```



This plot shows four groups of fraudulent activities:

Group A) The fraudulent activities in this group are associated with 8156 negative balance difference in the original account and no transaction having original balance gain. This makes sense, as the money has taken out of the original account.

```
# The number fraudulant activity for which
# trans_orig is negative.
fraud_data %>% filter(trans_orig < 0) %>% count()
```

```
# There is no fraudulant activity for which
# trans_orig is positive.
fraud_data %>% filter(trans_orig > 0) %>% count()
```

Group B) For 57 fraudulent transactions, original balance has not changed. These are the transactions falling on the vertical line at trans_orig=0.

```
# Fraudulant activities for which trans_orig
# remains unchanged.
fraud_data %>% filter(trans_orig == 0) %>% count()
```

Among this group, there are 16 cases for which transaction amount is $0 and no money has also been added to the recipient. It's not clear why these tranasctions are marked as fraud, since no money has been transfered or cashed out. For this project, however, these transactions are included in predictive algorithms.

```
fraud_data %>% filter(trans_orig == 0 & amount == 0) %>%
    select(amount, trans_orig, trans_dest, isFraud) %>%
    count()
```

Group C) In 4070 transactions with mostly associated with type “transfer”, although money has taken out of the original account, it has not been deposited into destination account. These transactions fall on the horizental line with trans_dest=0.

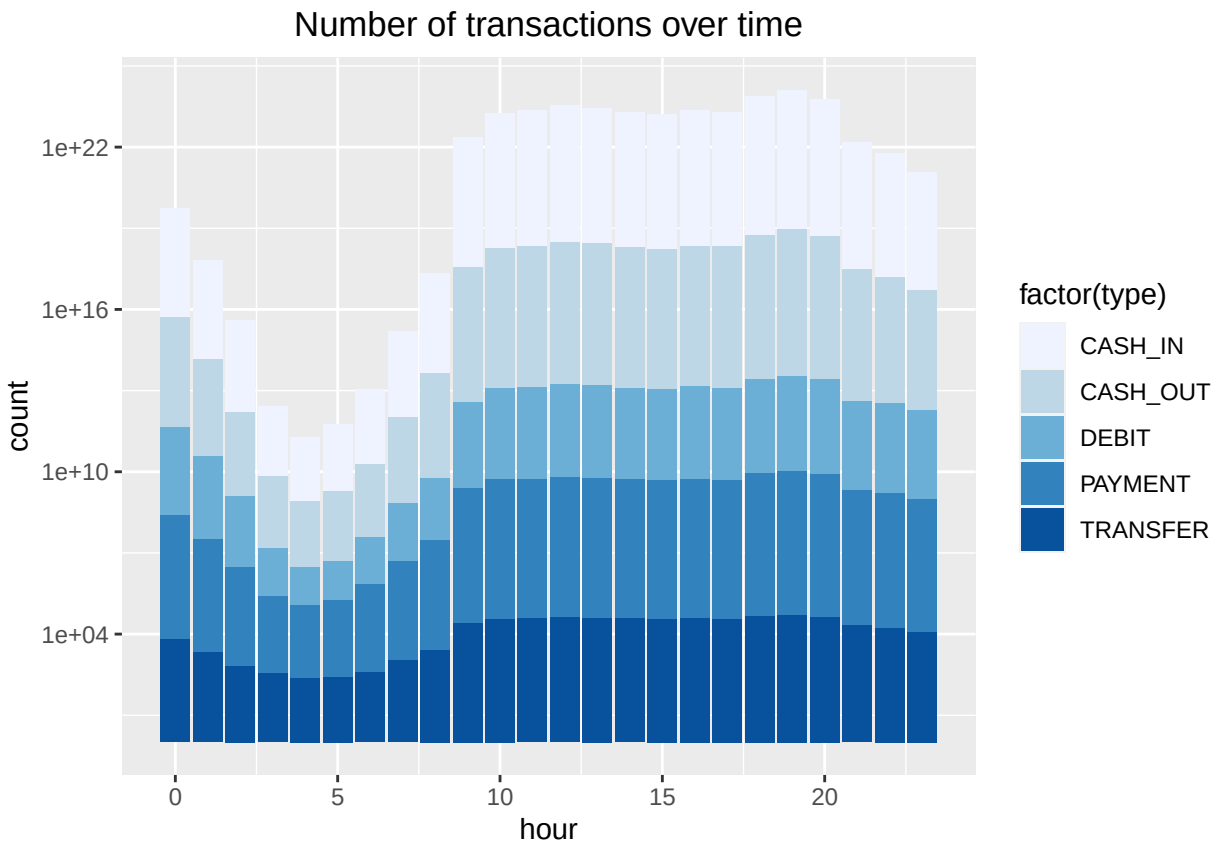
```
fraud_data %>% filter(amount != 0) %>% select(type,
    amount, trans_orig, trans_dest, isFraud) %>% filter(trans_dest ==
    0) %>% count()
```

Group D) In most transactions with type “cash_out” money has been deposited in the destination account. In most such cases, the balance gained at destination is equal to balance loss at origin.

Time Effect

The following plot shows the number of transactions over time of day for each transaction type. This plot shows that although fewer transactions occur overnight, the proportion of transaction types are fairly constant in each hour.

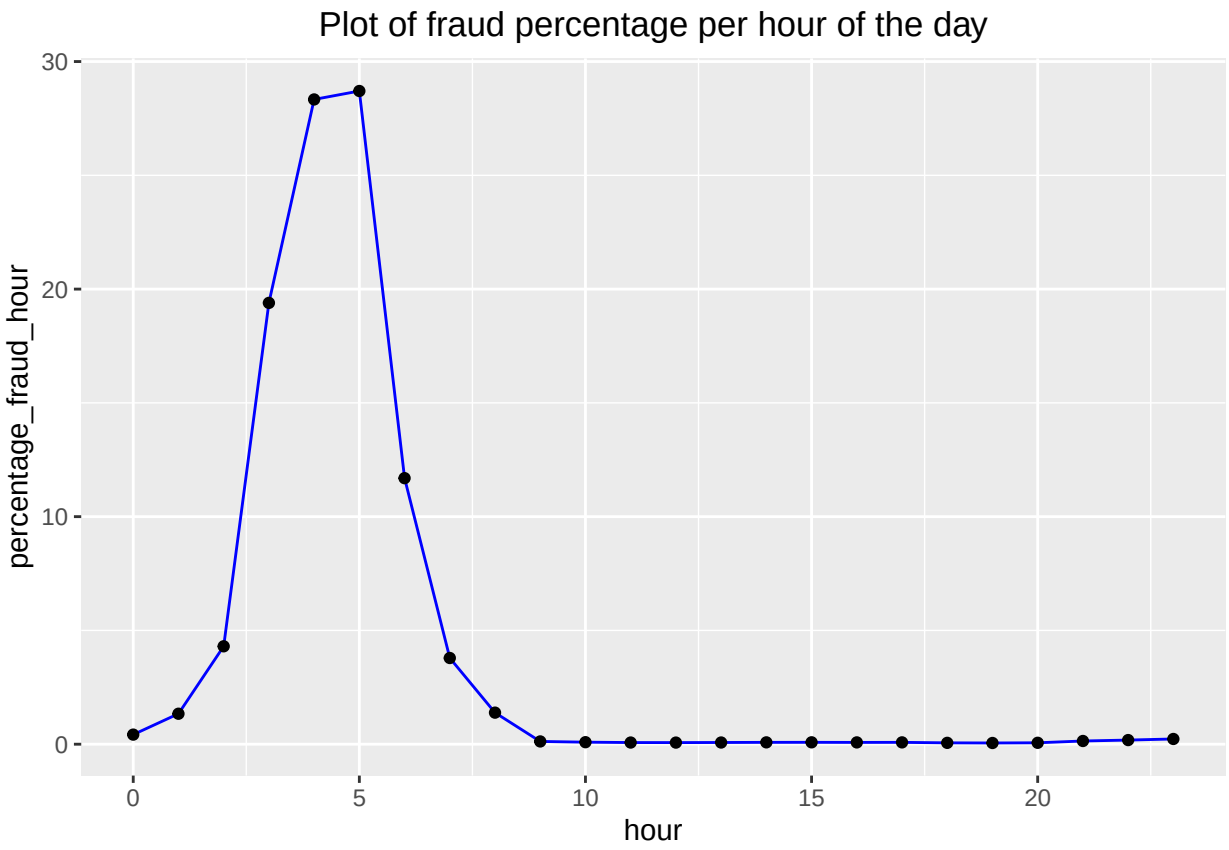
```
# Plot of transaction type per hour
rawdata %>% ggplot(aes(x = hour, fill = factor(type))) +
    geom_bar(stat = "count") + scale_y_log10() + scale_fill_brewer(palette = c("Blues")) +
    ggtitle("Number of transactions over time") + theme(plot.title = element_text(hjust = 0
```



The following code calculates the percentage of fraud in each hour and then plots the fraud percentage over time of the day. It’s evident that the ratio of fradulent activities are much higher overnight compared to daytime.

```
# Fraud percentage per hour of the day.
rawdata %>% group_by(hour) %>% summarize(n_fraud_hour = sum(isFraud ==
    1), n_nonfraud_hour = sum(isFraud == 0), percentage_fraud_hour = n_fraud_hour/n_nonfrau
    100) %>% ggplot(aes(hour, percentage_fraud_hour)) +
```

```
    geom_line(col = "blue") + geom_point() + ggtitle("Plot of fraud percentage per hour of
    theme(plot.title = element_text(hjust = 0.5))
```



Data Cleaning

As shown in above plots and tables, variables setp, isFlaggedFraud, nameOrig, and nameDest have no bearning in identifying fradulent activities. Also, oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest are correlated with each other, so only summarized columns "trans_orig" and "trans_dest" are included in modelling. In addition, transaction types CASH_IN, PAYMENT, and DEBIT are not associated with any fradulent transactions, therefore removed from prediction dataset. The CASH_OUT and TRANSFER types need to be encoded to be able to use in machine learning algorithm. Dummy variables are created to create two columns each representing a transaction type.

```
# Data cleaning
dat <- rawdata %>% select(type, amount, oldbalanceOrg,
    newbalanceOrig, oldbalanceDest, newbalanceDest,
    trans_orig, trans_dest, hour, isFraud) %>% filter(type %in%
    c("CASH_OUT", "TRANSFER"))

# converting dependent variable to factor
dat$isFraud <- as.factor(dat$isFraud)


# Creating dummy variables
dat <- dummy_columns(dat, select_columns = "type")
```

```
dat <- dat %>% select(type_CASH_OUT, type_TRANSFER,
    amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest,
    newbalanceDest, trans_orig, trans_dest, hour, isFraud)
```

Creating a Subset of Data for Machine Learning

The dataset as above 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 the sake of this project, only a subset of the original dataset is used. Note that for more powerfull computers the following code can be removed.

```
# Creating a random subset of original data
set.seed(1)
dat <- sample_frac(dat, size = 0.3, replace = FALSE)
```

MODELLING

For the purpose of this project the data is divided into training and test sets, as follows:

```
set.seed(1)
index <- createDataPartition(dat$isFraud, times = 1,
    p = 0.2, list = FALSE)
training_set <- dat %>% slice(-index)
validation_set <- dat %>% slice(index)
```

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

```
# Splitting training_set
set.seed(1)
ind <- createDataPartition(training_set$isFraud, times = 1,
    p = 0.2, list = FALSE)
training_set_split <- training_set %>% slice(-ind)
test_set <- training_set %>% slice(ind)
```

The number of training_set, validation_set, training_set_split, and test_set rows are as folows:

```
nrow(training_set_split)
```

```
## [1] 531918
```

```
nrow(training_set)
```

```
## [1] 664898
```

```
nrow(test_set)
```

```
## [1] 132980
```

```
nrow(validation_set)
```

```
## [1] 166225
```

Base Model- Guessing

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

```
set.seed(1)
y_hat_guessing <- sample(c("1", "0"), length(ind),
    replace = TRUE) %>% factor()
```

The overall accuracy is the overall proportion that is predicted correctly. This will be extracted from the confusion matrix as follows:

```
set.seed(1)
y_hat_guessing <- sample(c("1", "0"), length(ind),
    replace = TRUE) %>% factor()

# The overall accuracy is the overall proportion
# that is predicted correctly.
cm_1 <- confusionMatrix(data = y_hat_guessing, test_set$isFraud)

# Calculating overall accuracy
Accuracy_1 <- cm_1$overall["Accuracy"]
Accuracy_1
```

```
##  Accuracy
## 0.5001579
```

```
# Calculating Sensitivity and Specificity
Guessing <- cm_1$byClass[c("Sensitivity", "Specificity")]

kable(Guessing, "latex", caption = "Model Specifications") %>%
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 4: Model Specifications

	x
Sensitivity	0.5001735
Specificity	0.4951691

Not surprisingly, the accuracy is not very good. The sensitivity and specificity are also not good.

Time Effect Model

As noted above, it is established that most fraudulent activities occur overnight. So, we migh be able to predict better, if we incorporate time into our predictions as follows:

```
y_hat_time <- ifelse(test_set$hour >= 1 & test_set$hour <=
    7, "1", "0") %>% factor()

# Confusion Matrix
cm_2 <- confusionMatrix(data = y_hat_time, test_set$isFraud)
```

```
# Calculating overall accuracy
cm_2$overall["Accuracy"]
```

```
##  Accuracy
## 0.9942548
```

The accuracy has improved a lot using this simple model. 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. But is the opposite true? If this algorithm predicts fraud, is it actually a fraud? For this, we need to also look at specifity.

```
# Calculating Sensitivity and Specificity
TimeEffect <- cm_2$byClass[c("Sensitivity", "Specificity")]

# Comapring Models
model_specifications <- rbind(Guessing, TimeEffect)

kable(model_specifications, "latex", caption = "Model Specifications") %>%
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 5: Model Specifications

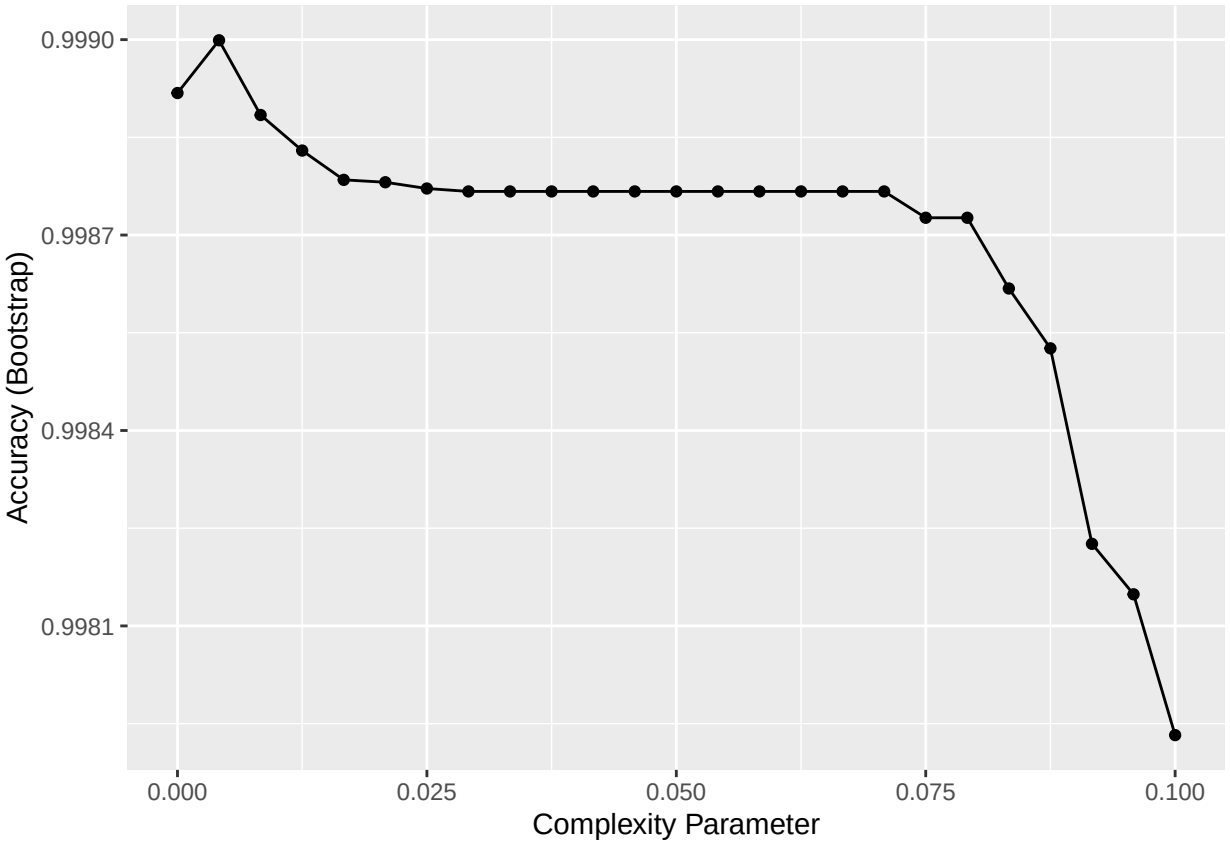
	Sensitivity	Specificity
Guessing	0.5001735	0.4951691
TimeEffect	0.9963188	0.3333333

The confusion matrix above shows that this algorithm has very low specifity. As there are a lot more non-fraudulent transactions compared to fradulent transactions, this model mistakenly predicts non-fradulent transactions as fraud. In this project, it is better to favor sensitivity over specificity because it is important to identify fraudulent transactions correctly in expense of having some legal transactions being blocked. However, we'll see if there is a predictive algorithm to improve specificity as well.

Rpart Classification Model

In rpart algorithm, the dataset is by split recursively. This means that the subsets that resulted from a split are further 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.

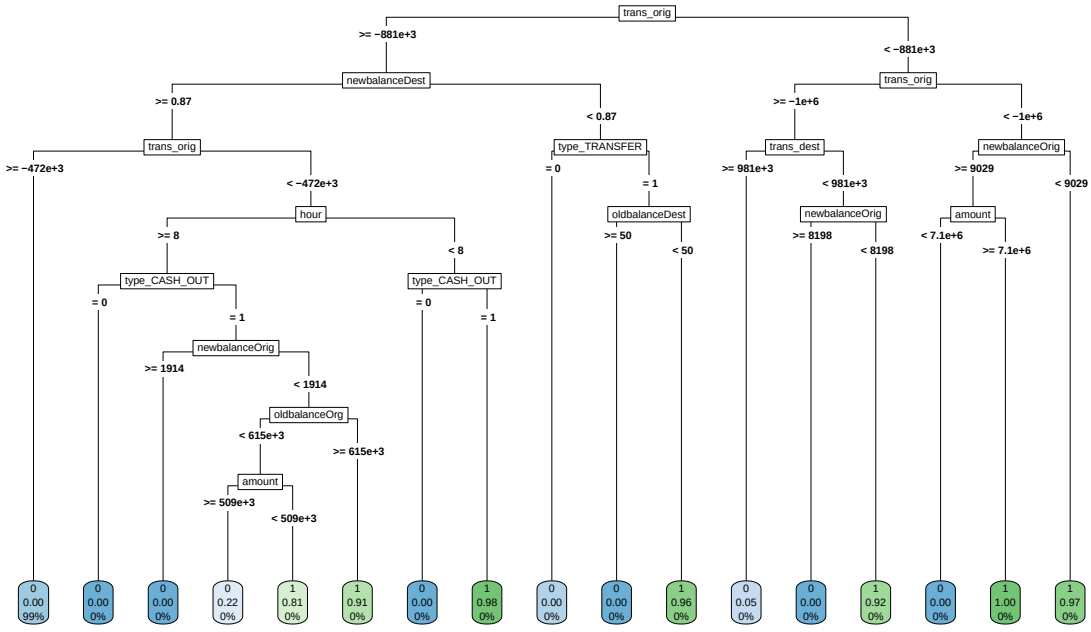
```
fit_rpart <- train(isFraud ~ ., method = "rpart", tuneGrid = data.frame(cp = seq(0,
    0.1, len = 25)), data = training_set_split)
ggplot(fit_rpart) + theme(plot.title = element_text(hjust = 0.5))
```



The following plot shows the rpart decision tree.

```
# Plotting the decision tree
rpart.plot(fit_rpart$finalModel, type = 5, main = "Decision Tree")
```

Decision Tree



```
# Prediction
y_hat_rpart <- predict(fit_rpart, test_set)

# Confusion Matrix
cm_3 <- confusionMatrix(data = y_hat_rpart, test_set$isFraud)

# Calculating model accuracy
cm_3$overall["Accuracy"]
```

```
##  Accuracy
## 0.9989923
```

```
# Calculating Sensitivity and Specificity
Rpart <- cm_3$byClass[c("Sensitivity", "Specificity")]

model_specifications <- rbind(Guessing, TimeEffect,
    Rpart)
kable(model_specifications, "latex", caption = "Model Specifications") %>%
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 6: Model Specifications

	Sensitivity	Specificity
Guessing	0.5001735	0.4951691
TimeEffect	0.9963188	0.3333333
Rpart	0.9998567	0.7222222

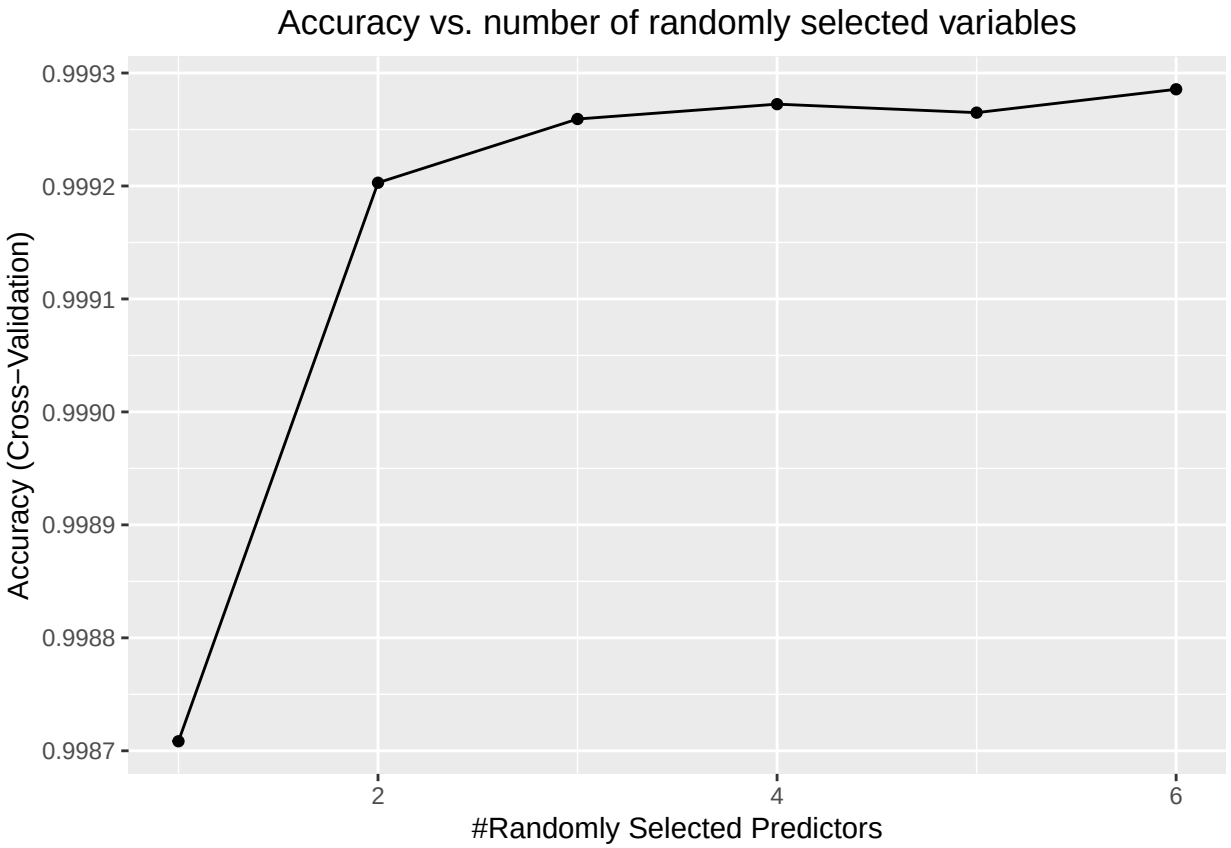
Our model has significantly been improved. This model is based on only one tree. Usually, a combination of a number of trees will help improve our predictions. Therefore, the Random Forest method is also motivated:

Random Forest Model

Rather than using only one decision tree as in rpart model, the random forest model uses multiple decision trees. The term "random" is used because training dataset is randomly sampled and the predictors are randomly selected. Accuracy, sensitivity, and specificity has improved even more compared to rpart decision tree. Therefore, this algorithm is selected for prediction. The model is trained for "mtry" tuning parameter.

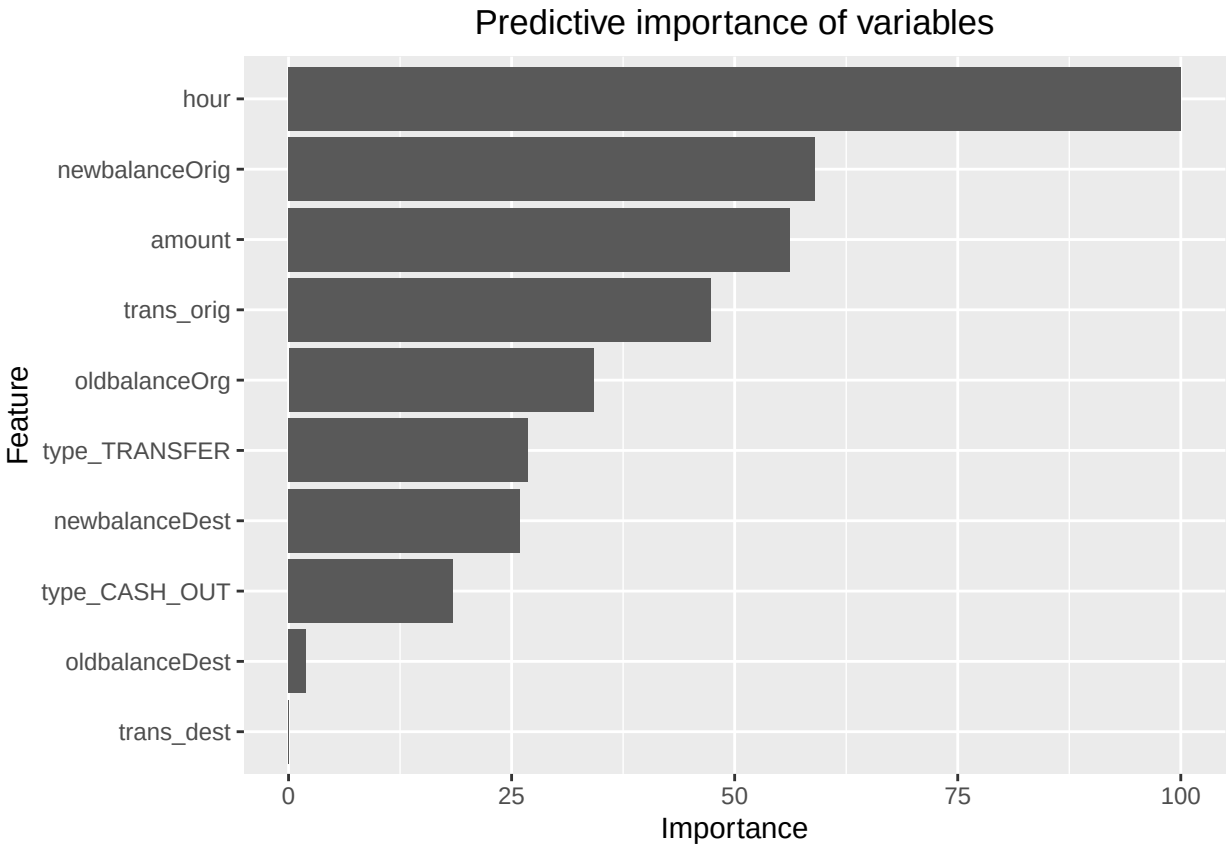
```
set.seed(1)
fit_rf <- train(isFraud ~ ., data = training_set_split,
    trControl = trainControl(method = "cv", number = 5),
    importance = TRUE, method = "rf", ntree = 25, tuneGrid = data.frame(mtry = seq(1,
        6, 1)))

fit_rf$bestTune  #gives best mtry
```

```
# Tuning parameters
ggplot(fit_rf) + ggtitle("Accuracy vs. number of randomly selected variables") +
    theme(plot.title = element_text(hjust = 0.5))
```



The most important parameters for random forest predictions are plotted. And model parameters are calculated.

```
ggplot(varImp(fit_rf)) + ggtitle("Predictive importance of variables") +
    theme(plot.title = element_text(hjust = 0.5))
```




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

# Confusion Matrix
cm_4 <- confusionMatrix(data = y_hat_rf, test_set$isFraud)
cm_4$overall["Accuracy"]
```

```
##  Accuracy
## 0.9994134
```

```
RandomForest <- cm_4$byClass[c("Sensitivity", "Specificity")]

model_specifications <- rbind(Guessing, TimeEffect,
    Rpart, RandomForest)

kable(model_specifications, "latex", caption = "Model Specifications") %>%
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 7: Model Specifications

	Sensitivity	Specificity
Guessing	0.5001735	0.4951691
TimeEffect	0.9963188	0.3333333
Rpart	0.9998567	0.7222222
RandomForest	0.9999246	0.8357488

Testing the final model on validation set

After choosing the desired algorithm, the final model (random 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)

# Predictions
y_hat_final <- predict(fit_rf, validation_set)

# Confusion Matrix
cm_final <- confusionMatrix(data = y_hat_final, validation_set$isFraud)

cm_final$overall["Accuracy"]
```

```
##  Accuracy
## 0.9993082
```

```
RandomForest_final <- cm_final$byClass[c("Sensitivity",
    "Specificity")]

model_specifications <- rbind(Guessing, TimeEffect,
    Rpart, RandomForest, RandomForest_final)

kable(model_specifications, "latex", caption = "Model Specifications") %>%
    kable_styling(font_size = 10) %>% kable_styling(latex_options = "HOLD_position")
```

Table 8: Model Specifications

	Sensitivity	Specificity
Guessing	0.5001735	0.4951691
TimeEffect	0.9963188	0.3333333
Rpart	0.9998567	0.7222222
RandomForest	0.9999246	0.8357488
RandomForest_final	0.9999276	0.8007737

CONCLUSION

The Random Forest model is proved to be the best algorithm that can accurately predict fradulent and non-fradulent 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 71%, 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 staff for providing this absolutely valuable resource. I would also thank all the fellow students who take the time to read and grade this project.