Machine Learning Model For Financial Fraud Detection

For this project, I will be applying machine learning techniques to predict the outcome in a given dataset. I will be using a dataset from kaggle that is composed of 6 million of money transactions. Here the goal is to oredict if a given transaction is a Fraud or not.

Dataset Overview

Context

There is a lack of public available datasets on financial services and specially in the emerging mobile money transactions domain. Financial datasets are important to many researchers and in particular to us performing research in the domain of fraud detection. Part of the problem is the intrinsically private nature of financial transactions, that leads to no publicly available datasets.

This dataset was generated using the simulator called PaySim as an approach to such a problem. PaySim uses aggregated data from the private dataset to generate a synthetic dataset that resembles the normal operation of transactions and injects malicious behaviour to later evaluate the performance of fraud detection methods.

Content

PaySim simulates mobile money transactions based on a sample of real transactions extracted from one month of financial logs from a mobile money service implemented in an African country. The original logs were provided by a multinational company, who is the provider of the mobile financial service which is currently running in more than 14 countries all around the world.

Headers

This is a sample of 1 row with headers explanation:

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	PAYMENT	9839.64	C1231006815	170136	160296.4	M1979787155	0	0	0	0

Column	Description
step	maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation)
type	CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
amount	amount of the transaction in local currency.

Column	Description
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.

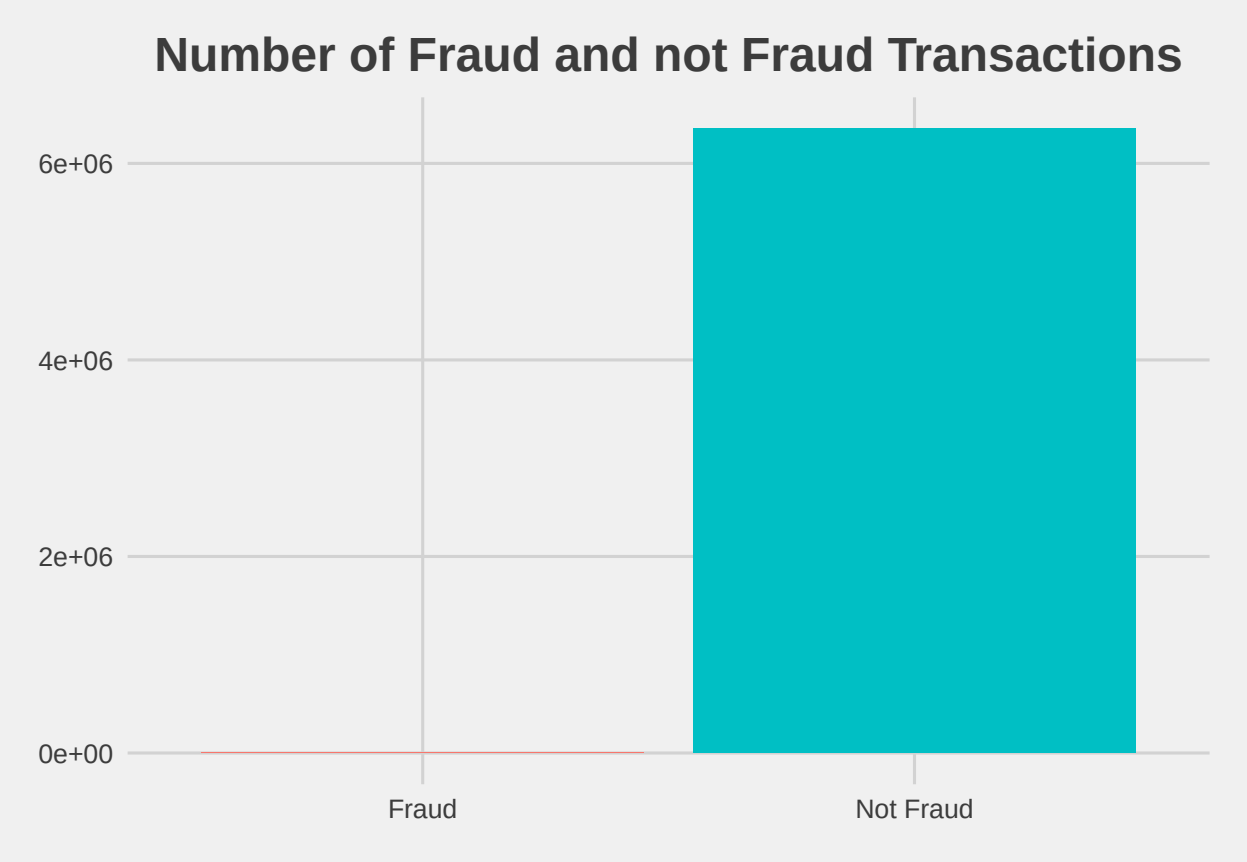
```
head(data)
```

```
## # A tibble: 6 x 11
##    step type  amount nameOrig oldbalanceOrg newbalanceOrig nameDest
##   <dbl> <chr>  <dbl> <chr>            <dbl>          <dbl> <chr>
## 1     1 PAYM~  9840. C123100~        170136        160296. M197978~
## 2     1 PAYM~  1864. C166654~         21249         19385. M204428~
## 3     1 TRAN~   181  C130548~           181             0  C553264~
## 4     1 CASH~   181  C840083~           181             0  C389970~
## 5     1 PAYM~ 11668. C204853~         41554         29886. M123070~
## 6     1 PAYM~  7818. C900456~         53860         46042. M573487~
## # ... with 4 more variables: oldbalanceDest <dbl>, newbalanceDest <dbl>,
## #   isFraud <dbl>, isFlaggedFraud <dbl>
```

Data Analysis

Let´s look at our data distribuition grouped by Fraud and not Fraud Transactions.

```
data %>%
  mutate(isFraud = ifelse(isFraud == 1, "Fraud", "Not Fraud")) %>%
  ggplot(aes(isFraud, fill = isFraud)) +
  geom_bar() +
  ggtitle("Number of Fraud and not Fraud Transactions") +
  theme_fivethirtyeight() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))
```




```
data %>%
  summarize(Fraud = sum(isFraud == 1), Not_Fraud = sum(isFraud == 0))
```

```
## # A tibble: 1 x 2
##   Fraud Not_Fraud
##   <int>     <int>
## 1  8213   6354407
```

As we can see there are a lot of transactions that weren´t fraud and just a few that are fraud. This can cause or model to always predict for not fraud and always get a high accuracy. To solve this we will split the data into a set of 50% fraud and 50% not fraud.

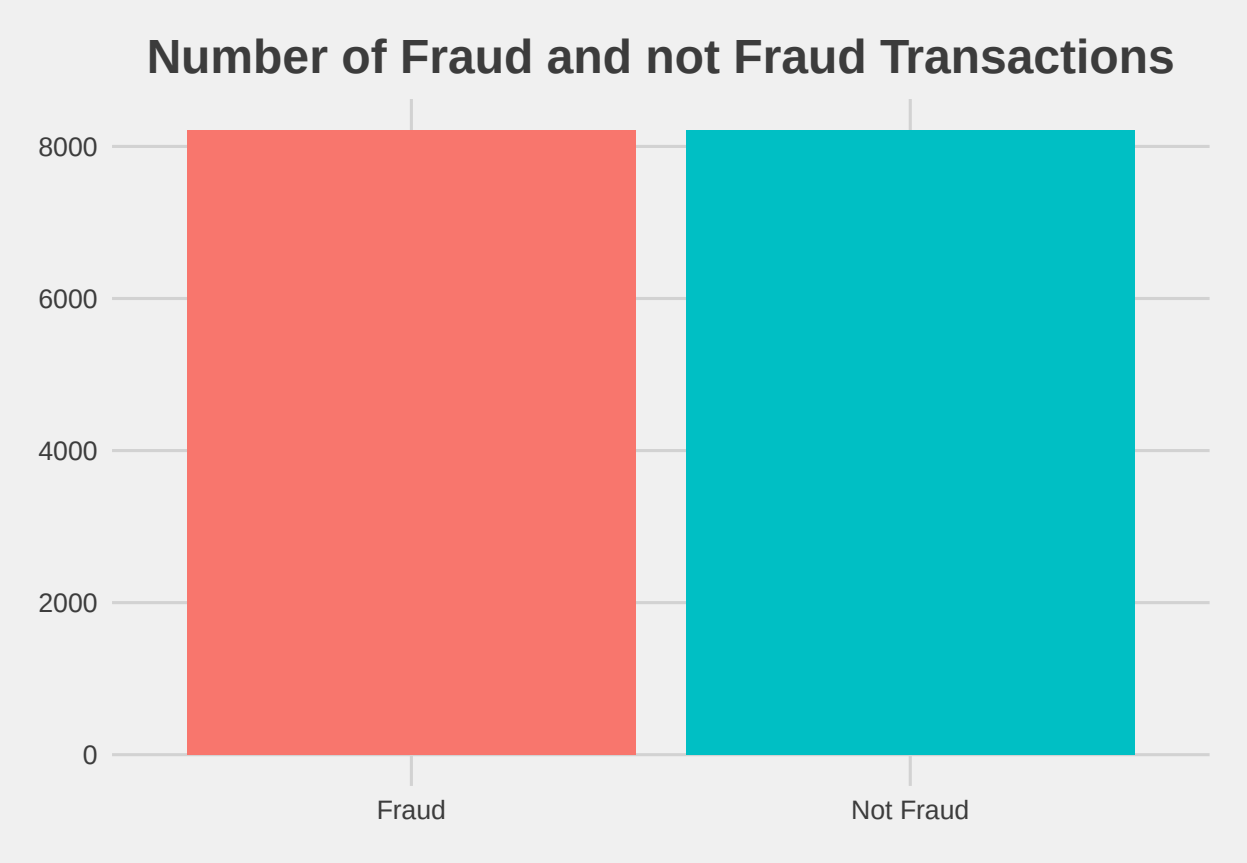
```
set.seed(0)
size <- dim(data[data$isFraud == 1,])[1]
temp_df_fraud <- data[data$isFraud == 1,]
temp_df_not_fraud <- data[data$isFraud == 0,][sample(seq(1, size), size),]

df <- full_join(temp_df_not_fraud, temp_df_fraud)
```

```
## Joining, by = c("step", "type", "amount", "nameOrig", "oldbalanceOrg", "newbalanceOrig", "nameDest",
```

```
rm(temp_df_fraud, temp_df_not_fraud)
```

As we can see now we have a dataser with equal proportion of fraud and not fraud transactions.

```
df %>%
  mutate(isFraud = ifelse(isFraud == 1, "Fraud", "Not Fraud")) %>%
  ggplot(aes(isFraud, fill = isFraud)) +
  geom_bar() +
  ggtitle("Number of Fraud and not Fraud Transactions") +
  theme_fivethirtyeight() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))
```



Now we can start constructing the prediction model.

Methods and Models

Our goal is to predict wether a certain transaction is a fraud or not, given tha value of the predictors. First let´s drop all the non-numeric columns, as they don´t apport too much to the data.

```
df_num <- df %>%
  select(-c(nameOrig, nameDest, type))
head(df_num)
```

```
## # A tibble: 6 x 8
```

```
##    step amount oldbalanceOrg newbalanceOrig oldbalanceDest newbalanceDest
##   <dbl>  <dbl>         <dbl>          <dbl>          <dbl>          <dbl>
## 1     1 10060.        10776           716.              0              0
## 2     7  7125.          590              0              0              0
## 3     4  3118.         6344           3226.             0              0
## 4     3 23441.            0              0              0              0
## 5     1 46179.      7292261.       7338440.        306791.              0
## 6     1 24495.      4593261.       4568765.             0               0
## # ... with 2 more variables: isFraud <dbl>, isFlaggedFraud <dbl>
```

Now let´s create the training and testing datasets (which are mutal exclusive). This will make our model don´t be overfitted to the trainig data and will make our predictions more accurate to the ones of the real values for any given dataset.

```
set.seed(0)
test_index <- createDataPartition(df_num$isFraud, times = 1, p = 0.2, list = FALSE) # first create the

# select our test set
test_x <- select(df_num, -isFraud)[test_index,]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
test_y <- df_num$isFraud[test_index]

# select the reamining indexes for our train set
train_x <- select(df_num, -isFraud)[-test_index,]
train_y <- df_num$isFraud[-test_index]

# change de training data as factor because we will only have to values
train_y <- as.factor(train_y)
```

Now, with our train and test data set. We can train a machine learning model that predicts wether a transactions is a Fraud or not given the numeric predictors.

Model 1: K Means

Is a grouping method, which aims at dividing a set of n observations into k groups in which each observation belongs to the group whose mean value is closest.

```
predict_kmeans <- function(x, k) {
    centers <- k$centers    # extract cluster centers
    # calculate distance to cluster centers
    distances <- sapply(1:nrow(x), function(i){
                        apply(centers, 1, function(y) dist(rbind(x[i,], y)))
                })
  max.col(-t(distances))  # select cluster with min distance to center
```

```
}
set.seed(0)
k <- kmeans(train_x, centers = 2)
kmeans_preds <- ifelse(predict_kmeans(test_x, k) == 1, 0, 1)
# mean(kmeans_preds == test_y)
```

Results

```
results <- data_frame(method = "K means", accuracy = mean(kmeans_preds == test_y)) # save the results i
results %>% knitr::kable()
```

method	accuracy
K means	0.5

Model 2: GLM

GLM will keep the weighted sum of all the features, but allow non-Gaussian outcome distributions (not like a linear regression model) and connect the expected mean of this distribution and the weighted sum through a possibly nonlinear function.

```
set.seed(0)
# Train the model
train_glm <- train(train_x, train_y,
                   method = "glm")

# Predict for the test set
glm_preds <- predict(train_glm, test_x)
# mean(glm_preds == test_y)
```

Results

```
results <- bind_rows(results, # add accuracy to the df
                          data_frame(method="Logistic regression",
                                     accuracy = mean(glm_preds == test_y)))
results %>% knitr::kable()
```

method	accuracy
K means	0.5000000
Logistic regression	0.9942179

Model 3: LDA

Linear Discriminant Analysis (LDA) is a method used to find a linear combination of features that characterize or separate two or more kinds of objects or events. The resulting combination can be used as a linear classifier.

```
set.seed(0)
# Train the model
train_lda <- train(train_x, train_y,
                     method = "lda")

# Predict for the test set
lda_preds <- predict(train_lda, test_x)
# mean(lda_preds == test_y)
```

Results

```
results <- bind_rows(results, # add accuracy to the df
                           data_frame(method="LDA",
                                      accuracy = mean(lda_preds == test_y)))
results %>% knitr::kable()
```

method	accuracy
K means	0.5000000
Logistic regression	0.9942179
LDA	0.8898357

Model 4: KNN

It is a method that simply searches the closest observations to the one you are trying to predict and classifies the point of interest based on most of the data that surrounds it.

```
set.seed(0)
# Train the model
train_knn <- train(train_x, train_y,
                     method = "knn",
                     tuneGrid = data.frame(k = seq(1.95, 2, 0.01)))
# Predict for the test set
knn_preds <- predict(train_knn, test_x)
# mean(knn_preds == test_y)
# train_knn$bestTune %>% pull()
```

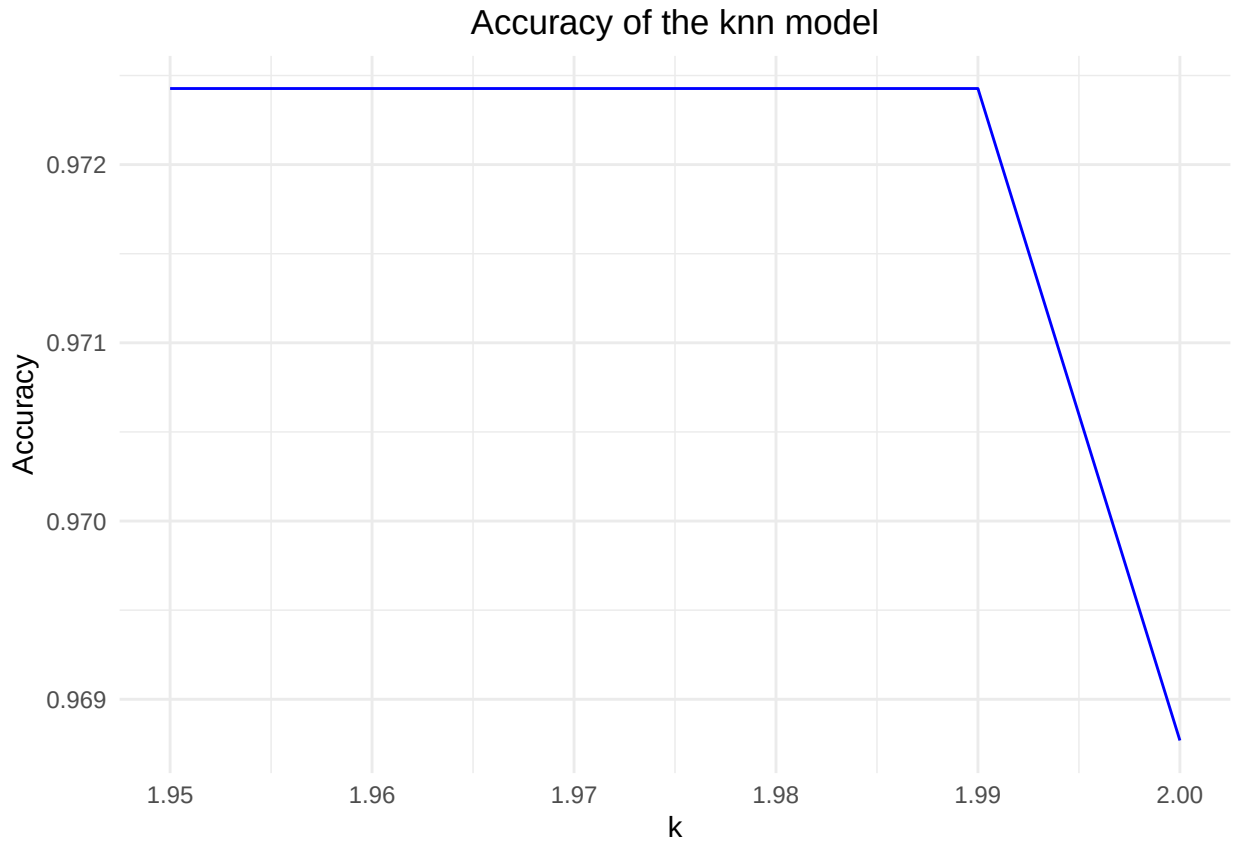
Results

```
results <- bind_rows(results, # add accuracy to the df
                           data_frame(method="KNN",
                                      accuracy = mean(knn_preds == test_y),
                                      tune = train_knn$bestTune %>% pull()))
results %>% knitr::kable()
```

method	accuracy	tune
K means	0.5000000	NA
Logistic regression	0.9942179	NA
LDA	0.8898357	NA
KNN	0.9811321	1.99

The best accuracy is given a k of 1.99, as it is ilustrated in the next plot.

```
train_knn$results %>%
  ggplot(aes(x = k, y = Accuracy)) +
  geom_line(color = "blue") +
  ggtitle("Accuracy of the knn model") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



Model 5: Random Forest

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

```r
set.seed(0)
# Train the model
train_rf <- train(train_x, train_y,
                  method = "rf",
                  tuneGrid = data.frame(mtry = seq(5.4,5.6,0.1)),
                  importance = TRUE)
# Predict for the test set
rf_preds <- predict(train_rf, test_x)
# mean(rf_preds == test_y)
# train_rf$bestTune %>% pull()
```

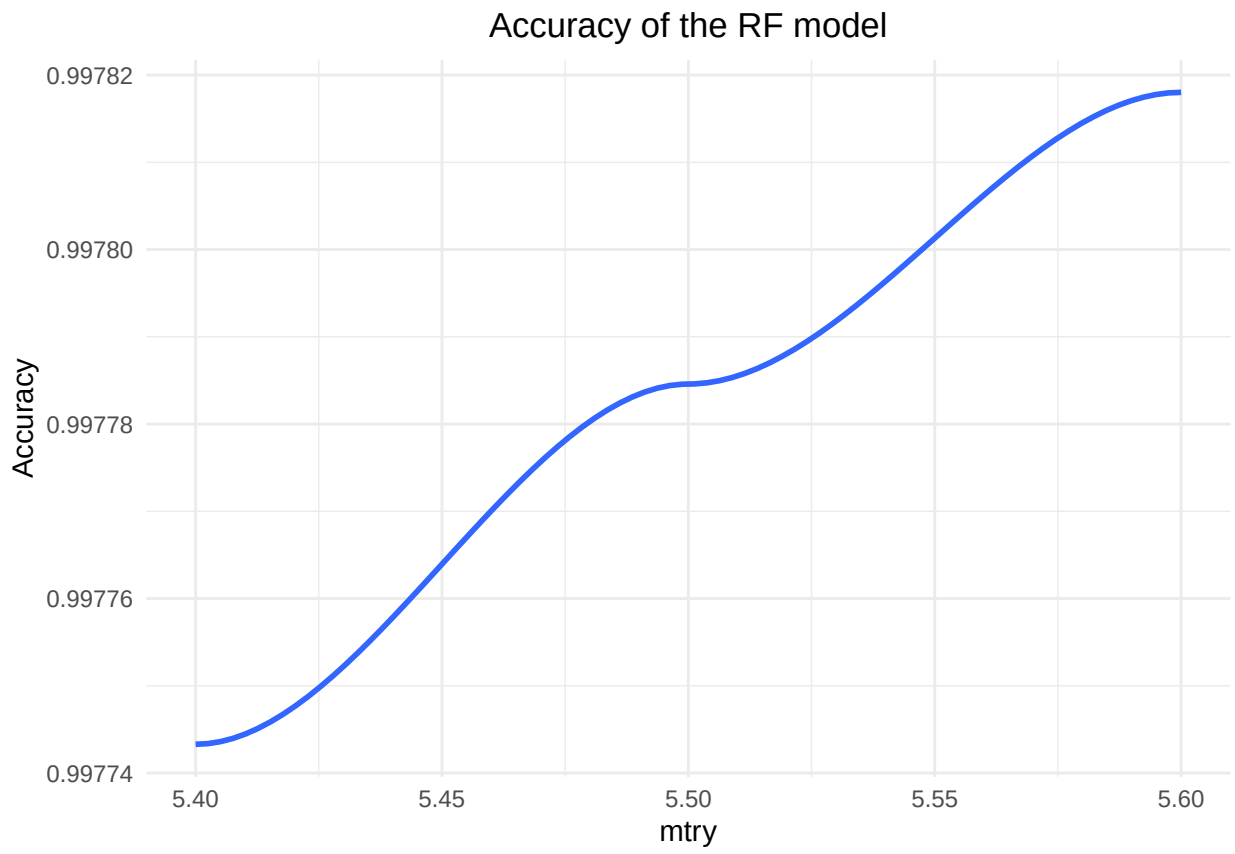
Results

```r
results <- bind_rows(results, # add accuracy to the df
                           data_frame(method="RF",
                                      accuracy = mean(rf_preds == test_y),
                                      tune = train_rf$bestTune %>% pull()))
results %>% knitr::kable()
```

method	accuracy	tune
K means	0.5000000	NA
Logistic regression	0.9942179	NA
LDA	0.8898357	NA
KNN	0.9811321	1.99
RF	0.9972611	5.60

The best accuracy is given a mtry value of 5.5, as it is ilustrated in the next plot.

```r
train_rf$results %>%
  ggplot(aes(x = mtry, y = Accuracy)) +
  geom_smooth() +
  ggtitle("Accuracy of the RF model") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



Model 6: Ensemble

We have created a few machine learning models with different accuracy values. But may be we can improve this further by combining all of this models into one.

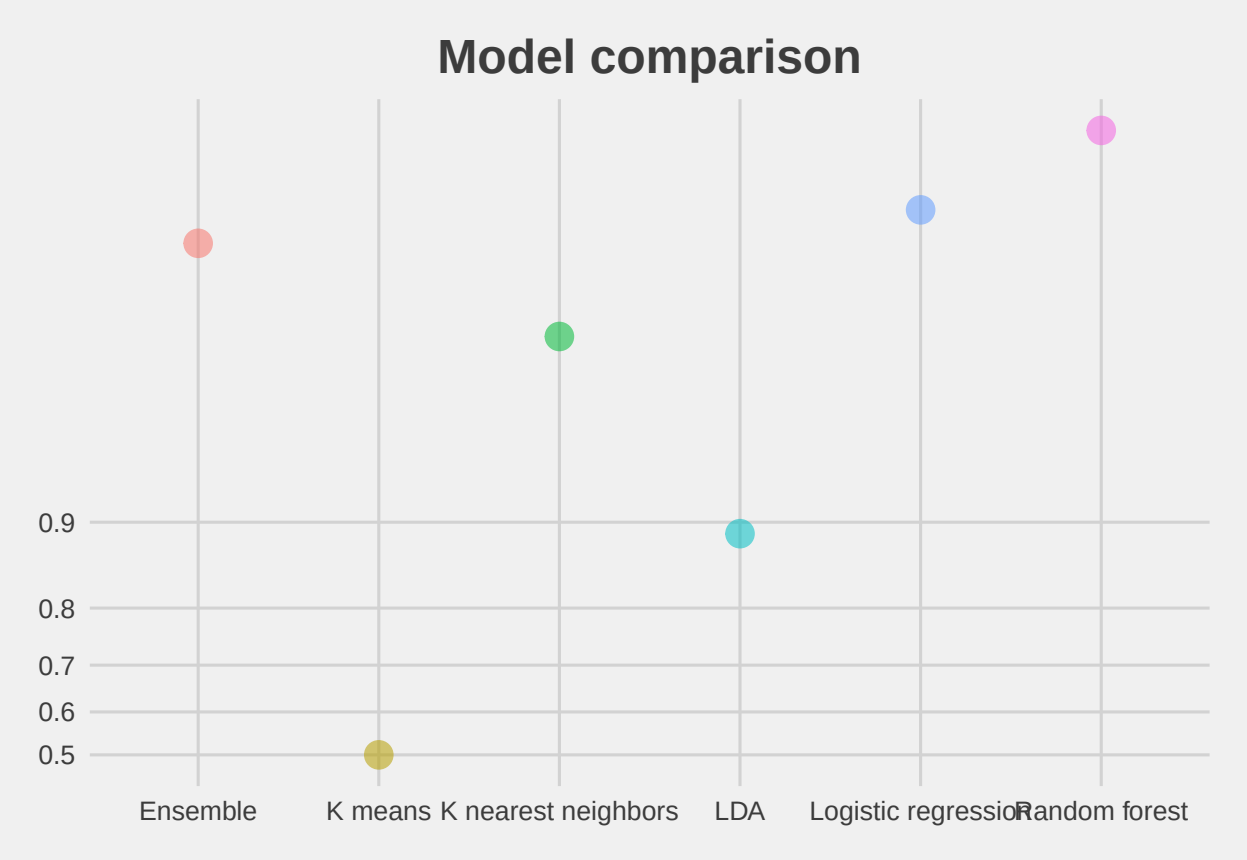
For this we will use all of the models prediction and use a mayority vote to decide wich vale is the correct one. By doing this we can expect a better accuracy than the average model accuracy.

```
# Generate a matrix for the models predictions
models <- matrix(c(kmeans_preds, glm_preds, lda_preds, knn_preds, rf_preds), ncol = 5)

# Make a prediction using a mayority vote
ensemble_preds <- ifelse(rowMedians(models) == 1, 0, 1)

# Generate the data frame
models <- c("K means", "Logistic regression", "LDA", "K nearest neighbors", "Random forest", "Ensemble")
accuracy <- c(mean(kmeans_preds == test_y),
              mean(glm_preds == test_y),
              mean(lda_preds == test_y),
              mean(knn_preds == test_y),
              mean(rf_preds == test_y),
              mean(ensemble_preds == test_y))
data.frame(Model = models, Accuracy = accuracy) %>% knitr::kable()
```

Model	Accuracy
K means	0.5000000
Logistic regression	0.9942179
LDA	0.8898357
K nearest neighbors	0.9811321
Random forest	0.9972611
Ensemble	0.9920876

The ensemble model got a very good accuracy, however it´s still behind the Logistic Regression and Random Forest models, as seen in the next plot.

```
data.frame(Model = models, Accuracy = accuracy) %>%
  ggplot(aes(x = Model, y = Accuracy, size = 8 , color = Model, alpha = 0.5)) +
  geom_point() +
  theme_fivethirtyeight() +
  ggtitle("Model comparison") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(trans = "logit")
```



We could improve it a little bit by just using the models with an accuracy above 90%.

```
#kmeans_preds_num <- ifelse(kmeans_preds == "B", 1, 2)
models <- matrix(c(glm_preds, knn_preds, rf_preds), ncol = 3)
```

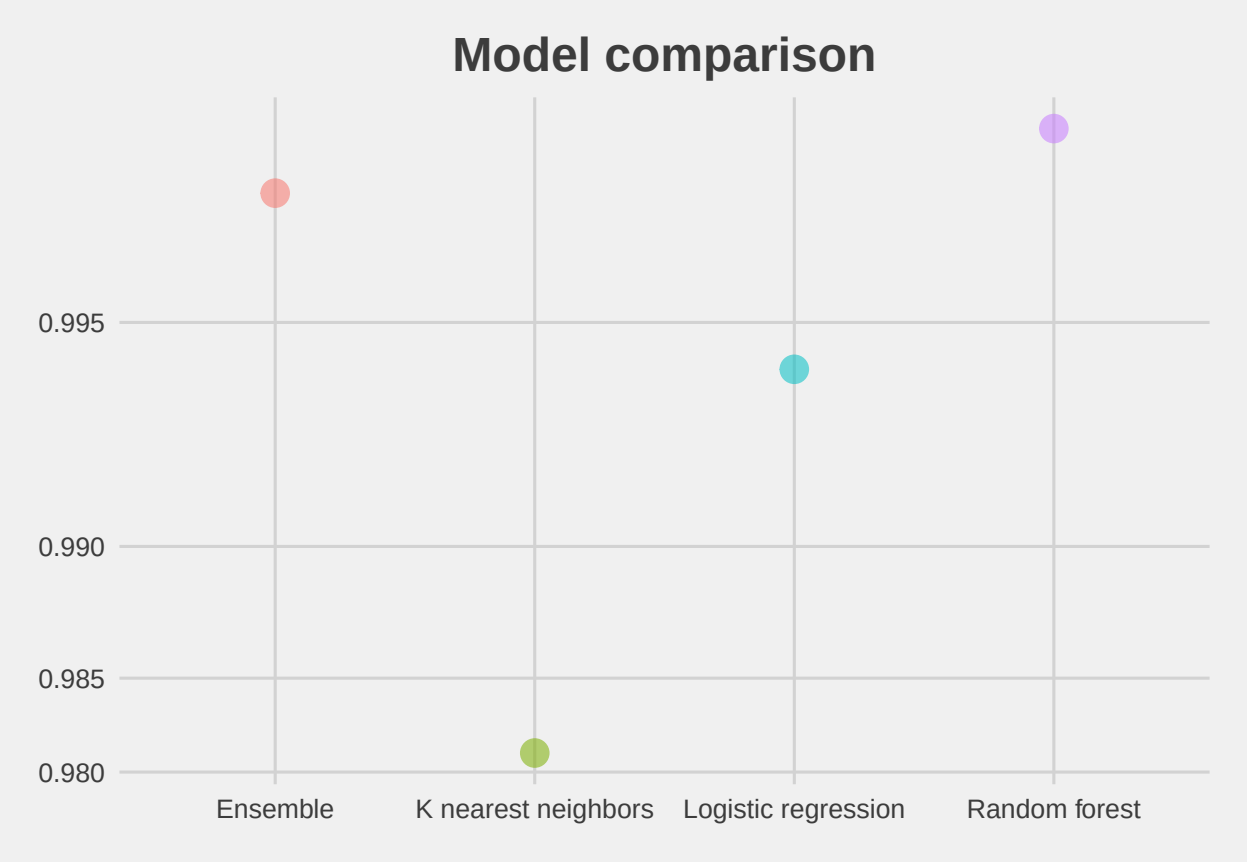
```r
ensemble_preds <- ifelse(rowMedians(models) == 1, 0, 1)

models <- c("Logistic regression", "K nearest neighbors", "Random forest", "Ensemble")
accuracy <- c(mean(glm_preds == test_y),
              mean(knn_preds == test_y),
              mean(rf_preds == test_y),
              mean(ensemble_preds == test_y))
data.frame(Model = models, Accuracy = accuracy) %>% knitr::kable()
```

Model	Accuracy
Logistic regression	0.9942179
K nearest neighbors	0.9811321
Random forest	0.9972611
Ensemble	0.9966525

```r
results <- bind_rows(results, # add accuracy to the df
                          data_frame(method="Ensemble",
                                     accuracy = mean(ensemble_preds == test_y)))
```

Now the Ensemble model improve a lot, with an Accuracy of 0.996. However it´s still second to the accuracy of the Random Forest model, as seen in the next plot.

```r
data.frame(Model = models, Accuracy = accuracy) %>%
  ggplot(aes(x = Model, y = Accuracy, size = 8 , color = Model, alpha = 0.5)) +
  geom_point() +
  theme_fivethirtyeight() +
  ggtitle("Model comparison") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(trans = "logit")
```



The best model is the Random Forest with an accuracy of 0.9975654, that´s pretty high!

Results

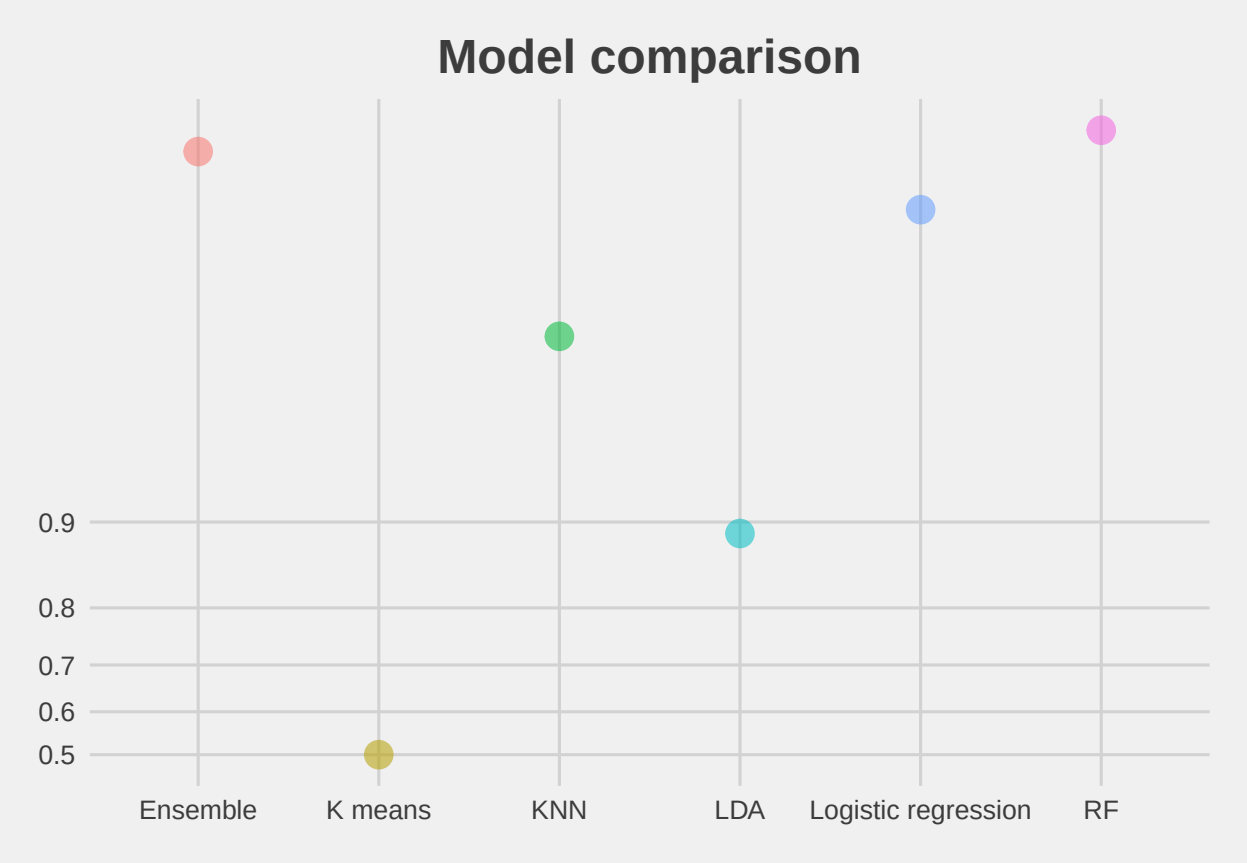
At the end we got 6 different models, included one that combined the best models in the list. The highest accuracy model was the one created with the Random Forest method and the one that had the lowest accuracy was the one created with the K Means method.

```
results %>% knitr::kable()
```

method	accuracy	tune
K means	0.5000000	NA
Logistic regression	0.9942179	NA
LDA	0.8898357	NA
KNN	0.9811321	1.99
RF	0.9972611	5.60
Ensemble	0.9966525	NA

The models accuracy can be better comprehended with the next plot.

```r
results %>%
  ggplot(aes(x = method, y = accuracy, size = 8 , color = method, alpha = 0.5)) +
  geom_point() +
  theme_fivethirtyeight() +
  ggtitle("Model comparison") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(trans = "logit")
```



Conclusion

This dataset was very clean and ready for a machine learning algorithm, however it still needed some modifications in order to make the predictions more accurate. Then, after analizing the data and using some visualization tools, a prediction model for the data was created. Altought the first approach wasn't the best one, the ones created after were really accurate. At the end I got an accuracy of ***0.9975654*** for the Random Forest method, this accuracy is even better than the accuracy of all the methods combined. Therefore I think this model can be used in the real life for Fraud detection because of the very low error margin. However there's still room for improvement, for example I could use the origin and destination data in addtion to all the predictors that I used to try and improve the algorithm, or create a Desicion Tree with all the predictors and see if the overall accuracy is better than the one that I got.