Credit Card Fraud Detection

Abilash Paruchuri

17/06/2022

## Summary

There has been a dramatic increase in the use of credit cards due to rapid technological advancements. With these advances, e-payment and eCommerce systems have been adopted, which has contributed to the increase in the number of fraud cases associated with credit cards. Therefore, to curb this problem, there is a need to implement powerful mechanisms that help to detect these credit card frauds. This project, therefore, implements different machine learning algorithms like random forest and logistic regression to perform analysis and predictions on credit card fraud transactions. The data set implemented for this project is retrieved from the Kaggle.com website. In addition, the project has performed an extensive review of existing literature and models related to credit card fraud detection. A comparative study has been performed to understand the existing gap in this project. As a result, the project has implemented different models. Their performance was evaluated based on confusion matrix, precision, recall, and accuracy to identify the best model for the application of detecting and predicting credit card fraud transactions.

## Introduction

Credit card fraud transactions can be defined as the unauthorized use of an account by an individual other than the authorized owner of that account (Patidar & Sharma, 2011). In most cases, there are a number of preventive measures that have been used to deal with credit card fraud. However, these measures have not been effective since more and more credit frauds are experienced. Therefore, this calls for implementing machine learning and data science to ensure the solution to credit card fraud is automated. This is because, with time, the patterns in credit card fraud often change, making it hard to track them, but with machine learning techniques, it will be easier. These machine learning algorithms help analyze all the authorized transactions and detect suspicious transactions in an automated manner.

## Literature Review

In a study by Mittal & Tyagi (2019), the authors have proposed a hybrid approach that can be implemented in detecting credit card frauds through isolation forest and random forest algorithms in the identification of anomaly-based transactions. However, from this model, there was no confidentiality and privacy. In another study, they identified that for credit card fraud detection, the logistic regression model performs only best in regression problems and not in the real-time non-linear datasets, making it unsuitable for this operation. Xuan et al. (2018) identified that the Random Forest algorithm better detects credit card frauds. A study by Awoyemi et al. (2017) illustrated that constantly changing profiles and standard fraud transactions are the main two problems associated with credit card fraud detection. With the implementation of Naive Bayes, k-Nearest Neighbor, and logistic regression algorithms, k-Nearest Neighbor was identified as the best algorithm for detecting credit card frauds. However, from the previous research, there has been a gap in the research and analysis of credit card fraud detection. Previous systems’ accuracy, confidentiality, and privacy have been an issue. Therefore this project intends to identify the most suitable technique that will have the highest accuracy and provide privacy in credit card fraud detection. ## Theory

H1: For many financial institutions, they are always faced with several credits and debit fraud transactions. H2: There is an association between low accuracy detection systems and a higher number of fraudulent transactions in any financial company.

## Warning: package 'ranger' was built under R version 4.1.2

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.2

## Loading required package: lattice

## Warning: package 'dplyr' was built under R version 4.1.2

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':  
##   
## between, first, last

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

## Data

The dataset Creditcard.csv has been retrieved from <https://www.kaggle.com/mlg-ulb/creditcardfraud>. It includes:

## Time V1 V2 V3 V4 V5 V6  
## 1 0 -1.3598071 -0.07278117 2.5363467 1.3781552 -0.33832077 0.46238778  
## 2 0 1.1918571 0.26615071 0.1664801 0.4481541 0.06001765 -0.08236081  
## 3 1 -1.3583541 -1.34016307 1.7732093 0.3797796 -0.50319813 1.80049938  
## 4 1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888 1.24720317  
## 5 2 -1.1582331 0.87773675 1.5487178 0.4030339 -0.40719338 0.09592146  
## 6 2 -0.4259659 0.96052304 1.1411093 -0.1682521 0.42098688 -0.02972755  
## V7 V8 V9 V10 V11 V12  
## 1 0.23959855 0.09869790 0.3637870 0.09079417 -0.5515995 -0.61780086  
## 2 -0.07880298 0.08510165 -0.2554251 -0.16697441 1.6127267 1.06523531  
## 3 0.79146096 0.24767579 -1.5146543 0.20764287 0.6245015 0.06608369  
## 4 0.23760894 0.37743587 -1.3870241 -0.05495192 -0.2264873 0.17822823  
## 5 0.59294075 -0.27053268 0.8177393 0.75307443 -0.8228429 0.53819555  
## 6 0.47620095 0.26031433 -0.5686714 -0.37140720 1.3412620 0.35989384  
## V13 V14 V15 V16 V17 V18  
## 1 -0.9913898 -0.3111694 1.4681770 -0.4704005 0.20797124 0.02579058  
## 2 0.4890950 -0.1437723 0.6355581 0.4639170 -0.11480466 -0.18336127  
## 3 0.7172927 -0.1659459 2.3458649 -2.8900832 1.10996938 -0.12135931  
## 4 0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279 1.96577500  
## 5 1.3458516 -1.1196698 0.1751211 -0.4514492 -0.23703324 -0.03819479  
## 6 -0.3580907 -0.1371337 0.5176168 0.4017259 -0.05813282 0.06865315  
## V19 V20 V21 V22 V23 V24  
## 1 0.40399296 0.25141210 -0.018306778 0.277837576 -0.11047391 0.06692807  
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953 0.10128802 -0.33984648  
## 3 -2.26185710 0.52497973 0.247998153 0.771679402 0.90941226 -0.68928096  
## 4 -1.23262197 -0.20803778 -0.108300452 0.005273597 -0.19032052 -1.17557533  
## 5 0.80348692 0.40854236 -0.009430697 0.798278495 -0.13745808 0.14126698  
## 6 -0.03319379 0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658  
## V25 V26 V27 V28 Amount Class  
## 1 0.1285394 -0.1891148 0.133558377 -0.02105305 149.62 0  
## 2 0.1671704 0.1258945 -0.008983099 0.01472417 2.69 0  
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66 0  
## 4 0.6473760 -0.2219288 0.062722849 0.06145763 123.50 0  
## 5 -0.2060096 0.5022922 0.219422230 0.21515315 69.99 0  
## 6 -0.2327938 0.1059148 0.253844225 0.08108026 3.67 0

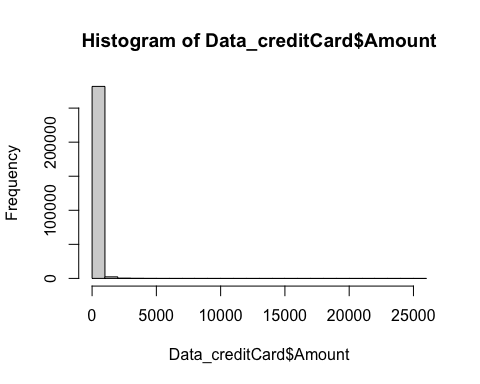
The dataset had to be explored to identify what stands out for preprocessing to build the machine learning models.

# exploring the creditCard data  
head(Data\_creditCard)

## Time V1 V2 V3 V4 V5 V6  
## 1 0 -1.3598071 -0.07278117 2.5363467 1.3781552 -0.33832077 0.46238778  
## 2 0 1.1918571 0.26615071 0.1664801 0.4481541 0.06001765 -0.08236081  
## 3 1 -1.3583541 -1.34016307 1.7732093 0.3797796 -0.50319813 1.80049938  
## 4 1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888 1.24720317  
## 5 2 -1.1582331 0.87773675 1.5487178 0.4030339 -0.40719338 0.09592146  
## 6 2 -0.4259659 0.96052304 1.1411093 -0.1682521 0.42098688 -0.02972755  
## V7 V8 V9 V10 V11 V12  
## 1 0.23959855 0.09869790 0.3637870 0.09079417 -0.5515995 -0.61780086  
## 2 -0.07880298 0.08510165 -0.2554251 -0.16697441 1.6127267 1.06523531  
## 3 0.79146096 0.24767579 -1.5146543 0.20764287 0.6245015 0.06608369  
## 4 0.23760894 0.37743587 -1.3870241 -0.05495192 -0.2264873 0.17822823  
## 5 0.59294075 -0.27053268 0.8177393 0.75307443 -0.8228429 0.53819555  
## 6 0.47620095 0.26031433 -0.5686714 -0.37140720 1.3412620 0.35989384  
## V13 V14 V15 V16 V17 V18  
## 1 -0.9913898 -0.3111694 1.4681770 -0.4704005 0.20797124 0.02579058  
## 2 0.4890950 -0.1437723 0.6355581 0.4639170 -0.11480466 -0.18336127  
## 3 0.7172927 -0.1659459 2.3458649 -2.8900832 1.10996938 -0.12135931  
## 4 0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279 1.96577500  
## 5 1.3458516 -1.1196698 0.1751211 -0.4514492 -0.23703324 -0.03819479  
## 6 -0.3580907 -0.1371337 0.5176168 0.4017259 -0.05813282 0.06865315  
## V19 V20 V21 V22 V23 V24  
## 1 0.40399296 0.25141210 -0.018306778 0.277837576 -0.11047391 0.06692807  
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953 0.10128802 -0.33984648  
## 3 -2.26185710 0.52497973 0.247998153 0.771679402 0.90941226 -0.68928096  
## 4 -1.23262197 -0.20803778 -0.108300452 0.005273597 -0.19032052 -1.17557533  
## 5 0.80348692 0.40854236 -0.009430697 0.798278495 -0.13745808 0.14126698  
## 6 -0.03319379 0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658  
## V25 V26 V27 V28 Amount Class  
## 1 0.1285394 -0.1891148 0.133558377 -0.02105305 149.62 0  
## 2 0.1671704 0.1258945 -0.008983099 0.01472417 2.69 0  
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66 0  
## 4 0.6473760 -0.2219288 0.062722849 0.06145763 123.50 0  
## 5 -0.2060096 0.5022922 0.219422230 0.21515315 69.99 0  
## 6 -0.2327938 0.1059148 0.253844225 0.08108026 3.67 0

From the data, it is clear that there are 28 variables that are anonymous, one amount column, one label column, and one time column. To identify the relation between the variables, a histogram has been created for visualization.

hist(Data\_creditCard$Amount)



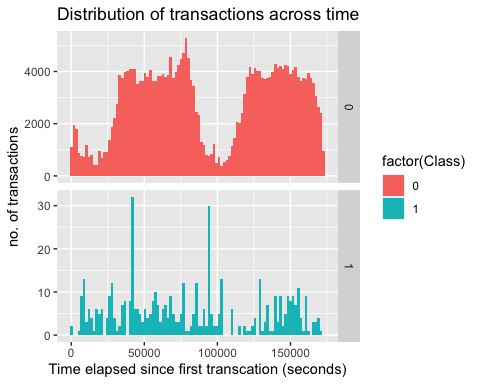
To ensure that the data was clean for analysis and classification, there was the need to check for ny missing values in the columns

# check whether there are any missing values in columns  
colSums(is.na(Data\_creditCard))

## Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10   
## 0 0 0 0 0 0 0 0 0 0 0   
## V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21   
## 0 0 0 0 0 0 0 0 0 0 0   
## V22 V23 V24 V25 V26 V27 V28 Amount Class   
## 0 0 0 0 0 0 0 0 0

It is clear that there are no missing values in the columns for the dataset.

## Methodology

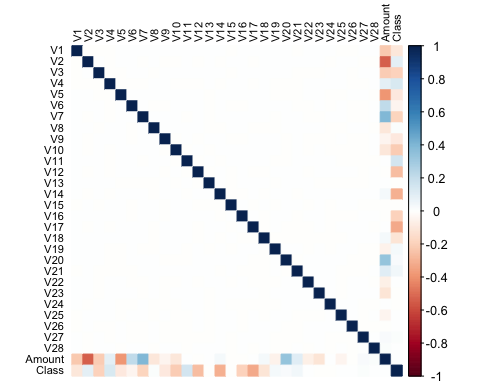
Having identified that there are no missing values, the next thing was to identify what variables are essential and what variables need to be eliminated. Therefore, the first thing involve visualizing the transactions over time to identify if it is an important factor for analysis in this project. 

In the fraud detection, it is clear that time do not contribute so much and therefore was to be removed from the data. To check if there is any correlation between amount and class, and all the variables a correlation plot is created.

#install.packages("corrplot")  
library(corrplot)

## corrplot 0.92 loaded

# correlation of anonymous variables with amount and class  
correlation <- cor(Data\_creditCard[, -1], method = "pearson")  
corrplot(correlation, number.cex = 1, method = "color", type = "full", tl.cex=0.7, tl.col="black")



It was identified that there is no correlation for most of the features. All the anonymous variables are independent to each other. The Amount variable is normalized with mean 0 since all the anonymous variables are standardized.

# scaling the data using standardization and remove the first column (time) from the data set  
Data\_creditCard$Amount <- scale(Data\_creditCard$Amount)  
data\_scaled <- Data\_creditCard[, -c(1)]  
head(data\_scaled)

## V1 V2 V3 V4 V5 V6  
## 1 -1.3598071 -0.07278117 2.5363467 1.3781552 -0.33832077 0.46238778  
## 2 1.1918571 0.26615071 0.1664801 0.4481541 0.06001765 -0.08236081  
## 3 -1.3583541 -1.34016307 1.7732093 0.3797796 -0.50319813 1.80049938  
## 4 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888 1.24720317  
## 5 -1.1582331 0.87773675 1.5487178 0.4030339 -0.40719338 0.09592146  
## 6 -0.4259659 0.96052304 1.1411093 -0.1682521 0.42098688 -0.02972755  
## V7 V8 V9 V10 V11 V12  
## 1 0.23959855 0.09869790 0.3637870 0.09079417 -0.5515995 -0.61780086  
## 2 -0.07880298 0.08510165 -0.2554251 -0.16697441 1.6127267 1.06523531  
## 3 0.79146096 0.24767579 -1.5146543 0.20764287 0.6245015 0.06608369  
## 4 0.23760894 0.37743587 -1.3870241 -0.05495192 -0.2264873 0.17822823  
## 5 0.59294075 -0.27053268 0.8177393 0.75307443 -0.8228429 0.53819555  
## 6 0.47620095 0.26031433 -0.5686714 -0.37140720 1.3412620 0.35989384  
## V13 V14 V15 V16 V17 V18  
## 1 -0.9913898 -0.3111694 1.4681770 -0.4704005 0.20797124 0.02579058  
## 2 0.4890950 -0.1437723 0.6355581 0.4639170 -0.11480466 -0.18336127  
## 3 0.7172927 -0.1659459 2.3458649 -2.8900832 1.10996938 -0.12135931  
## 4 0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279 1.96577500  
## 5 1.3458516 -1.1196698 0.1751211 -0.4514492 -0.23703324 -0.03819479  
## 6 -0.3580907 -0.1371337 0.5176168 0.4017259 -0.05813282 0.06865315  
## V19 V20 V21 V22 V23 V24  
## 1 0.40399296 0.25141210 -0.018306778 0.277837576 -0.11047391 0.06692807  
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953 0.10128802 -0.33984648  
## 3 -2.26185710 0.52497973 0.247998153 0.771679402 0.90941226 -0.68928096  
## 4 -1.23262197 -0.20803778 -0.108300452 0.005273597 -0.19032052 -1.17557533  
## 5 0.80348692 0.40854236 -0.009430697 0.798278495 -0.13745808 0.14126698  
## 6 -0.03319379 0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658  
## V25 V26 V27 V28 Amount Class  
## 1 0.1285394 -0.1891148 0.133558377 -0.02105305 0.24496383 0  
## 2 0.1671704 0.1258945 -0.008983099 0.01472417 -0.34247394 0  
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 1.16068389 0  
## 4 0.6473760 -0.2219288 0.062722849 0.06145763 0.14053401 0  
## 5 -0.2060096 0.5022922 0.219422230 0.21515315 -0.07340321 0  
## 6 -0.2327938 0.1059148 0.253844225 0.08108026 -0.33855582 0

For preparation of creating the models and fitting, the dataset has been split in the ration of 80% training and 20% testing sets.

The training data after split was visualized to identify how balanced the training set is.

# visualize the training data  
data\_train %>% ggplot(aes(x = factor(Class), y = prop.table(stat(count)), fill = factor(Class))) +  
 geom\_bar(position = "dodge") +  
 scale\_y\_continuous(labels = scales::percent) +  
 labs(x = 'Class', y = 'Percentage', title = 'Training Class distributions') +  
 theme\_grey()



Data is heavily unbalanced. It is clear that there is 99% of non-fraudulent data. With this, the model might perform less accurately. Therefore, the data has been sampled through the use of Down sampling technique.

set.seed(90)  
down\_data\_train <- downSample(x = data\_train[, -30],  
 y = data\_train$Class)  
table(down\_data\_train$Class)

##   
## Not Fraud Fraud   
## 394 394

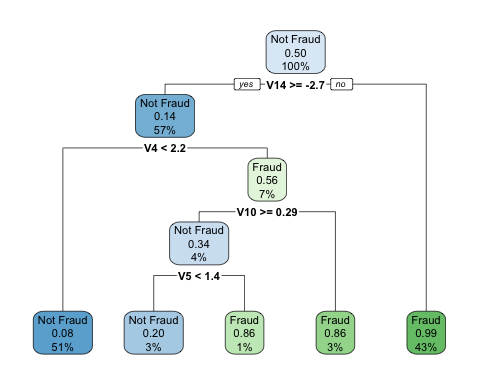
## Results

A decision tree model was first created. From this model, it is identified that the v14 variable is a significant variable that is the most significant in separating the non-fraud and fraud transactions.

#install.packages("rpart.plot")  
library(rpart) #  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.1.2

DT <- rpart(Class ~ . , down\_data\_train, method = 'class')  
predicted\_val <- predict(DT, down\_data\_train, type = 'class')  
probability <- predict(DT, down\_data\_train, type = 'prob')  
rpart.plot(DT)



Second, a Random Forest model has been created. The area under the ROC curve have been identified for the Random Forest model. This means that Random Forest hs an accuracy of 0.9628.

#install.packages("pROC")  
library(Rborist)

## Rborist 0.2-3

## Type RboristNews() to see new features/changes/bug fixes.

library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

x = down\_data\_train[, -30]  
y = down\_data\_train[,30]  
  
rf <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)  
  
  
rf\_pred <- predict(rf, data\_test[,-30], ctgCensus = "prob")

## Warning in data.table::setDT(x): Some columns are a multi-column type (such as  
## a matrix column): [29]. setDT will retain these columns as-is but subsequent  
## operations like grouping and joining may fail. Please consider as.data.table()  
## instead which will create a new column for each embedded column.

prob <- rf\_pred$prob  
roc(data\_test$Class, prob[,2], plotit = TRUE, col = 'blue')

## Setting levels: control = Not Fraud, case = Fraud

## Setting direction: controls < cases

##   
## Call:  
## roc.default(response = data\_test$Class, predictor = prob[, 2], plotit = TRUE, col = "blue")  
##   
## Data: prob[, 2] in 56863 controls (data\_test$Class Not Fraud) < 98 cases (data\_test$Class Fraud).  
## Area under the curve: 0.9622

#install.packages("xgboost")  
library(xgboost)

## Warning: package 'xgboost' was built under R version 4.1.2

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

set.seed(40)  
  
#Convert class labels from factor to numeric  
labels <- down\_data\_train$Class  
y <- recode(labels, 'Not Fraud' = 0, "Fraud" = 1)  
xgbFit <- xgboost(data = data.matrix(down\_data\_train[,-30]),   
 label = y,  
 eta = 0.1,  
 gamma = 0.1,  
 max\_depth = 10,   
 nrounds = 300,   
 objective = "binary:logistic",  
 colsample\_bytree = 0.6,  
 verbose = 0,  
 nthread = 7  
)

The XGBoost model has an accuracy of 0.9696.

# XGBoost predictions  
pred\_xgb <- predict(xgbFit, data.matrix(data\_test[,-30]))  
roc(data\_test$Class, pred\_xgb)

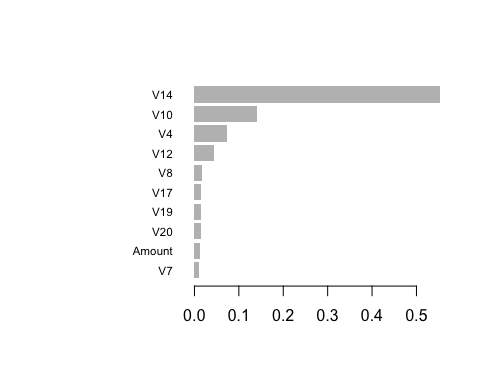
## Setting levels: control = Not Fraud, case = Fraud

## Setting direction: controls < cases

##   
## Call:  
## roc.default(response = data\_test$Class, predictor = pred\_xgb)  
##   
## Data: pred\_xgb in 56863 controls (data\_test$Class Not Fraud) < 98 cases (data\_test$Class Fraud).  
## Area under the curve: 0.9701

Lastly, the project identifies the names of the variables that have a crucial role for fraud detection.

names <- dimnames(data.matrix(down\_data\_train[,-30]))[[2]]  
  
# Compute feature importance matrix  
importance\_matrix <- xgb.importance(names, model = xgbFit)  
# Nice graph  
xgb.plot.importance(importance\_matrix[1:10,])



Similarly, as identified from the Decision Tree model, v14 has stood out to be a significant variable in identifying or distinguishing either transactions are fraud or non fraud.

## Conclusion

The credit fraud prediction in this paper was a success. It was possible to identify the major factors that need to be considered to distinguish between fraud and non-fraud transactions. In addition, from the models, it was identified that the most suitable applicable model in detecting credit card fraud transactions is XGBoost model as it performed better that the Random Forest model.

# References

Mittal, S., & Tyagi, S. (2019, January). Performance evaluation of machine learning algorithms for credit card fraud detection. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 320-324). IEEE.

Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018, March). Random forest for credit card fraud detection. In 2018 IEEE 15th international conference on networking, sensing and control (ICNSC) (pp. 1-6). IEEE.

Patidar, R., & Sharma, L. (2011). Credit card fraud detection using neural network. International Journal of Soft Computing and Engineering (IJSCE), 1(32-38).

Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017, October). Credit card fraud detection using machine learning techniques: A comparative analysis. In 2017 international conference on computing networking and informatics (ICCNI) (pp. 1-9). IEEE.