CREDIT CARD FRAUD ANALYSIS USING MACHINE LEARNING TECHNIQUES

DATA SCIENCE FOUNDATION

AYOOLUWA DORCAS BABALOLA

(B1202001)

Contents

Credit Card Fraud Analysis Using Machine Learning Techniques

Ayooluwa Dorcas Babalola
Student ID: B1202001
Data Science Foundation
School of Computing, Engineering and Digital Technologies
Teesside University

***Abstract*— Online transactions and use of credit card are the preferred method of sales transactions in this age. And with the increasing use of credit card, so does the likelihood of credit card fraud increase. Due to ignorance, many have shared their card details, security pins and one-time passwords to fraudsters posing as bank agents. Immediately these details are released, the account gets wiped off. Loses incurred due to credit card and online transactions fraud have affected both the companies and individuals involved. To prevent more cases of credit card fraud, banks and financial institutions should make use of credit card fraud detection methods.**

This report will analyze the some of the most common fraud detection methods discussed in this research are Support Vector Machine (SVM), K-Nearest Neighbor algorithm (KNN) and Random Forest. This research shows the credit card fraud types, and methodologies on detecting credit card fraud with data sets gotten from professional survey organizations.

This report will be divided into 2 parts. The first part focuses on analyzing previous studies. The other part focuses on the methods used and proffer solutions.

Keywords—credit card fraud detection, support vector machine, K-Nearest Neighbor, random forest.

I. INTRODUCTION

A credit card is a thin plastic or metallic card issued by financial organisations to its users for a variety of purposes, including money withdrawal and deposit, transaction payment, and borrowing money, all of which are usually limited. While credit cards were introduced for good reasons, they have resulted in a significant increase in fraudulent incidents. ATMs, retail readers, swipe machines, online and bank transactions all read these cards.

The unlawful use of a payment card (credit or debit card) to make purchases or make payments into another account is known as credit card fraud. Card fraud transactions totalled 829 billion pounds in 2019, with 22 billion transactions (Fraud - The Facts 2020), indicating that billions of pounds are lost every year.

There are 4 categories of credit card fraud which are further discussed below:

1. Card-not-present (CNP) Fraud

CNP fraud occurs when the card is not with the fraudster and is done online or over the phone, as the term implies. With the rise in the number of e-commerce sites, CNP fraud is becoming increasingly widespread. Precautionary measures to take to avoid CNP fraud include not entering your card information on just any site and not giving out your card information over the phone.

Few credit card companies can suspend your card due to CNP fraud if they observe suspicious activity on it (a large transaction in an unusual area), and it won't be lifted until you produce proof that you completed the transactions.

2. Lost and stolen card fraud

If one’s credit card is lost or stolen, and it is used by credit card fraudsters and one could lose all their money. It is important that one puts a call through to the bank to cancel the card if one doesn’t need it anymore or if it is stolen.

3. False application fraud

If someone gets access to some of a person's personal information (bank statements, utility bills), they can apply for a credit card using that information. They might obtain credit worth thousands of pounds and destroy one's credit report by applying for a credit card in their name.

4. Fraudulent skimming

Skimming occurs when your card information is stolen through an electronic device. Skimming is as simple as having someone pass you by with a skimmer. Fraudulent activity on your account might be carried out once your card details have been obtained.

5. Scamming

When a fraudster poses as someone else to obtain the victim's credit card information, this is known as impersonation fraud. The imposter poses as a bank employee, a senior co-worker, or even a family member. Because you freely handed out your bank account information, it will be difficult to prove that this is a hoax.

Credit Card Fraud Detection

Many credit card fraud detection systems have been installed to check transactions on a regular basis and detect any suspicious activity without interfering with normal operations. Machine learning-based artificial intelligence is an excellent tool for detecting credit card fraud. Machine learning automatically detects these forgeries in real-time streaming data by discovering hidden correlations. Machine learning techniques such as logistic regression, decision trees, random forest, Nave Bayes, Artificial Neural Networks (ANN) model, and Support Vector Machines can be used to categorise and detect fraud. Many of these card detection systems are based on pattern matching and meta learning, in addition to artificial intelligence.

II. LITERTURE REVIEW

In [1] a study was carried out on nine different methods of credit card fraud detection. These methods focused on the speed, accuracy and cost of the detection mechanisms and were thereafter compared. This study showed the strengths and weaknesses associated with each of these methods and concluded that Fuzzy Neural Networks, Support Vector Machines, K-Nearest Neighbor, Artificial Immune System and Bayesian Network are fast in detecting credit card fraud.

In [2] the study shows the prediction of credit card in numerous banking system using different machine learning techniques and algorithms. According to them, these banking systems must be always proactive to serve their customers. The prediction model that gave the highest accuracy of more than 80% in this work is the random forest technique.

In [3] the study proposed genetic algorithms to help in fraud detection. This optimization technique aims to obtain an evolved solution which reforms artificial structures using initialization, selection, crossover, and mutation operators.

In [4] the accuracy of credit card fraud prediction was compared using four different techniques: Random Forest, Naïve Bayes, Logistic Regression and Multi-layer Perceptron using the same dataset as in this work. The Random Forest technique was said to have 99.95% accuracy in the prediction of the credit card fraud detection. This technique was recommended to banks to strengthen their security system.

III. METHODOLOGY

A. *Problem Statement*

The purpose of this study is to investigate which machine learning algorithm is better for predicting whether a credit card transaction is fraudulent or not. We display the target variable class's distribution, consider missing and null values, monitor variable correlation, analyse the data, and predict it using a variety of machine-learning algorithms.

B. *Overview of Data*

This data is a comma separated values (csv) file collected from Kaggle, “Credit Card Fraud Detection” [5].

The dataset contains transactions made by European credit cardholders in two days of September 2013. The data contains 492 fraud transactions from a total transaction of 284,207. Also, majority of the attributes (named V1…V28) of the dataset are encrypted and this has withheld the access to more background information due to confidentiality. Time, Amount and Class are the only declared features in the dataset.

Time – the time in seconds elapsed between each transaction from the first transaction.

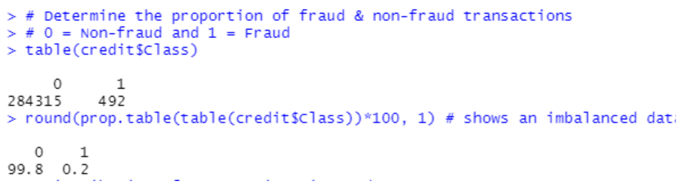
Amount – the transaction amount

Class - is the response variable and it takes value 1 in case of fraud and 0 for non-fraud.

Also, the model was split into the training and test sets which was used for further analysis.

C. *Data Pre-processing*

First, we check for the proportionality of the target variable for the number and percentage of the non-fraud transactions (0) to the fraud transactions (1). We find that we have 284315 non-fraud transaction making 99.8% of the total transaction and the fraud transaction to be 492 which is 0.2% of the total transactions recorded at that time.



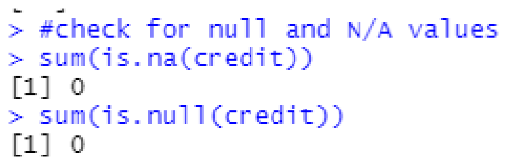
```
> # Determine the proportion of fraud & non-fraud transactions
> # 0 = Non-fraud and 1 = Fraud
> table(credit$Class)

     0      1
284315    492
> round(prop.table(table(credit$Class))*100, 1) # shows an imbalanced dat

   0    1
99.8  0.2
```

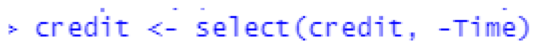
The following pre-processing steps were taken in preparation for the machine learning models to ensure accuracy:

- Checked for missing and NA values and found none



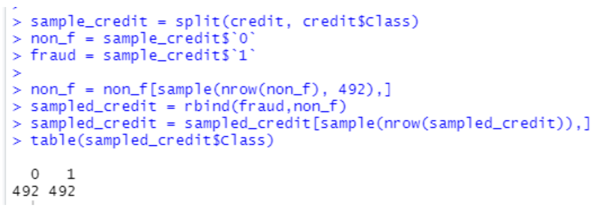
```
> #check for null and N/A values
> sum(is.na(credit))
[1] 0
> sum(is.null(credit))
[1] 0
```

- Encoded the target variable into numeric using the as.factor() method
- The ‘time’ feature was removed for further analysis in the model. This attribute was removed because it doesn’t directly affect when or when not a fraud transaction is committed.



```
> credit <- select(credit, -Time)
```

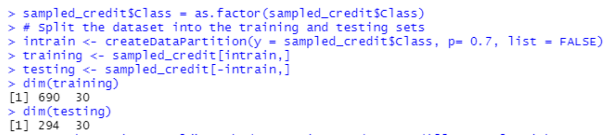
- Data reduction using the sampling method: We reduce the number of non-fraud transaction to the total number of fraud transactions to achieve a balance dataset and give a more accurate prediction. We use the sampling method.
  We first **‘split’** the target variable ‘class’ into its unique fraud and non-fraud transaction. We then **down-sample** the non-fraud class to 492 to be an equal number with the fraudulent transactions. Then we bind the transactions together again using **‘rbind’** and sample using the **‘sample’** function.



```
> sample_credit = split(credit, credit$Class)
> non_f = sample_credit$`0`
> fraud = sample_credit$`1`
>
> non_f = non_f[sample(nrow(non_f), 492),]
> sampled_credit = rbind(fraud,non_f)
> sampled_credit = sampled_credit[sample(nrow(sampled_credit)),]
> table(sampled_credit$Class)

  0   1
492 492
```

- Split the data into the training and the testing sets: To evaluate our model's predicted accuracy, it seems logical to divide data into a training portion and a test portion, allowing the model to be trained on one dataset while being tested on a different, new dataset. We train the data for model fitting and test the data for the estimation of the model’s accuracy. In this study, we divide the training and testing trains into 70 and 30 percent respectively find their dimensions.



```
> sampled_credit$Class = as.factor(sampled_credit$Class)
> # Split the dataset into the training and testing sets
> intrain <- createDataPartition(y = sampled_credit$Class, p= 0.7, list = FALSE)
> training <- sampled_credit[intrain,]
> testing <- sampled_credit[-intrain,]
> dim(training)
[1] 690  30
> dim(testing)
[1] 294  30
```

D. *Exploratory Data Analysis*

Exploratory Data Analysis is very important in finding trends and patterns in the dataset.

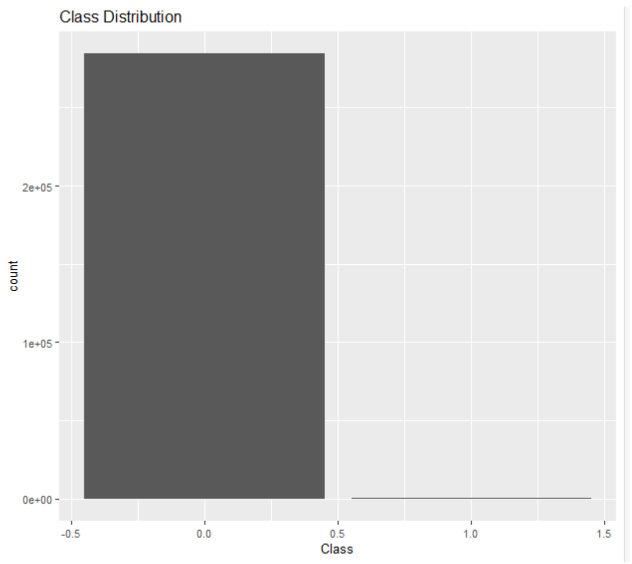


Fig. 1: A graph of fraudulent and non-fraudulent transactions

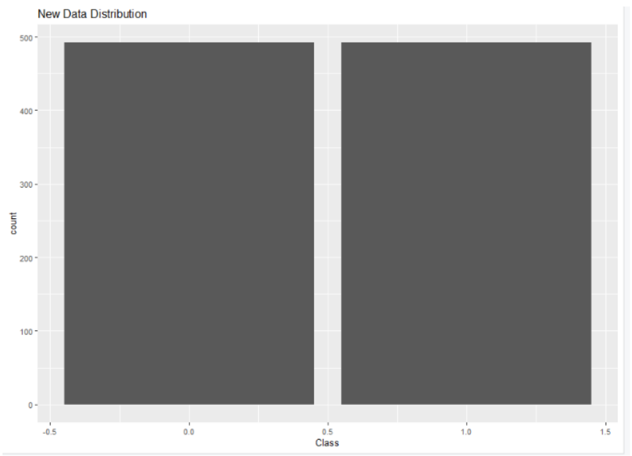


Fig. 2: A graph of fraudulent and non-fraudulent transactions after balancing the distribution.

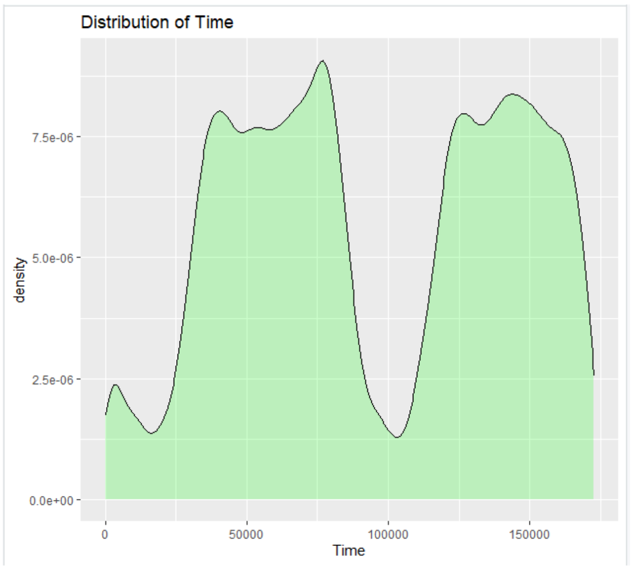


Fig. 3: A graph showing the distribution of the time of all transactions

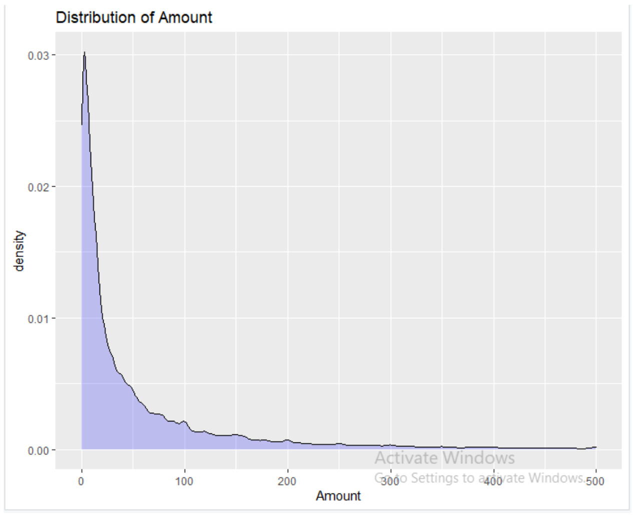


Fig. 4: A graph showing the distribution of amount. This further shows that there are more fraudulent transactions of £10.

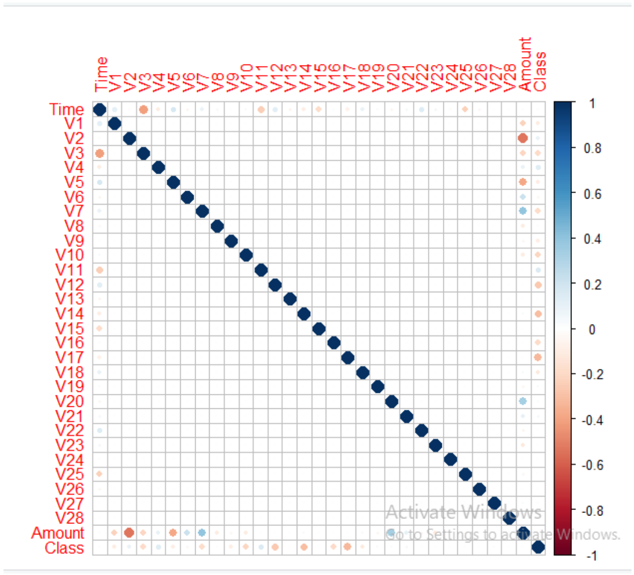


Fig. 5: The correlation plot. This shows that the correlation amount variables are insignificant due to the encrypted features and imbalanced dataset.

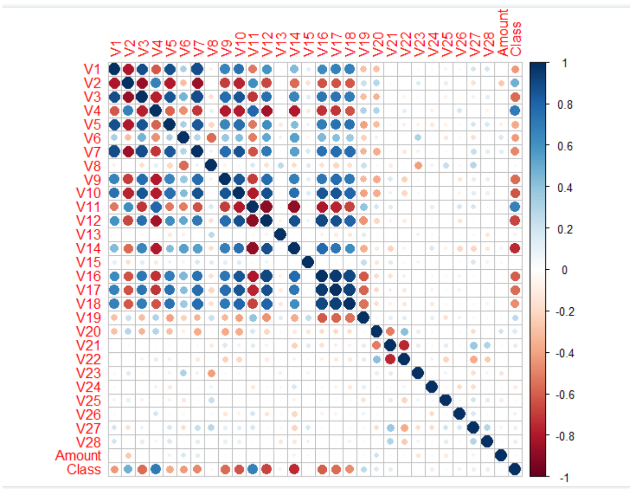


Fig. 6: Correlation plot after sampling which shows that the variables have become more correlated

IV. IMPLEMENTED MACHINE LEARNING TECHNIQUES

In this work, three machine learning techniques have been used to detect, predict, and check the accuracy of credit card fraud deduction dataset. These three techniques are Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and Random Forest. These techniques are explained in the paragraphs below

.

A. *Support Vector Machine (SVM)*

SVM is a supervised learning machine algorithm that creates a hyperplane that separates data into two classes- positive and negative. The classifier finds the separation between the hyperplanes. The maximum separation gives the hyperplane at the edge, which are called the support vectors. This method can be used to detect credit card fraud because it is well suited for binary classification. It must be trained to obtain a learned model just as any artificial intelligence tool [6].

In detecting credit card fraud, if the test is still at the centred region, it is considered normal and to get the most accurate SVM predictions, you will need a large set of data. In an imbalanced set, the result will tend towards the positive set which will make the result not so accurate therefore making SVM suffer from large number of instances just like in this dataset. To overcome this problem, only selected instances are used from the target variable in this model using the sampling method.

In Confusion Matrix, the row signifies the actual class while the column signifies the predicted class. True positive (TP) is the actual class of fraud, True Negative (TN) is the actual class of non-fraudulent transactions, False Positive (FP) was wrongly classified as fraud and False Negative (FN) was wrongly classified as non-fraud. All-together, 18 transactions were wrongly classified with 13 false positive values and 5 false negative. The accuracy achieved with this method is 94%. We then go ahead to plot a four-fold plot and a ROC curve of the SVM method.

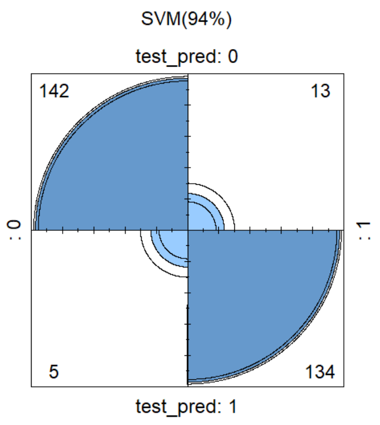


Fig. 7: Four-fold plot of the SVM prediction

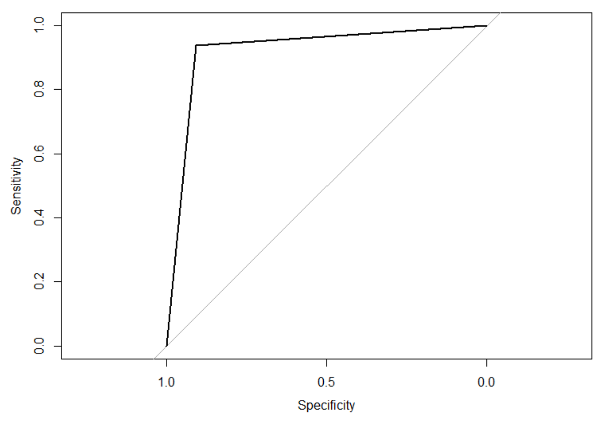


Fig. 8: ROC for SVM

Testing = 294	Predicted = 0	Predicted = 1
Actual (0)	TP = 142	FP = 13
Actual (1)	FN = 5	TN = 134

B. *K-Neareast Neighbor (KNN)*

The K-nearest neighbor algorithm is a non-parametric classification algorithm. It employs the k -nearest value to classify a new observation. The algorithm is dependent on the k value chosen, however the simplest approach to choose k is to run the program multiple times with various k values and select the best performing model. KNN is simple to implement and can withstand noisy training data. Also, it works effectively with large data sets.

In this study, we first create another dataframe for class, our target variable and train and test the sampled data. We then proportion the classification and assign our k-values. Moving on compute the confusion matrix and optimize to prove the accuracy of the model. 40 predictions were wrongly predicted with 28 false negatives and 12 false positives and 86% accuracy.

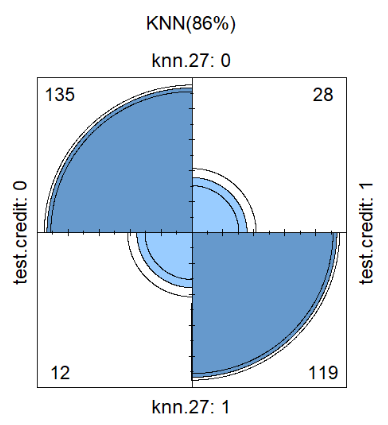


Fig. 9: Four-fold plot of the KNN prediction

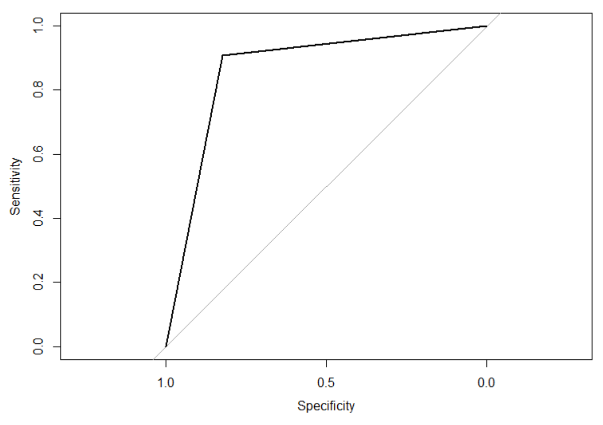


Fig. 10: ROC for KNN

Testing = 294	Predicted = 0	Predicted = 1
Actual (0)	TP = 135	FP = 28
Actual (1)	FN = 12	TN = 119

C. *Random Forest*

The Random Forest selects the features that are independent variables at random, as well as the rows by random. It is developed This is one of the most extensively used machine learning algorithms. Low Bias and High Variance are two features of decision trees. Low bias suggests that if a decision tree is built to its full depth, it will be appropriately trained for the training data set with minimal training error. Random forest gives a very good estimate stating which variables are important in the classification. 20 predictions were wrongly predicted with 16 false negatives and 4 false positives and 93% accuracy.

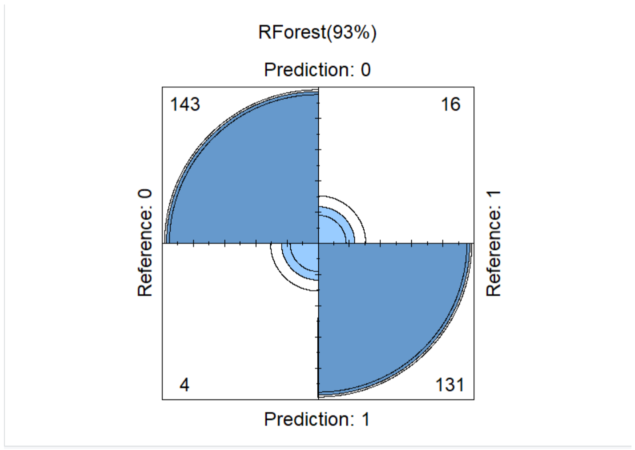


Fig. 11: Four-fold plot of the Random Forest prediction

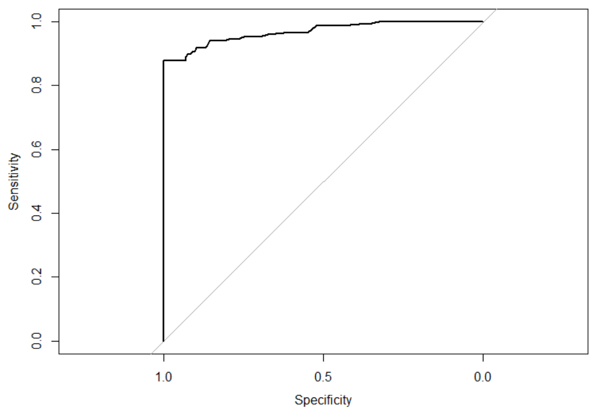


Fig. 12: ROC for Random Forest

N = 294	Predicted = 0	Predicted = 1
Actual (0)	TP = 143	FP = 16
Actual (1)	FN = 4	TN = 131

V. RESULTS AND DISCUSSION

This study began with a clear problem statement and objectives. On this foundation, a data mining methodology was chosen and applied, based on the examination of three possible algorithms, to facilitate the defined objectives, answer the issue, and conduct a relevant analysis. Based on the results of the study, three classifiers were used to accommodate the study's indicated goals.

Based on the metrics gathered, all the models performed admirably. To establish which metric(s) is/are most relevant, we must analyze the real-world ramifications.

Upon comparing all the three models, it was shown that the SVM algorithm has a higher accuracy level in predicting credit card fraud and a higher kappa as compared to the KNN and Random Forest.

This suggests that Support Vector Machine (SVM) will be an excellent model for predicting whether following transactions would be fraudulent. This machine learning will be of enormous assistance to banks and other financial institutions in reducing fraudulent transactions and losses to the bare minimum for both the consumer and the bank.

Performance Evaluation

Models	Machine Learning Algorithms			
	Accuracy	*Prediction*	*Recall*	*AUC*
SVM	0.939	0.905	0.9633	0.940
KNN	0.864	0.819	0.901	0.868
Random Forest	0.932	0.897	0.958	0.978

VI. CONCLUSION

From the exploratory data analysis, it shows that fraud can happen at any time and does not have to be the peak of transactions. Also, a large amount of fraud occurs in small transactions.

To better understand which algorithm would perform best on the given dataset, the following algorithms are used:

- Support Vector Machine
- K-Nearest Neighbor
- Random Forest

These machine learning classification techniques and methods were used to analyze and predict the accuracy of credit card fraud detection after which they were compared in terms of their accuracy to know which gave the best accuracy and prediction rate. Accuracy is important because financial institutions desire a model that can accurately detect both fraudulent and non-fraudulent activities. When the cost of a false positive is high, precision is a valuable metric to consider. In this situation, the cost to the financial institution is greater in terms of total customer satisfaction if a real non-fraud transaction is forecasted as fraud.

According to the four-fold plot, the KNN model gave an accuracy of 86%. The SVM and Random Forest models both gave high accuracies and are best depending on our need which are 94% and 93% respectively.

For this purpose of this work, the Support Vector Machine (SVM) model with a higher accuracy, AUC and lower false negative rate will be taken as the best fit. However, this does not mean that others should be discarded.

VII. REFERENCES

[1] Masoumeh Zareapoor, Seeja.K.R and M. Afshar. Alam (2012), “Analysis of Credit Card Fraud Detection Techniques: Based on Certain Design Criteria”, International Journal of Computer Applications (0975- 8887) Volume 52 – No 3.

[2] Y. Sayjadah, I.A.T. Hashem, F. Alotaibi and K.A. Kasmiran (2018), “Credit Card Default Prediction Using Machine Learning Techniques”, Fourth International Conference on Advances in Computing, Communication Automation (ICACCA).

[3] Rinky D. Patel and Dheeraj Kumar Singh (2013), “Credit Card Fraud Detection & Preventing Fraud Using Genetic Algorithm”, International Journal of Computing and Engineering (IJSCE) Volume 2, Issue 6, January 2013.

[4] Muhammad Zeeshan Younas (2020), “Credit Card Fraud Detection Using Machine Learning Algorithms”, Universe International Journal of Interdisciplinary Research, Volume 1 Issue 4, September 2020.

[5]https://www.kaggle.com/mlg-ulb/creditcardfraud

[6] V. Dheepa and R. Dhanapal (2020), “Behaviour Based Credit Card Fraud Detection Using Support Vector Machines”, ICTAT Journal on Soft Computing, Volume 2, Issue 4.

VIII.APPENDIX

```
# Credit Card Fraud Detection using Machine Learning Techniques in R

# Install the required packages and import the libraries

install.packages("tidyverse")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("caret")
install.packages("data.table")
install.packages("plotly")
install.packages("corrplot")
install.packages("kernlab")
install.packages("e1071")
install.packages("class")
install.packages("randomForest")
install.packages("pROC")


library(tidyverse)
library(dplyr)
library(ggplot2)
library(caret)
library(data.table)
library(plotly)
library(corrplot)
library(kernlab)
library(e1071)
library(class)
library(randomForest)
library(pROC)


# Load the dataset
credit <- read.csv("C:/Users/b1202001/OneDrive - Teesside University/Data  Science Foun
dations/MY ICA/creditcard.csv", sep =',', header = TRUE)

# Data Exploration
# View the structure of the dataset
str(credit)
# View the summary
summary(credit)
# View the head and tail
```

```r
head(credit, 5)
tail(credit, 5)

# Attributes of the dataset
names(credit)

#check for null and N/A values
sum(is.na(credit))
sum(is.null(credit))

# Summary of Amount
summary(credit$Amount)
sd(credit$Amount)
var(credit$Amount)

# Data Exploration

# Visualize the class of fraud and non-fraud transactions
ggplot(credit, aes(x = Class, fill = Class)) +
  geom_bar() +
  ggtitle("Class Distribution")

# Determine the proportion of fraud & non-fraud transactions
# 0 = Non-fraud and 1 = Fraud
table(credit$Class)
round(prop.table(table(credit$Class))*100, 1) # shows an imbalanced data set

# Distribution of transaction time and amount
ggplot(credit, aes(Time)) +
  scale_x_continuous() +
  geom_density(fill = 'Green', alpha = 0.2) +
  labs(title = "Distribution of Time")

# Distribution of Amount
ggplot(credit, aes(Amount)) +
  scale_x_continuous() +
  geom_density(fill = 'Blue', alpha = 0.2) +
  labs(title = "Distribution of Amount") +
  xlim(0,500)


credit %>%
  filter(Amount < 10, Class == 1) %>%
  ggplot(aes(x = Amount, fill = Class)) +
  geom_histogram() +
  ggtitle('Count of Fraud under 10')

credit %>%
  filter(Amount >= 10, Class == 1) %>%
  ggplot(aes(x = Amount, fill = Class)) +
  geom_histogram() +
  ggtitle('Count of Fraud over 10 Amount')


# Correlation Plot
credit$Class <- as.numeric(credit$Class)
M <- cor(credit)
corrplot(M, method = 'circle')
# The correlation plot shows that the amount  attribute is insignificant
# This can be due to the imbalanced dataset or encrypted data

credit <- select(credit, -Time)   # removing time from the data
head(credit_data)


# Data Modeling
# The data is imbalanced and this is common in fraud detection. This factor would affec
t the prediction of the model
# This would be balanced using the sampling method 'sample'
# We first split the data set into their unique fraud and non-fraud transactions

sample_credit = split(credit, credit$Class)
non_f = sample_credit$`0`
fraud = sample_credit$`1`

non_f = non_f[sample(nrow(non_f), 492),]
sampled_credit = rbind(fraud,non_f)
sampled_credit = sampled_credit[sample(nrow(sampled_credit)),]
table(sampled_credit$Class)

# Visualisation of new data distribution

ggplot(sampled_credit, aes(x = Class,  fill = Class)) +
  geom_bar() +
  theme(text = element_text(size=10)) +
  ggtitle("New Data Distribution")

# New correlation plot
# Variables have become more correlated
sampled_credit$Class <- as.numeric(sampled_credit$Class)
M <- cor(sampled_credit)
corrplot(M, method = 'circle')

# Distribution of amount with time
ggplot(credit, aes(x = Amount, fill = Class)) +
  geom_density(alpha = 0.33) +
  scale_x_continuous(limits = c(0, 250), breaks = seq(0, 250, 50)) +
  labs(title = "Distribution of Class with Amount",
       x = "Amount",
       y = "Density",
       col = "Class") +
  scale_fill_discrete(labels = c("Not Fraud, Fraud")) +
  theme_minimal() +
  theme(plot.background = element_rect("cornsilk2"),
        panel.background = element_rect("khaki2"),
        plot.title = element_text(face='bold', color = 'skyblue',
                                  hjust = 0.5, size = 12))


sampled_credit$Class = as.factor(sampled_credit$Class)

# Split the dataset into the training and testing sets
intrain <- createDataPartition(y = sampled_credit$Class, p= 0.7, list = FALSE)
training <- sampled_credit[intrain,]
testing <- sampled_credit[-intrain,]

dim(training)
dim(testing)

# Use the trainControl() method to train our data on different algorithms
trainctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

svm_Linear <- train(Class~., data = training, method = "svmLinear", trControl = trainct
rl, preProcess = c("center", "scale"), tuneLength = 10)
svm_Linear


# Test prediction
test_pred <- predict(svm_Linear, newdata = testing)
test_pred

# Use confusion matrix to predict the accuracy of our model
SVM_CF <- confusionMatrix(table(test_pred, testing$Class))
SVM_CF

# ROC of SVM
roc_SVM <- roc(as.numeric(test_pred), as.numeric(testing$Class))
plot(roc_SVM)
auc(roc_SVM)

# Use of KNN algorithm to build the model
# Finding the number of observation
NROW(training)

#Creating seperate dataframe for 'Class' feature which is our target.
train.credit <- sampled_credit[intrain,30]
test.credit <-sampled_credit[-intrain,30]

knn.27 <- knn(train=training, test=testing, cl=train.credit, k = 27)
knn.28 <- knn(train=training, test=testing, cl=train.credit, k= 28)
```

```r
#Calculate the proportion of correct classification for k = 27, 28
ACC.27 <- 100 * sum(test.credit == knn.27)/NROW(test.credit)
ACC.28 <- 100 * sum(test.credit == knn.28)/NROW(test.credit)

ACC.27
ACC.28

# Check prediction against actual value in tabular form for k=27 and 28
table(knn.27 ,test.credit)
table(knn.28 ,test.credit)

# Use confusionmatrix with KNN to check the accuracy
KNN_CF <- confusionMatrix(table(knn.27 ,test.credit))
KNN_CF


#Optimization
i=1
k.optm=1
for (i in 1:28){ knn.mod <- knn(train=training, test=testing, cl=train.credit, k=i)
k.optm[i] <- 100 * sum(test.credit == knn.mod)/NROW(test.credit)
k=i
cat(k,'=',k.optm[i],'')}

# Graphically represent the accuracy
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")

# ROC for KNN
knn_roc <- roc(as.numeric(knn.27),as.numeric(test.credit))
plot(knn_roc)
auc(knn_roc)

# Random Forest
randforest_model <- randomForest(Class~., data = training,
                                 type = "classification", ntree = 100, mtry = 1)
randforest_model

# Testing
randforest_predict <- predict(randforest_model, testing, type = "class")
randforest_CF <- confusionMatrix(data = randforest_predict, testing$Class)
randforest_CF

# ROC for Random Forest
roc_randforest <- roc(testing$Class,
    predict(randforest_model, newdata = testing, type  = 'prob')[,2],
    plot=T)
roc_randforest
auc(roc_randforest)
```

```r
# Four fold plot of all the models

fourfoldplot(SVM_CF$table,
             main = paste('SVM(', round(SVM_CF$overall[1]*100), '%)', sep = ''))

fourfoldplot(KNN_CF$table,
             main = paste('KNN(', round(KNN_CF$overall[1]*100), '%)', sep = ''))

fourfoldplot(randforest_CF$table,
             main = paste("RForest(", round(randforest_CF$overall[1]*100), "%)", sep =
""))
```