STUDY ON CREDIT CARD FRAUD DETECTION

Department of Statistics

University of Allahabad

PROJECT REPORT

Submitted as a Fulfillment of M.Sc. Final Year

Major Project 2022

Under Supervision

Dr. G. Madhusudan

Assistant Professor

Department of Statistic

University of Allahabad

Submitted By

Riya Vishwakarma

M.Sc. Final Year(2022)

Enrollment No.-U2075016

University of Allahabad

CERTIFICATE

This is to certify that Ms. Riya Vishwakarma has done the major project on the topic Study on Credit Card Fraud Detection under the supervision of Dr. G. Madhusudan.This report is submitted to the department of Statistics, University of Allahabad for the partial fulfilment of requirement for the award of the degree of Master of Science in Statistics.

Dr. G. Madhusudan
(Supervisor)

Prof. Shekhar Srivastava
(Head of Department)

ACKNOWLEDGMENTS

This project has been successful and it would have not been possible to complete it without efforts of the following people. So hereby I would like to convey my thanks and regards to them. I would like to express my heartfelt gratitude to my supervisor Dr. G. Madhusudan for providing his valuable guidance , comments, suggestions and most importantly the way he motivated and inspired me to work harder throughout the course of the project.

I would also like to thanks our faculty members Dr. P.S. Pundir, Dr. Girish Chandra, Dr. Abhay Pratap Pandey and Dr. Priyanka Singh and our previous guest faculty member Mr. Pulkit Srivastava, Dr. Anuj Singh, Dr. Amit Kumar , Mr. Prashant Kumar Sonker , Dr. Rohit Patawa and special thanks to my Co-ordinator Mr. Akhilesh Kumar from the core of my heart for constantly mentoring and guiding us to complete this project, without them accomplishing this would have been very tough.

I would also like to express my appreciation towards my seniors, batch mates and special thanks to Syed Faizul Abbas, Satyam Kushwaha and Yashjeet Pratap Singh and juniors for motivating me throughout my project.

ABSTRACT

Analysis of credit scoring is an effective credit risk assessment technique, which is very necessary for the field of banking sector. Machine learning has a variety of applications in the banking sector and it has been widely used for data analysis. Modern techniques such as machine learning have provided a self -regulating process to analyze the data using classification techniques. The classification method is supervised learning process in which the computer learns from input data provided and makes use of this information to classify the new dataset. A credit transaction that needs to be accepted or rejected is trained and implemented on the dataset using a different machine learning algorithm. In this project we'll go through the Random Forest Classifier and Logistic Regression. For carrying out the credit card fraud detection ,we will make use of the card transactions dataset that contains a mix of fraud as well as non -fraudulent transactions.

CONTENTS

1. OBJECTIVES

1. Detect Credit Card Fraud with Machine Learning in PYTHON.

2. To Compare Logistics Regression and Random Forest Classifier for Credit Card fraud detection.

2. INTRODUCTION

'Fraud' in credit card transactions is unauthorized and unwanted usage of an account by someone other than the owner of that account. Necessary prevention measures can be taken to stop this abuse and the behaviour of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future. In other words, Credit Card Fraud can be defined as a case where a person uses someone else's credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used. Fraud detection involves monitoring the activities of populations of users in order to estimate, perceive or avoid objectionable behaviour, which consist of fraud, intrusion, and defaulting. This is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated. This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of time.

These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize. Machine learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the

transaction was genuine or fraudulent. The investigators provide a feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time.

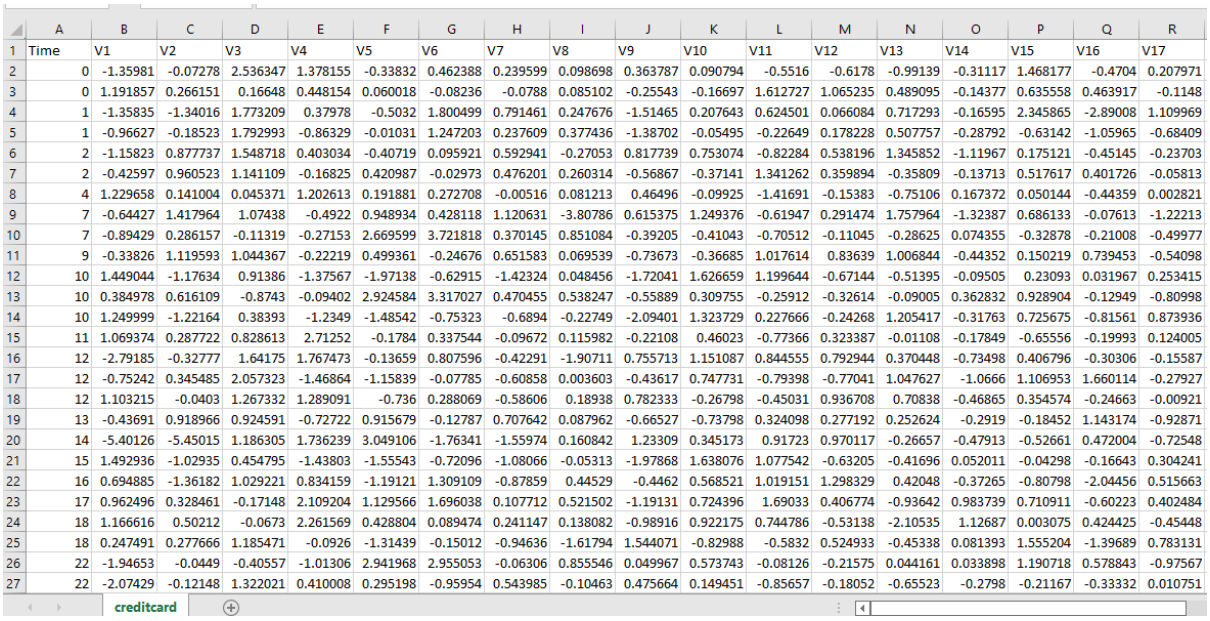
Fraud detection methods are continuously developed to defend criminals in adapting to their fraudulent strategies. These frauds are classified as:

- Credit Card Frauds : Online and Offline
- Card Theft
- Account Bankruptcy
- Device Intrusion
- Application Fraud
- Counterfeit Card
- Telecommunication Fraud

3. DATA SOURCE AND DESCRIPTION

First of all, we obtained our dataset from github, a data analysis website which provides datasets. Inside this dataset, there are 31 columns out of which 28 are named as v1-v28 to protect sensitive data. The other columns represent Time, Amount and Class. Time shows the time gap between the first transaction and the following one. Amount is the amount of money transacted. Class 0 represents a valid transaction and 1 represents a fraudulent:

https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_3_GettingStarted/SimulatedDataset.html.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
2	0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971
3	0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148
4	1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969
5	1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409
6	2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703
7	2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813
8	4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821
9	7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.291474	1.757964	-1.32387	0.686133	-0.07613	-1.22213
10	7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977
11	9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098
12	10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.67144	-0.51395	-0.09505	0.23093	0.031967	0.253415
13	10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998
14	10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729	0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936
15	11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005
16	12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587
17	12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731	-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927
18	12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798	-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24663	-0.00921
19	13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798	0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871
20	14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842	1.23309	0.345173	0.91723	0.970117	-0.26657	-0.47913	-0.52661	0.472004	-0.72548
21	15	1.492936	-1.02935	0.454795	-1.43803	-1.55543	-0.72096	-1.08066	-0.05313	-1.97868	1.638076	1.077542	-0.63205	-0.41696	0.052011	-0.04298	-0.16643	0.304241
22	16	0.694885	-1.36182	1.029221	0.834159	-1.19121	1.309109	-0.87859	0.44529	-0.4462	0.568521	1.019151	1.298329	0.42048	-0.37265	-0.80798	-2.04456	0.515663
23	17	0.962496	0.328461	-0.17148	2.109204	1.129566	1.696038	0.107712	0.521502	-1.19131	0.724396	1.69033	0.406774	-0.93642	0.983739	0.710911	-0.60223	0.402484
24	18	1.166616	0.50212	-0.0673	2.261569	0.428804	0.089474	0.241147	0.138082	-0.98916	0.922175	0.744786	-0.53138	-2.10535	1.12687	0.003075	0.424425	-0.45448
25	18	0.247491	0.277666	1.185471	-0.0926	-1.31439	-0.15012	-0.94636	-1.61794	1.544071	-0.82988	-0.5832	0.524933	-0.45338	0.081393	1.555204	-1.39689	0.783131
26	22	-1.94653	-0.0449	-0.40557	-1.01306	2.941968	2.955053	-0.06306	0.855546	0.049967	0.573743	-0.08126	-0.21575	0.044161	0.033898	1.190718	0.578843	-0.97567
27	22	-2.07429	-0.12148	1.322021	0.410008	0.295198	-0.95954	0.543985	-0.10463	0.475664	0.149451	-0.85657	-0.18052	-0.65523	-0.2798	-0.21167	-0.33332	0.010751

creditcard

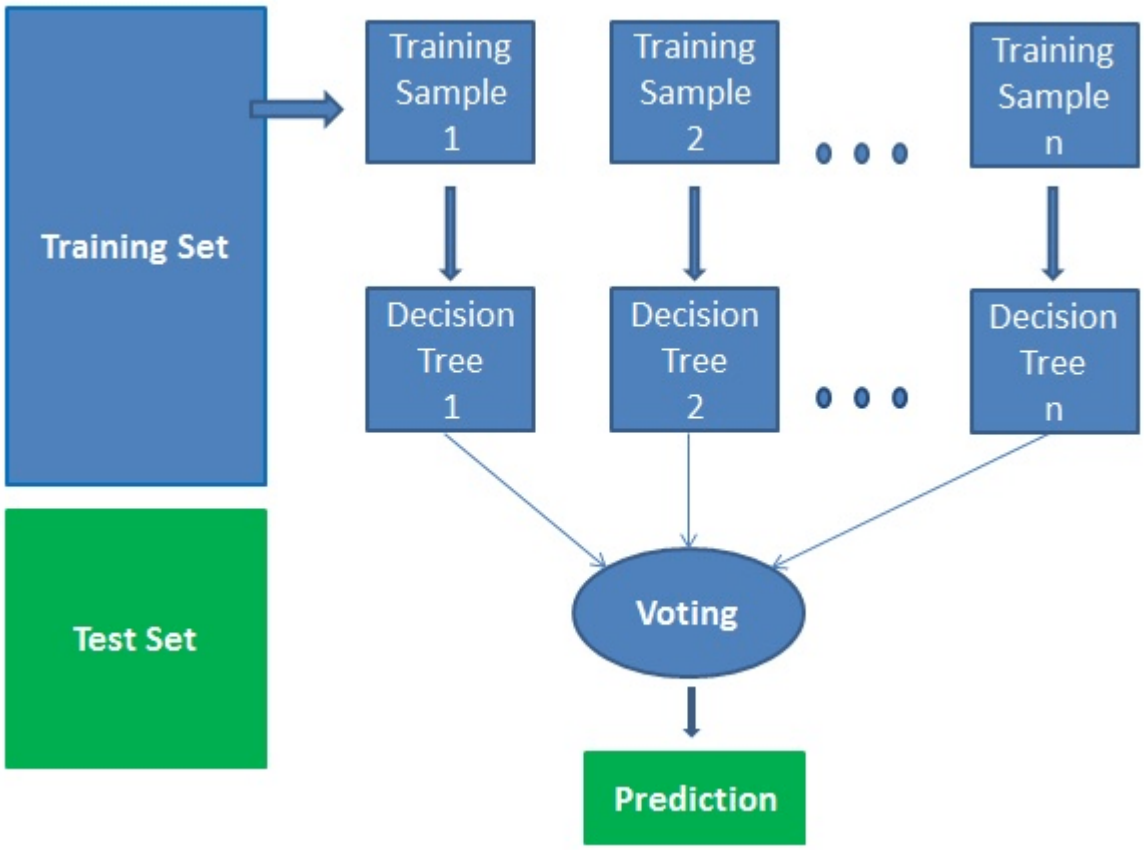
4. METHODOLOGY

4.1 Random Forest Classifier: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning**, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "**Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset**."Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest Classifier:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

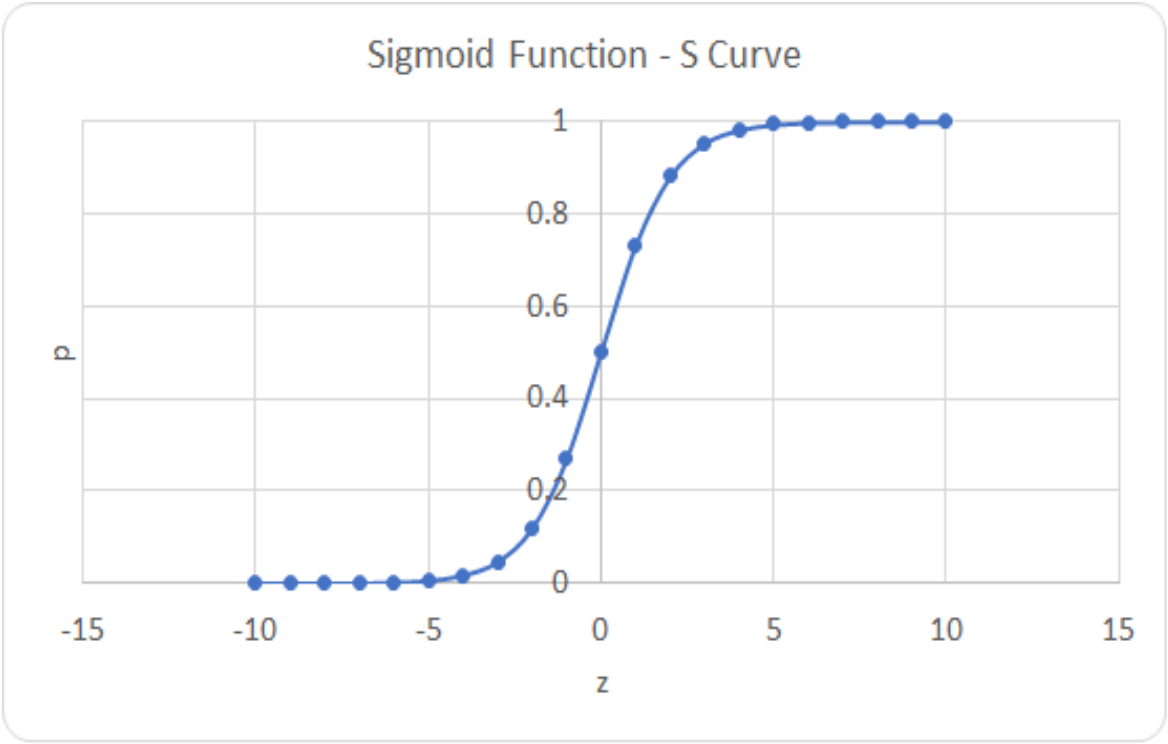
- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

4.2 Logistic Regression: Logistic regression is another powerful supervised ML algorithm used for binary

classification problems (when target is categorical). The best way to think about logistic regression is that it is a linear regression but for classification problems. Logistic regression essentially uses a logistic function defined below to model a binary output variable (Tolles & Meurer, 2016).

The primary difference between linear regression and logistic regression is that logistic regression's range is bounded between 0 and 1. In addition, as opposed to linear regression, logistic regression does not require a linear relationship between inputs and output variables. This is due to applying a nonlinear log transformation to the odds ratio.

$$Logistic\ Function = \frac{1}{1+e^{-x}}$$



Logistic regression is another fundamental method initially formulated by David Cox in 195832 that builds a logistic model (also known as the logit model). Its most significant advantage is that it can be used both for classification and class probability estimation, because it is tied with logistic data distribution. It takes a linear combination of features and applies to them a nonlinear sigmoidal function. In the basic version of logistic regression, the output variable is binary, however, it can be extended into multiple classes (then it is called multinomial logistic regression). The binary logistic model classifies specimen into two classes, whereas the multinomial logistic model extends this to an arbitrary number of classes without ordering them.

The mathematics of logistic regression rely on the concept of the “odds” of the event, which is a probability of an event occurring divided by the probability of an event not occurring. Just as in linear regression, logistic regression has weights associated with dimensions of input data. In contrary to linear regression, the relationship between the weights and the output of the model (the “odds”) is exponential, not linear.

SOFTWARE USED

5.1 PYTHON: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print

statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

5.2 MICROSOFT OFFICE

5.2.1 MICROSOFT WORD: Microsoft Word is a word processing software developed by Microsoft. It was first released on October 25, 1983, under the name Multi-Tool Word for Xenix systems. Subsequent versions were later written for several other platforms including: IBM PCs running DOS (1983), Apple Macintosh running the Classic Mac OS (1985), AT&T UNIX PC (1985), Atari ST (1988), OS/2 (1989), Microsoft Windows (1989), SCO Unix (1990) and macOS (2001).

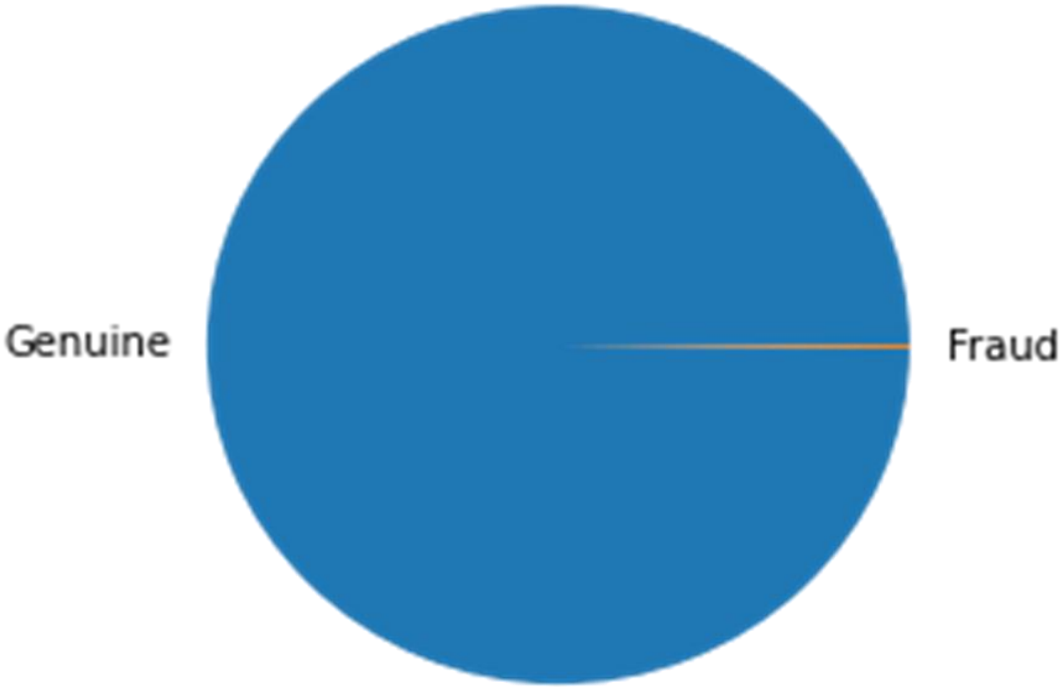
Commercial versions of Word are licensed as a standalone product or as a component of Microsoft Office suite of software, which can be purchased either with a perpetual license or as part of a Microsoft 365 subscription. Word can also be acquired by purchasing Windows RT or the discontinued Microsoft Works suite.

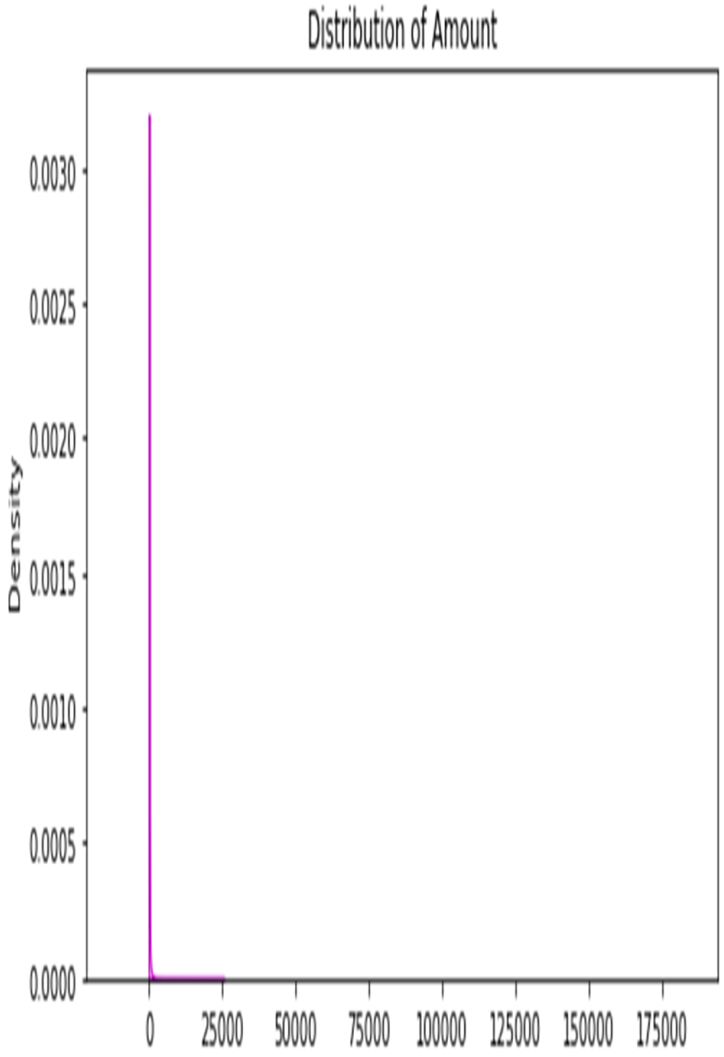
5.2.2MICROSOFT EXCEL: Microsoft Excel has the basic features of all spreadsheets, using a grid of cells arranged in numbered rows and letter-named columns to organize data manipulations like arithmetic operations. It has a battery of supplied functions to answer statistical, engineering, and financial needs. In addition, it can display data as line graphs, histograms and charts, and with a very limited three-dimensional graphical display. It allows sectioning of data to view its dependencies on various factors for different

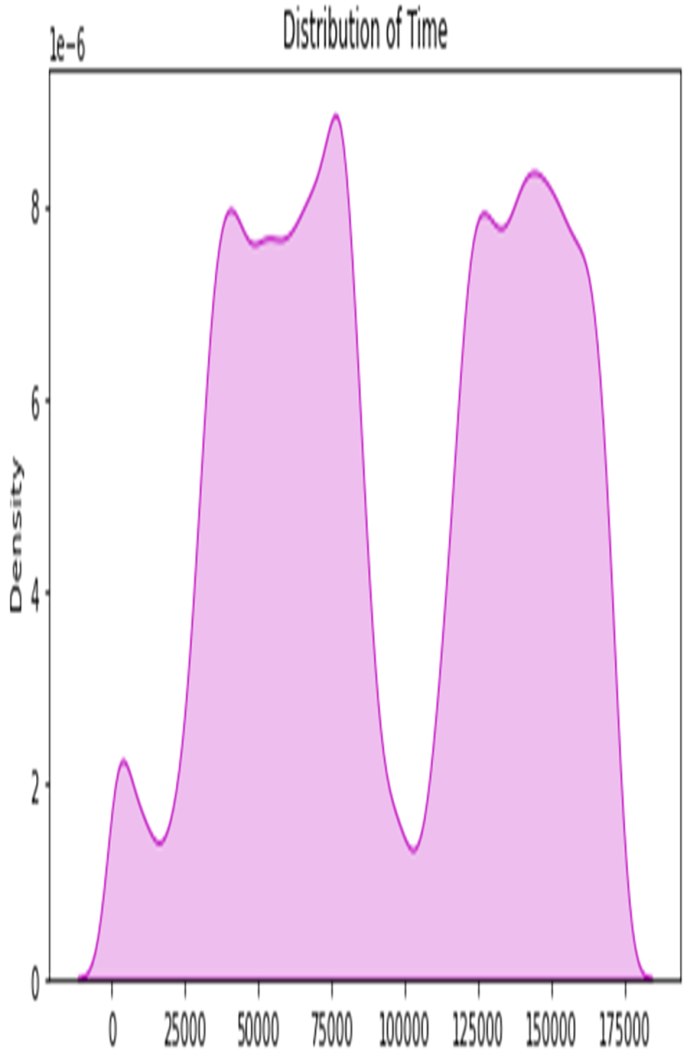
perspectives (using pivot tables and the scenario manager). A PivotTable is a tool for data analysis.

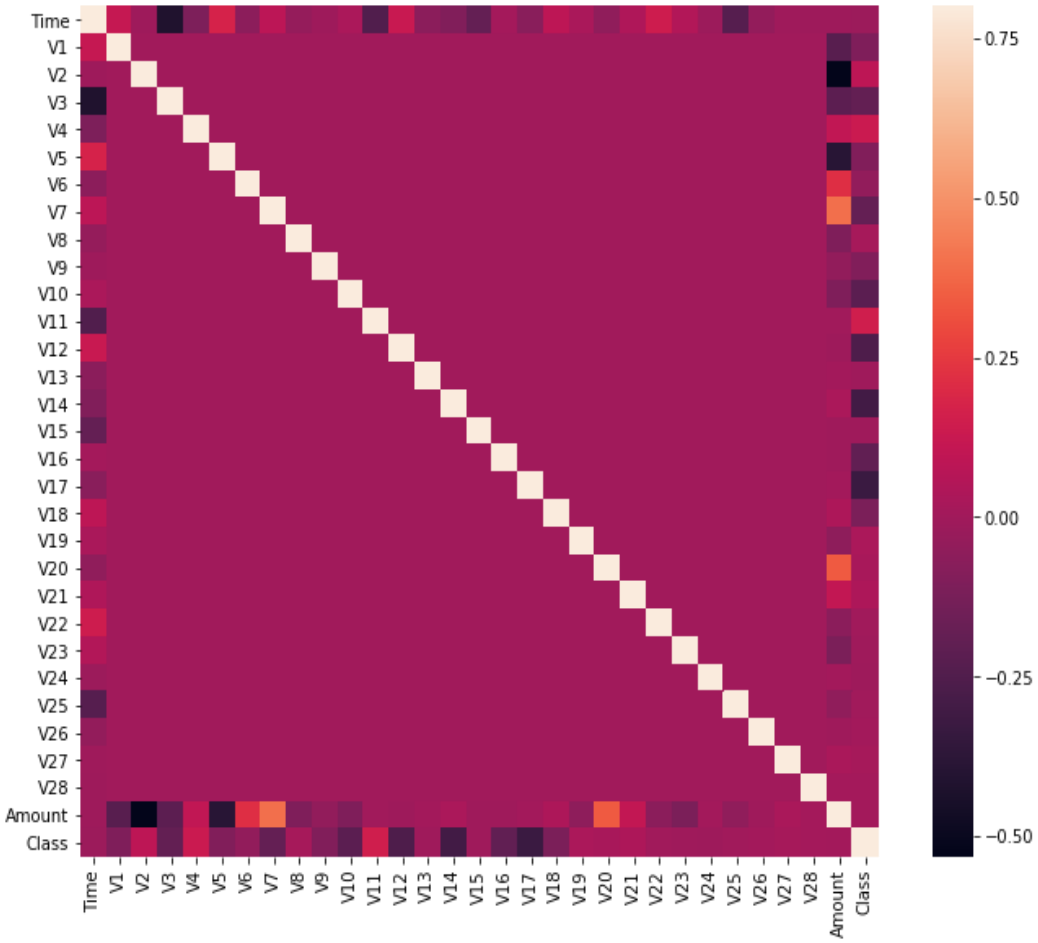
RESULTS

There is an imbalance in the data, with only 0.17% of the total cases being fraudulent.



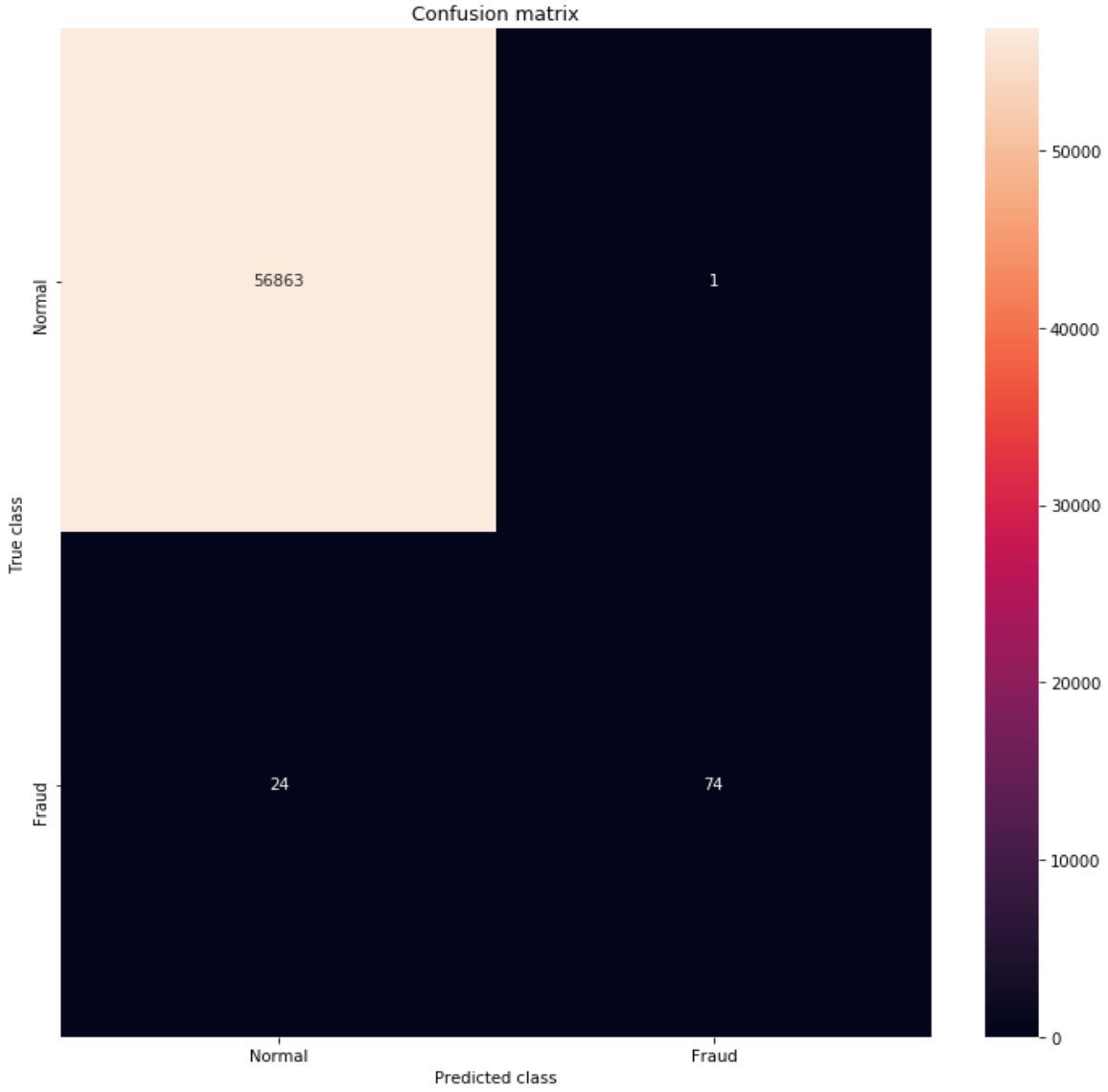

Distribution of Amount
Density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
25000
50000
75000
100000
125000
150000
175000


Distribution of Time
1e−6
Density
8
6
4
2
0
0
25000
50000
75000
100000
125000
150000
175000



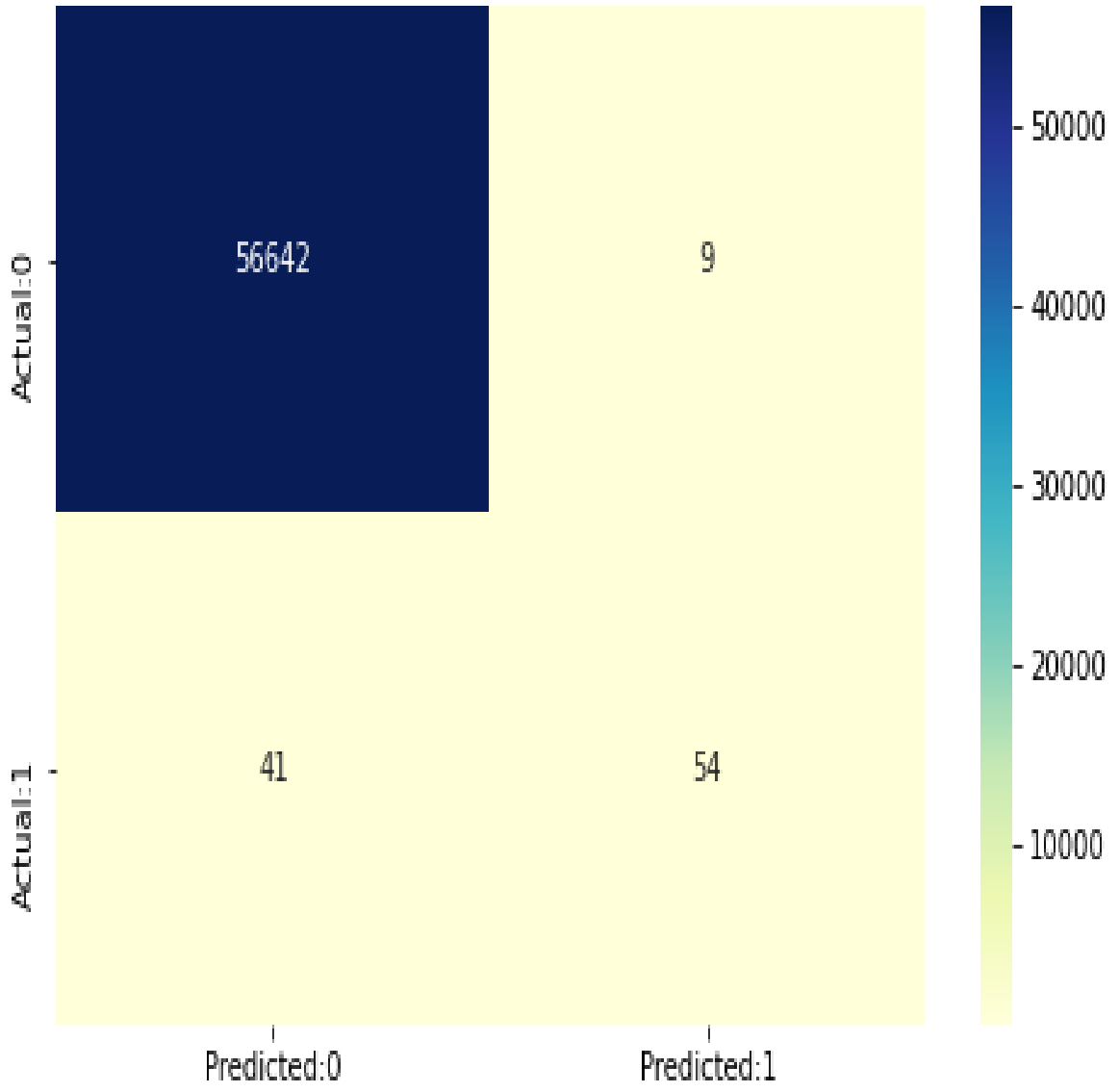
Variance Covariance Matrix For Random Forest Classifier

In the above variance covariance matrix we can clearly see that most of the features do not correlate to other features but there are some features that either has a positive or a negative correlation with each other. For example, V2 and V5 are highly negatively correlated with the feature called Amount. We also see some correlation with V20 and Amount. This gives us a deeper understanding of the Data available to us.



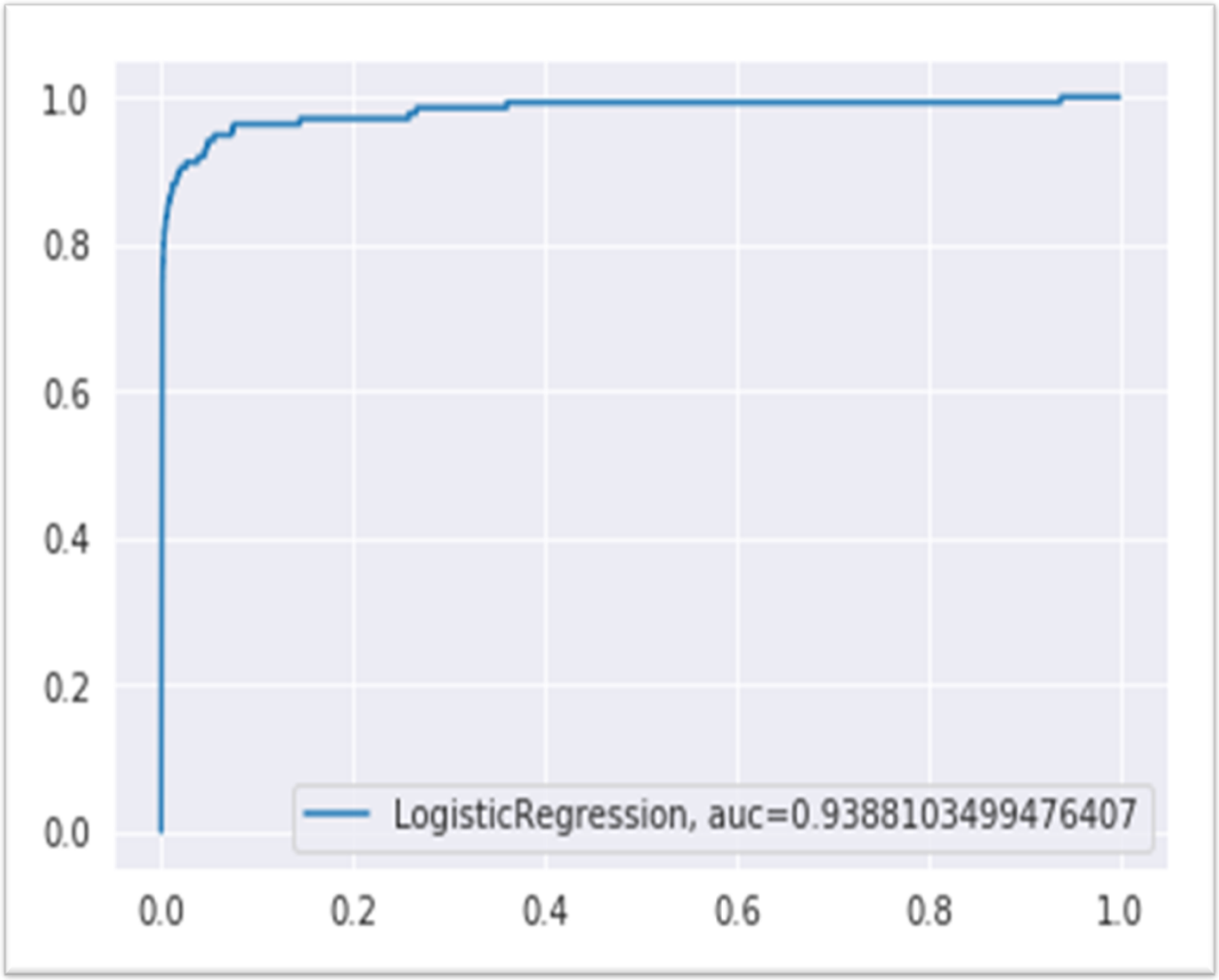
Confusion Matrix for Random Forest Classifier

Here, 56863 correct predictions, 74 wrong predictions and 24 type 2 error(False Positive) ,1 type 1 error(False Negative).



Confusion Matrix for Logistic Regression

Here, 56642 correct predictions, 54 wrong predictions and 41 type 2 error(False Positive) ,9 type 1 error(False Negative).


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
LogisticRegression, auc=0.9388103499476407

RESULT :

- Fraudulent Cases: 492
- Valid Transactions: 284315
- Proportion of Fraudulent Cases : 0.0017

For the Random Forest Classifier:

- The accuracy is 0.9995611109160493
- The precision is 0.9866666666666667
- The recall is 0.7551020408163265
- The F1-Score is 0.8554913294797689
- The Matthews correlation coefficient is0.8629589216367891

For the Logistic Regression :

- Accuracy of the Logistic Regression model is 0.9991148644726914
- F1 score of the Logistic Regression model is 0.6934673366834171
- Area under the ROC curve is 0.93881034994

7. Appendix

```
## import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
## Load the dataset from the csv file using pandas
## best way is to mount the drive on colab and
## copy the path for the csv file
data = pd.read_csv("C:/Users/Riya Vishwakarma/Downloads/creditcard_csv.zip")
## Grab a peek at the data
data.head()
## Grab a peek at the data
data.head()
##print the shape of the data
##data=data.sample(frac=0.1,random_state=48)
print(data.shape)
print(data.describe)
## Determine number of fraud cases in dataset
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierFraction = len(fraud)/float(len(valid))
```

```
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
print("Amount details of the fraudulent transaction")
fraud.Amount.describe()
print("details of valid transaction")
valid.Amount.describe()
# To see how small are the number of Fraud transactions
data_p = dataset.copy()
data_p[" "] = np.where(data_p["Class"] == 1 ,  "Fraud", "Genuine")

# plot a pie chart
data_p[" "].value_counts().plot(kind="pie")
# plot the named features
f, axes = plt.subplots(1, 2, figsize=(18,4), sharex = True)

amount_value = dataset['Amount'].values # values
time_value = dataset['Time'].values # values

sns.distplot(amount_value, hist=False, color="m",
kde_kws={"shade": True}, ax=axes[0]).set_title('Distribution of
Amount')
sns.distplot(time_value, hist=False, color="m", kde_kws={"shade":
True}, ax=axes[1]).set_title('Distribution of Time')

plt.show()
```

```
## Variance Covariance  matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
## dividing the X and the Y from the dataset
X = data.drop(['Class'], axis = 1)
Y = data["Class"]
print(X.shape)
print(Y.shape)
## getting just the values for the sake of processing
## (its a numpy array with no columns)
xData = X.values
yData = Y.values
## Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
## Split the data into training and testing sets
xTrain, xTest, yTrain, yTest = train_test_split(
        xData, yData, test_size = 0.2, random_state = 42)
## Building the Random Forest Classifier (RANDOM FOREST)
from sklearn.ensemble import RandomForestClassifier
## random forest model creation
rfc = RandomForestClassifier()
rfc.fit(XTrain,yTrain)
```

```
## predictions
yPred = rfc.predict(xTest)
## Evaluating the classifier
## printing every score of the classifier
## scoring in anything
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
 n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")
 acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))
 prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))
 rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))
 f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))
 MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
## printing the confusion matrix
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
```

```
plt.figure(figsize =(12, 12))
sns.heatmap(conf_matrix, xticklabels = LABELS,
        yticklabels = LABELS, annot = True, fmt ="d");
plt.title("Confusion matrix")
plt.show()
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_yhat = lr.predict(X_test)
print('Accuracy score of the Logistic Regression model is
{}'.format(accuracy_score(y_test, lr_yhat)))
print('F1 score of the Logistic Regression model is
{}'.format(f1_score(y_test, lr_yhat)))
clf = LogisticRegression(C=1, penalty='l2')
clf.fit(X_undersampled_train, Y_undersampled_train)
y_pred = clf.predict(X_test)

y_pred_probability = clf.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_probability)
auc = metrics.roc_auc_score(y_test, pred)
plt.plot(fpr,tpr,label="LogisticRegression, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

REFRENCES

- A Survey Paper On Credit Card Fraud Detection Techniques, International Journal of Scientific & Technology Research.
- Credit Card Fraud Detection Using Machine Learning Algorithms,Journal of Reasearch in Humanities and Social Science,Volume 8
- Credit Card Fraud Detection Using Machine Learning and Data Science,International Journal of Engineering and Technical Reasearch 08(09),September 2019
- C. Rajan (2011), “Data Mining Algorithms”, Published by Narosa Publishing House Pvt. Ltd.

- Website Link: https://www.Wikipedia.com