PREDICTIVE MODEL FOR CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS

BY

J. SHANMUGAPRIYA

20PCA017

Project Report Submitted

In partial fulfilment of the requirements for the Degree of

Master of Computer Applications

DEPARTMENT OF COMPUTER SCIENCE
AVINASHILINGAM INSTITUTE FOR HOME SCIENCE
ANDHIGHER EDUCATION FOR WOMEN
COIMBATORE – 641043

MAY – 2022

PREDICTIVE MODEL FOR CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS

BY

J. SHANMUGAPRIYA

20PCA017

Project Report Submitted

In partial fulfilment of the requirements for the Degree of

Master of Computer Applications

DEPARTMENT OF COMPUTER SCIENCE

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

COIMBATORE – 641043

MAY - 2022

Signature of the Head of the Department **Signature of Supervisor**

Viva-voce Examination held on _____ **Signature of Examiner**

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I would like to express our sincere thanks to God Almighty for his constant and grace that he has showed upon me, which kept me in good health, and sound mind without which my project would not have reached a successful end.

I would like to express my deep sense of reverential gratitude and sincere thanks to Prof.**Dr**.**S.P. Thyagarajan,** Chancellor, Avinashilingam Institute of Home Science and Higher Education for Women, Coimbatore, for the opportunity given to me for undertaking this study and for providing all the needed facilities during the course of my study.

I owe my great deal of gratitude to **Dr.V. Bharathi Harishankar**, Ph.D., FRSA Vice Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for extending all resources that facilitated the conduct of the present study.

I express my humble gratitude to **Dr. A. Kowsalya,** Registrar, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities necessary for the study.

I would express my boundless thanks to **Dr. G. Padmavathi**, M.Sc., M.Phil., Ph.D., and Dean, School of Physical Sciences & Computational Sciences, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for granting the facility required.

I wish to place on record my deep sense of gratitude to **Dr. Vasantha Kalyani David**, M.Sc., M.Phil. (Maths)., M.Phil. (CS)., Ph.D., Professor and Head, Department of Computer Science for support and encouragement to complete the project.

I am grateful to the project coordinator **Dr. (Mrs.). V Srividhya**, M.Sc., M.Phil., Ph.D., Assistant Professor, Department of Computer Science, who was instrumental in granting me the facilities required for doing project.

I owe great deal of gratitude to my esteemed guide **Dr.N. Valliammal, M.Sc., M.Phil. Ph.D** Assistant Professor(SG), Department of Computer Science, for imparting the tremendous assistance and well-timed support for triumph of my project.

Finally, yet importantly, I would like to thank my **parents, family members and friends** for their kind inspiration, support, encouragement, blessings and prayers, which were instrumental in the successful completion of the project.

I have great pleasure in expressing my deep sense of gratitude to all other teaching and nonteaching staff members of the Department of Computer Science, who stood behind the screen for the completion of the project.

I would extend my hearty thanks to one and all that helped me directly or indirectly for the successful completion of my project.

CONTENTS

CONTENTS

ABSTRACT

1.ABSTRACT

In this project, a methodology has been proposed that performs predictive model for credit card fraud detection using machine learning algorithms collected from anonymized credit card fraud dataset. main challenge for today's CCFD system is to improve fraud detection accuracy with the growing number of transactions per user per second. Predictive modelling is used to predict an unknown event that may occur in the future. This process is to create, test, and validate the model. This project investigates and checks the performance of the Random Forest Classifier, Decision tree on highly skewed credit card fraud data. The dataset of credit card transactions is sourced from European cardholders and contains 2,84,807 transactions. Some techniques are applied to the raw and pre-processed data. The performance of the techniques is evaluated based on accuracy, sensitivity, and precision. The proposed system indicates the optimal accuracy of Random Forest and Decision tree. The prediction results are further produced with effective visualization.

INTRODUCTION

2.INTRODUCTION

The project is entitled as "**Predictive Model For Credit Card Fraud Detection Using Machine Learning Algorithms"** is developed using python Programming environment. Financial fraud is a growing concern with far-reaching consequences in the government, corporate organizations, the finance industry, and in today's world, high dependency on internet technology has enjoyed increased credit card transactions, but credit card fraud has also accelerated in online and offline transactions. As credit card transactions become a widespread mode of payment, focus has been given to recent computational methodologies to handle the credit card fraud problem. There are numerous fraud detection solutions and software available to help businesses such as credit card, retail, e-commerce, insurance, and industries prevent fraud. Data mining techniques are one notable and popular method used in solving credit fraud detection problems. It is impossible to be absolutely certain about the true intention and rightfulness of an application or transaction. Seeking out possible evidence of fraud from the available data using mathematical algorithms is the best effective option. Credit card fraud detection is the process of identifying those transactions that are fraudulent into two classes: legit and fraud class transactions. Several techniques are designed and implemented to solve credit card fraud detection problems, such as genetic algorithms, artificial neural networks, frequent item set mining, machine learning algorithms, migrating birds optimization algorithms, which can also perform comparative analysis of random forest, and decision tree. Credit card fraud detection is very popular but also a difficult problem to solve. Firstly, due to the issue of having only a limited amount of data, credit cards make it challenging to match a pattern for a dataset. Secondly, there can be many entries in a dataset with fraud transactions that fit the pattern of legitimate behavior. Also, the problem has many constraints. Firstly, sensitive data sets are not easily accessible to the public and the results of research are often hidden and censored, making the results inaccessible. Due to this, it is sometimes challenging to benchmark certain models. Second, method improvement is made more difficult by the fact that security concerns limit the exchange of ideas and methods in fraud detection, particularly in credit card fraud detection.Lastly, the data sets are continuously evolving and changing, which makes the profiles of normal and fraudulent behaviours different from the legit transactions in the present, which may have been fraud in the past or vice versa. This project

evaluates two data mining approaches: decision trees, and random forests, and then a collative comparison is made to evaluate which model performed best.

Credit card transaction datasets are rarely available, highly imbalanced, and skewed. Optimal feature (variables) selection for the models, suitable metric is most important part of data mining to evaluate performance of techniques on skewed credit card fraud data. Fraudulent behavior profile is dynamic, that is fraudulent transactions tend to look like legitimate ones, also credit card fraud detection performance is greatly affected by type of sampling approach used, In the end, conclusions about results of classifier evaluative testing are made and collated. From the experiments the result that has been concluded is that Random Forest Classifier has a accuracy of 99.9% while decision tree shows accuracy of 99.6%.

SYSTEM CONFIGURATION

3. SYSTEM CONFIGURATION

3.1 SOFTWARE SPECIFICATION

Front end and back end: Python:

Python is a computer programming language often used to **build websites and software, automate tasks, and conduct data analysis**. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm Red Monk found that it was the second-most popular programming language among developers in 2021. Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

Python is a **high-level programming language** that has English-like syntax. This makes it easier to read and understand the code.

Advantages of Python

1. Easy to Read, Learn and Write

Python is a **high-level programming language** that has English-like syntax. This makes it easier to read and understand the code.

Python is really easy to **pick up** and **learn**, that is why a lot of people recommend Python to beginners. You need less lines of code to perform the same task as compared to other major languages like **C/C++** and **Java**.

2. Improved Productivity

Python is a very **productive language**. Due to the simplicity of Python, developers can focus on solving the problem. They don't need to spend too much time in understanding the **syntax** or behavior of the programming language. You write less code and get more things done.

3. Interpreted Language

Python is an interpreted language which means that Python directly **executes the code** line by line. In case of any error, it stops further execution and reports back the error which has occurred.

Python shows only one error even if the program has multiple errors. This makes **debugging** easier.

4. Dynamically Typed

Python doesn't know the type of variable until we run the code. It automatically assigns the data type during **execution**. The programmer doesn't need to worry about declaring variables and their data types.

5. Free and Open-Source

Python comes under the **OSI approved** open-source license. This makes it **free** to **use** and **distribute**. You can download the source code, modify it and even distribute your version of Python. This is useful for organizations that want to modify some specific behavior and use their version for development.

6. Vast Libraries Support

The standard library of Python is huge, you can find almost all the functions needed for your task. So, you don't have to depend on external libraries.

But even if you do, a **Python package manager (pip)** makes things easier to import other great packages from the **Python package index (PyPi)**. It consists of over 200,000 packages.

7. Portability

In many languages like C/C++, you need to change your **code** to run the program on different platforms. That is not the same with Python. You only write once and run it anywhere.

Disadvantages of Python

- Speed
- Memory Consumption
- Database Access
- Runtime Errors

Jupyter Notebook

The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others.it is an open source web application that you can use **to create and share documents that contain live code, equations, visualizations, and text**. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebook is an **open-source, web-based interactive environment,** which allows you to create and share documents that contain **live code, mathematical equations, graphics, maps, plots, visualizations,** and **narrative text**. It integrates with many programming languages like **Python, PHP, R, C#,** etc.

Pandas - Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Numpy - NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy provides a multi-dimensional array.

Seaborn - Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps us explore and understand your data.

Matplotlib - Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter.

Pros

- All in one place
- Easy to convert
- Easy to share
- Language independent
- Interactive code

Cons

- It is very hard to test long asynchronous tasks.
- Less Security
- It runs cell out of order
- In Jupyter notebook, there is no IDE integration, no linting, and no code-style correction.**Operating system: Windows 10**

3.2. HARDWARE SPECIFICATION

- Hard Disk : 465.76 GB
- Monitor : 21’ Color display
- RAM : 4.00 GB
- Processor : AMD PRO A4-3350B APU with Radeon R4 Graphics
- Processor speed : 2.00 GHz
- System Type : 64-bit Operating System

4. SYSTEM DEVELOPMENT

4.1 Dataset description

The credit card fraud detection project have used a dataset which consist of transactions made by European cardholders during month of July 2013 through various credit cards. Data has been collected and analyzed, the dataset consists of transactions occurred in span of two days, where have 492 fraudulent transactions out of 284,807 total transactions. The dataset is highly unbalanced where positive class (Frauds) accounts for only 0.172% of all transactions.

4.2 METHODS AND ALGORITHMS USED

Decision tree

Decision tree is a supervised learning algorithm. This technique is mostly used in classification problems. It performs effortlessly with continuous and categorical attributes. the decision tree algorithms also used in identifying the importance of the feature metrics. In decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. Which used in handpicking the features. Once the important features identified then the model trains with the training data to come up with a set of rules. These rules used in predicting future cases or for the test dataset. Here, the dataset is splitted into training dataset as 70% and test dataset as 30%.

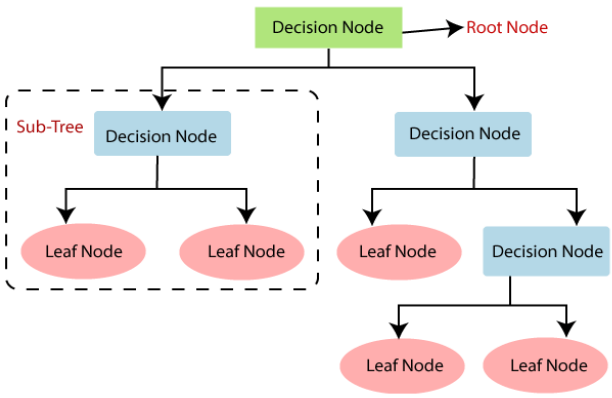


Figure 1. Decision tree model

Random forest

Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of overfitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because 8 each tree is trained independently of the others. The Random Forest algorithm has been found to provides a good estimate of the generalization error and to be resistant to overfitting. The random forest algorithm falls under the ensemble learning algorithm category. In the random forest algorithm, we build N decision tree models. Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.The below image: Figure 2. Random forest model. All the models predict the target value. Using the majority voting approach the final target value will be predicted for building the individual decision tree, the random forest algorithm randomly creates the sample dataset. These sample datasets are called as the bootstrap samples. Here, the dataset is splitted into training dataset as 70% and test dataset as 30%.

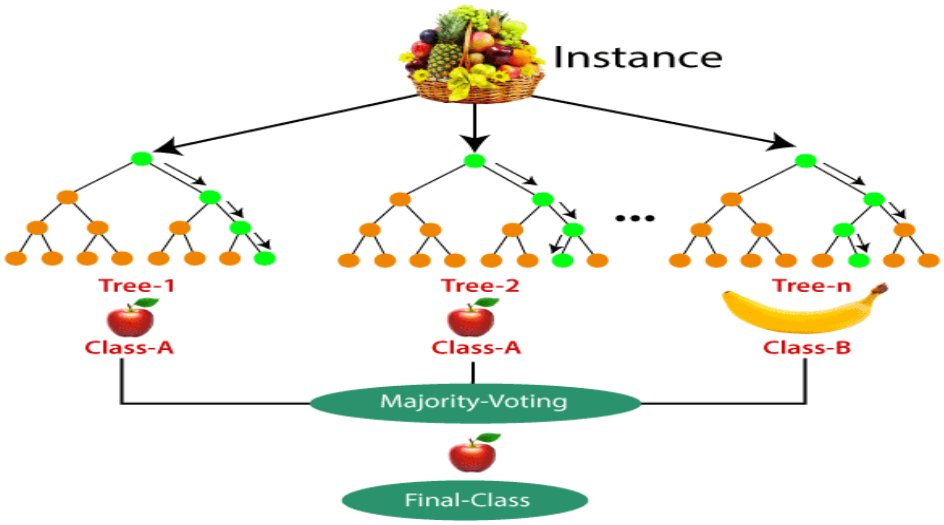


Figure 2. Random forest model

ROC-AUC Score

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. The area under the ROC curve (AUC) results were considered excellent for AUC values between 0.9-1, good for AUC values between 0.8-0.9, fair for AUC values between 0.7-0.8, poor for AUC values between 0.6-0.7 and failed for AUC values between 0.5-0.6.

	ROC-AUC SCORE
Decision Tree	0.94
Random Forest	0.87

Table 1. ROC-AUC SCORE of Random Forest and Decision Tree

The above table shows the ROC-AUC Score of Random Forest and Decision Tree model.

4.3. MODULES

- DATA PRE-PROCESSING.
- DATA EXPLORATION.
- REVIEWS CLASSIFIED AS POSITIVE OR NEGATIVE.
- VISUALIZATION.

4.3. MODULE DESCRIPTION

4.3.1 DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

DATA EXPLORATION: Data exploration refers to the initial step in data analysis in which data analysts use data visualization and statistical techniques to describe dataset characterizations. Exploring data sets and developing deep understanding about the data is one of the most important skills every data scientist should possess. People estimate that the time spent on these activities can go as high as 80% of the project time in some cases. NumPy, Matplotlib, Seaborn, and Pandas is used to perform data exploration. These are powerful libraries to perform data exploration in Python. The idea is to create a ready reference for some of the regular operations required frequently.

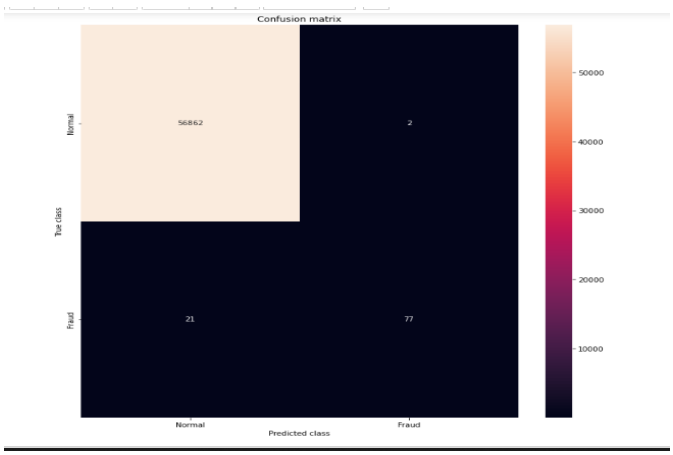
4.3.2. CORRELATION MATRIX WITH HEAT MAP

A correlation heatmap is **a graphical representation of a correlation matrix representing the correlation between different variables**. The value of correlation can take any value from -1 to 1. Correlation between two random variables or bivariate data does not necessarily imply a causal relationship. there is no notable correlation between features V1-V28. As there are certain correlations between some of the features and Time (inverse correlation with V3) and Amount (direct correlation with V7 and V20, inverse correlation with V1 and V5). Next, plot the correlated and inverse correlated values on the same graph using matplotlib Python Graph packages and visualize direct correlated values: {V20; Amount} and {V7; Amount}.

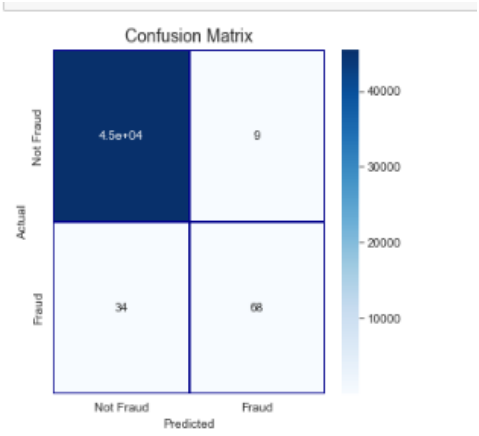
4.3.3. EVALUATION METRICS

Confusion Matrix

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives a holistic view of how well our classification model is performing and what kinds of errors it is making. For a binary classification problem, 2 x 2 matrix with 4 values were used for binary class problem.



4.3.3.1 Random forest



4.3.3.2 Decision tree

Accuracy

Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions. where this model have got high accuracy in

1.True Positive (TP):

The number of positive labels correctly predicted by trained models. This means the number of Class-1 samples correctly predicted as Class-1.

2.True Negative (TN):

The number of negative labels correctly predicted by trained models. This means the number of Class-0 samples correctly predicted as Class-0.

3.False Positive (FP):

The number of positive labels incorrectly predicted by trained models. This means the number of Class-1 samples incorrectly predicted as Class-0.

4.False Negative (FN):

The number of negative labels incorrectly predicted by trained models. This means the number of Class-0 samples incorrectly predicted as Class-1.

Precision

The precision metric shows the accuracy of the positive class. It measures how likely the prediction of the positive class is correct.

Precision = TP / TP + FP In this model we have got highest precision value as 0.94

Recall

Recall computes the ratio of positive classes correctly detected. This metric gives how good the model is to recognize a positive class.

Recall = TP/TP+FN

F1Score

F1 score is a weighted average score of the true positive (recall) and precision.

F1 = 2 x Precision * Recall / Precision + Recall

	Decision Tree	Random Forest
Accuracy	99.6%	99.9%
Precision	68.6%	97.4%
Recall	75.7%	78.5%
F1 score	72%	87.4%

Table 2 .Decision Tree and Random Forest Model

The above table shows the Accuracy, Precision, Recall, and F1 Score of Decision Tree and Random Forest model

4.2.2. IMPORTING DATASET AND ESSENTIAL LIBRARIES AND RETRIVING DATA.

The datasets contain transactions made by credit cards in **September 2013** by European cardholders. This dataset presents transactions that occurred in the last two days, where the dataset have **108 frauds out of 2,84,807 transactions.** The response variable is Feature Class, which has a value of 1 in the case of fraud **0 otherwise.

DATASET LINK - Major Project\creditcard1.csv

Pandas - Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Numpy - NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy provides a multi-dimensional array.

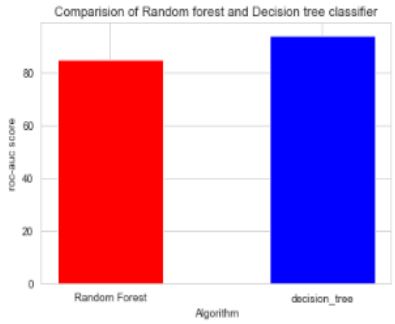
Seaborn - Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps us explore and understand your data.

Matplotlib - Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter.

5.RESULT

VISUALIZATION

```
In [250]: from sklearn.metrics import roc_curve,roc_auc_score

In [257]: import matplotlib.pyplot as plt
          x= ['Random Forest', 'decision_tree']
          y= [85,94]
          plt.bar(x,y,color=['red','blue'],width=0.5)
          plt.title('Comparision of Random forest and Decision tree classifier')
          plt.xlabel('Algorithm')
          plt.ylabel('roc-auc score')
          plt.show()
```




```
In [ ]:
```

Figure 4.Bar plot of Random forest and Decision tree

For the two models, only used the train and test set. Random Forest Classifier, provided AUC score of 0.87 by predicting target for the test set. Decision tree, provided AUC score of 0.94 by predicting target for the test set.

	ROC-AUC SCORE
Decision Tree	0.94
Random Forest	0.87

Table 3. ROC-AUC SCORE of Random Forest and Decision Tree

CONCLUSION

6.CONCLUSION AND FUTURE SCOPE

This project Investigated the data, checked for data unbalancing, visualized, and understood the relationship between different features. This project has also explained in detail, how machine learning can be applied to get better results in fraud detection along with the algorithm, pseudocode, explanation its implementation and experimentation results. used two predictive models to perform validation by splitting dataset into 2 parts, a train set, a validation set and a test set. For the two models, only used the train and test set. Random Forest Classifier, provided AUC score of 0.87 by predicting target for the test set. Decision tree, provided AUC score of 0.94 by predicting target for the test set.

FUTURE SCOPE

- This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code.
- The precision of the algorithms increases when the size of dataset is increased. Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of false positives.

BIBLIOGRAPHY

6.BIBLIOGRAPHY

WEB RESOURSES

Random Forest Classifier,
http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForest

Classifier. html

RESEARCH JOURNALS

[1] Y. Liu, Z. Qin, Z. Shi and J. Chen (2004), Rule discovery with particle swarm optimization, Advanced Workshop on Content Computing, Vol 3309.

[2] J. Ji, N. Zhang, C. Liu and N. Zhong (2006), An ant colony optimization algorithm for learning classification rules, in Proc. IEEE/WIC(2006), pp 1034–1037, ISBN:0-7695-2747-7.

[3] X. Zhao, J. Zeng, Y. Gao and Y. Yang (2006), Particle swarm algorithm for classification rules generation, Proc. of the Intelligent Systems Design and Applications, IEEE (2006), pp 957– 962.

[4] N. Holden and A. A. Frietas (2008), A Hybrid PSO/ACO algorithm for Discovering classification Rules in Data Mining, Proc. Genetic and evolution computation conf. (2008), pp 2745–2750 , Article ID 316145.

[5] H. Su, Y. Yang and L. Zha (2010), Classification rule discovery with DE/QDE algorithm, Expert Systems with Applications, Vol.37, issn*:* 0957-4174.

[6] Aamodt, E. Plaza (1994), Case-based reasoning: foundational issues, methodological variations, and system approaches, Artificial Intelligence Communications, IOS Press, Vol. 7: 1, pp. 39-59, DOI: 10.3233/AIC-1994-7104.

[7] M-J. Kim and I. Han (2003), Decision rules from qualitative bankruptcy data using genetic algorithms, Expert Systems with Applications, Vol 25., pp. 35, DOI:10.1016/S0957-4174(03)00102-7.

[8] T. Sousa, A. Neves, and A. Silva (2003), A particle swarm data miner, 11th Portuguese Conference AI, Workshop on Artificial Life and Evolutionary Algorithms, 43–53.

[9] K-S. Shin, Y-J. Lee, A genetic algorithm application in bankruptcy prediction modeling, Expert Systems with Applications (2002), Vol 23 (3), pp 321-328. DOI:10.1016/S0957-4174(02)00051-9.

[10] R. S. Parpinelli, H. S. Lopes and A. A. Frietas (2002), Data Mining with an Ant Colony Optimization Algorithm, IEEE (2002), Vol 6, pp 321 - 332 ISSN: 1089-778X DOI: 10.1109/TEVC.2002.802452.

[11] Lu Q, Ju C (2011) "Research on credit card fraud detection model based on class weighted support vector machine", Journal Convergence Information Technology l 6, Vol – 6,pp 62–68 DOI:10.4156/JCIT.

[12] S. Bhattacharyya, S. Jha, K. Tharakunnel, J. C. Westland (2011), "Data mining for credit card fraud: A comparative study", Decision Support Systems, vol. 50, no. 3, pp. 602-613, DOI:10.1016/j.dss.2010.08.008.

[13] Y. Sahin, E. Duman (2011), "Detecting credit card fraud by ANN and logistic regression", Innovations in Intelligent Systems and Applications (INISTA) 2011 International Symposium, pp. 315-319, ISBN:978-1-61284-919-5 DOI: 10.1109/INISTA.2011.5946108 .

[14] Selvani Deepthi Kavila,LAKSHMI S.V.S.S.,RAJESH B (2004)," Automated Essay Scoring using Feature Extraction Method " IJCER ,volume 7,issue 4(L), Page No. 12161-12165 ,ISSN (O) 2321-2004, ISSN (P) 2321-5526 DOI: 10.17148/IJIREEICE.2021.91209.

APPENDIX

7.APPENDIX

DATA FLOW DIAGRAM

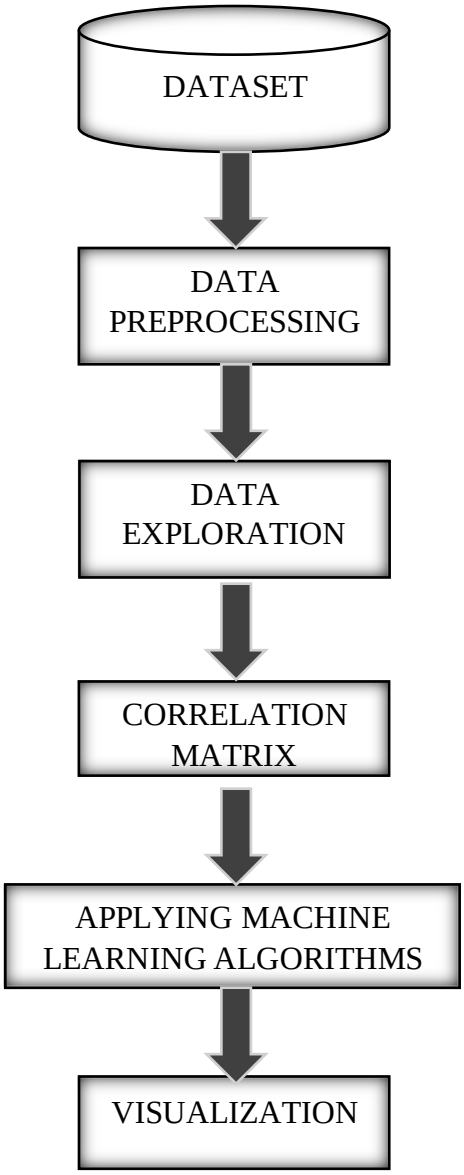


Figure 5.Data flow diagram

CREDIT CARD FRAUD DATASET

In order to system, find the creditcard1.csv file that have used in predictive model for credit card fraud detection system. the dataset used here which consist of transactions made by European cardholders during month of september 2013 through various credit cards. The dataset consists of transactions occurred in span of two days, where we have 492 fraudulent transactions out of 2,84,807 total transactions. The link is given below

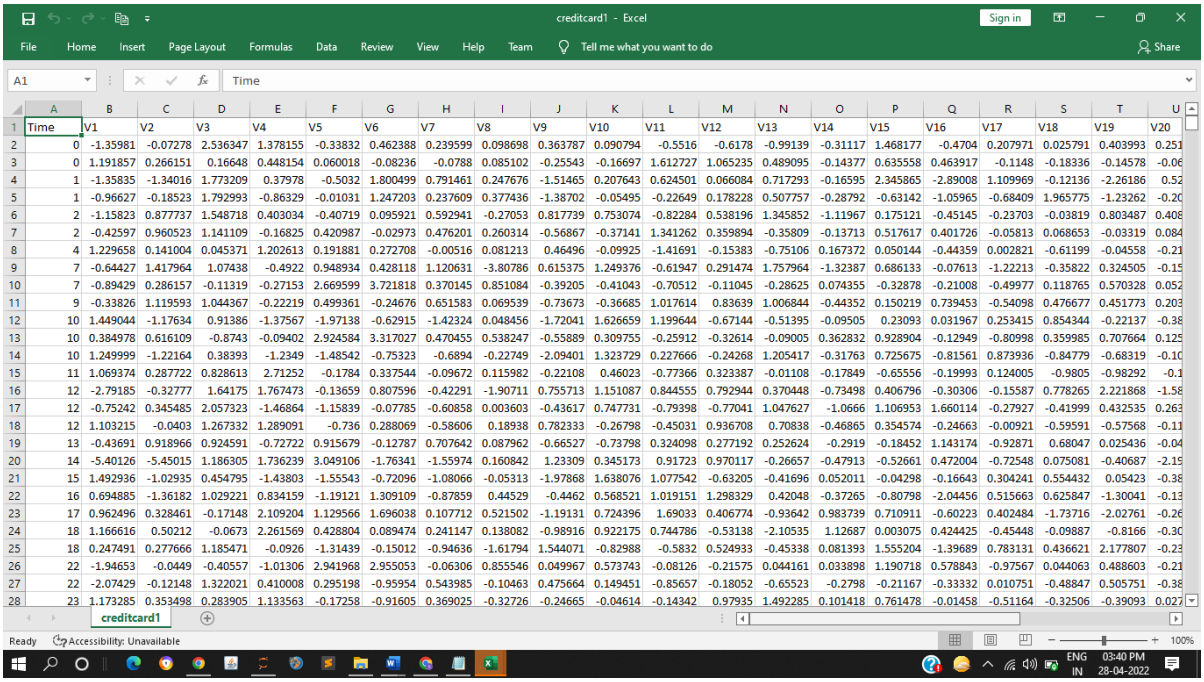
Major Project\creditcard1.csv

The information about credit card fraud transactions can be collected from the above link.

DATASET IN EXCEL

CREDIT CARD FRAUD DATASET

In this creditcard1.csv file there are **2,84,807 transactions** in the last two days**.**



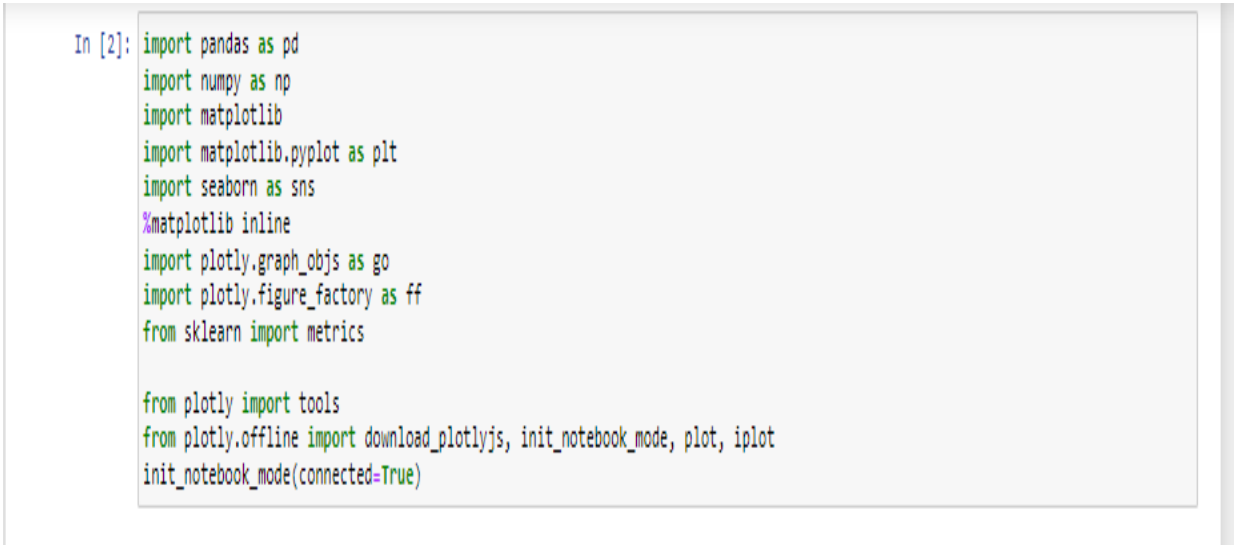
Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.403993	0.251
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0.14578	-0.06
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	-0.12136	-2.26186	0.52
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	1.965775	-1.23262	-0.20
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	-0.03819	0.803487	0.408
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	0.068653	-0.03319	0.084
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	-0.61199	-0.04558	-0.21
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	-0.35822	0.324505	-0.15
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	0.118765	0.570328	0.052
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	0.476677	0.451773	0.203
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	0.854344	-0.22137	-0.38
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	0.359985	0.707664	0.125
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	-0.84779	-0.68319	-0.10
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	-0.9805	-0.98292	-0.1
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	0.778265	2.221868	-1.58
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	-0.41999	0.432535	0.263
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	-0.59591	-0.57568	-0.11
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	0.68047	0.025436	-0.04
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	0.075081	-0.40687	-2.19
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	0.554432	0.05423	-0.38
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	0.625847	-1.30041	-0.13
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	-1.73716	-2.02761	-0.26
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	-0.09887	-0.8166	-0.30
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	0.436621	2.177807	-0.23
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	0.044063	0.488603	-0.21
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	-0.48847	0.505751	-0.38
23	1.173285	0.353498	0.283905	1.133563	-0.17258	-0.91605	0.369025	-0.32726	-0.24665	-0.04614	-0.14342	0.97935	1.492285	0.101418	0.761478	-0.01458	-0.51164	-0.32506	-0.39093	0.027

Figure 6. creditcard1.csv in excel

SCREENSHOTS

IMPORTING ESSENTIAL LIBRARIES

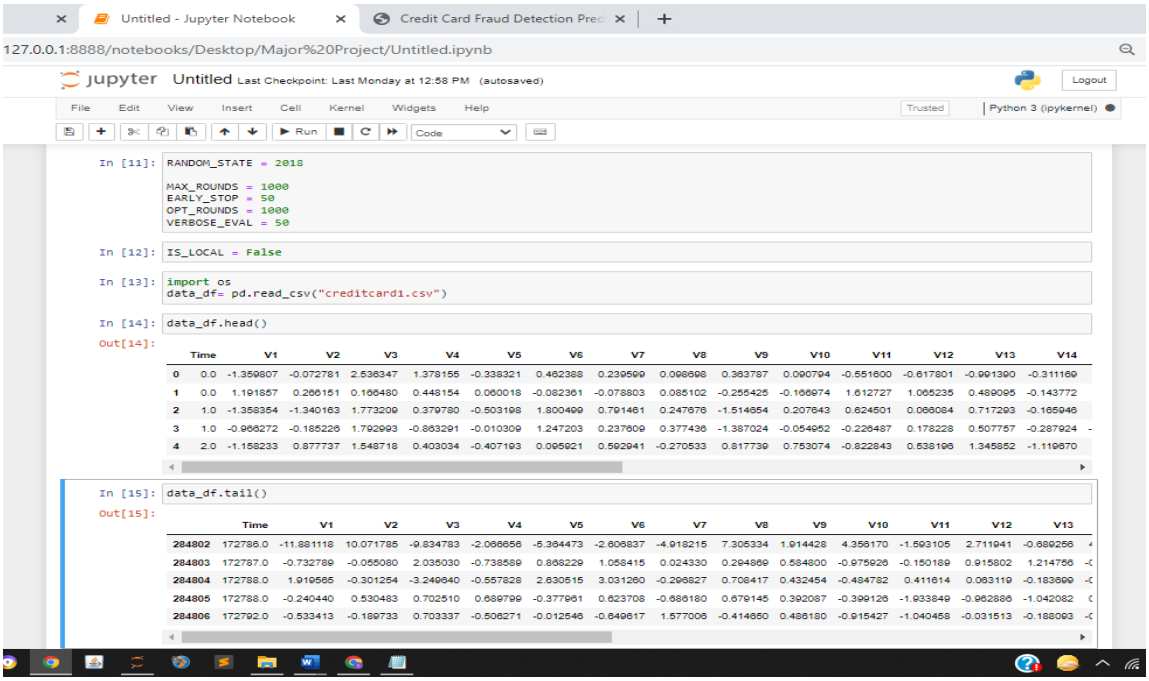
Here pandas, matplotlib, seaborn, Numpy, etc., libraries are imported.



```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
        import plotly.graph_objs as go
        import plotly.figure_factory as ff
        from sklearn import metrics

        from plotly import tools
        from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
        init_notebook_mode(connected=True)
```

Figure 7. Importing Libraries

Here credit card fraud data set. creditcard1.csv was imported.



Untitled - Jupyter Notebook | Credit Card Fraud Detection Pred

127.0.0.1:8888/notebooks/Desktop/Major%20Project/Untitled.ipynb

jupyter Untitled Last Checkpoint: Last Monday at 12:58 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
In [11]: RANDOM_STATE = 2018

         MAX_ROUNDS = 1000
         EARLY_STOP = 50
         OPT_ROUNDS = 1000
         VERBOSE_EVAL = 50

In [12]: IS_LOCAL = False

In [13]: import os
         data_df= pd.read_csv("creditcard1.csv")

In [14]: data_df.head()
```

Out[14]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670

```
In [15]: data_df.tail()
```

Out[15]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	4.356170	-1.593105	2.711941	-0.689256
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	-0.975926	-0.150189	0.915802	1.214756
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	-0.484782	0.411614	0.063119	-0.183699
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	-0.399126	-1.933849	-0.962886	-1.042082
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	-0.915427	-1.040458	-0.031513	-0.188093

Figure 8. Reading the csv file from the dataset.

DATA PREPROCESSING

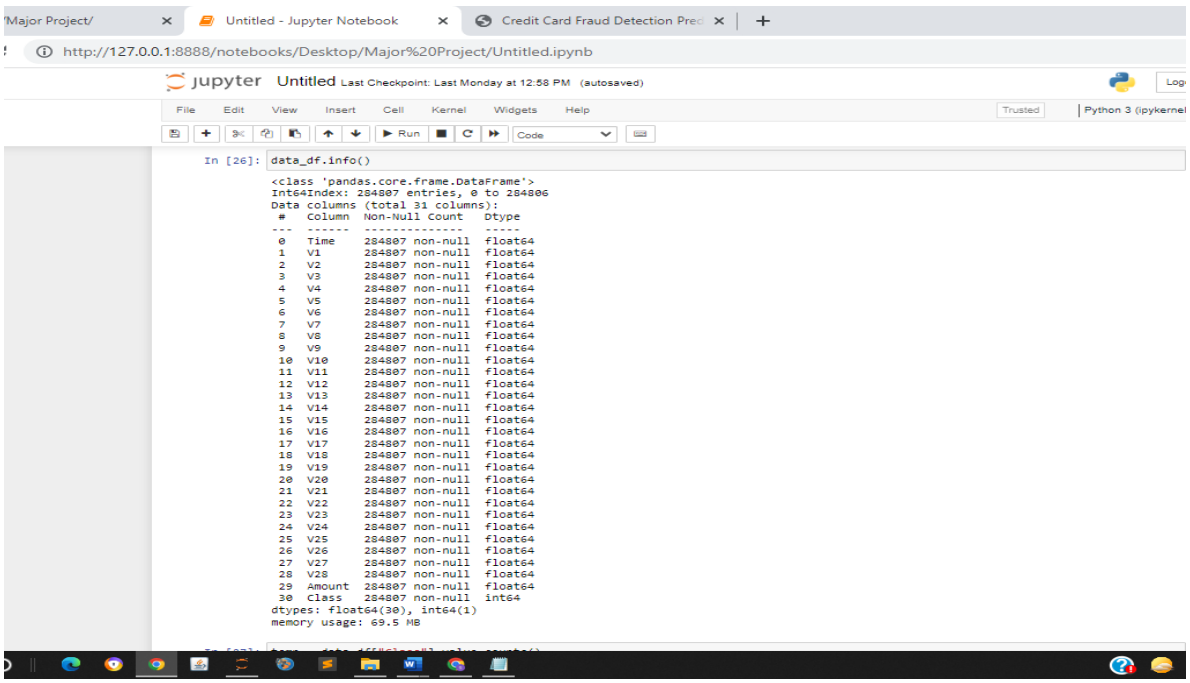


Figure 9. Null values and features

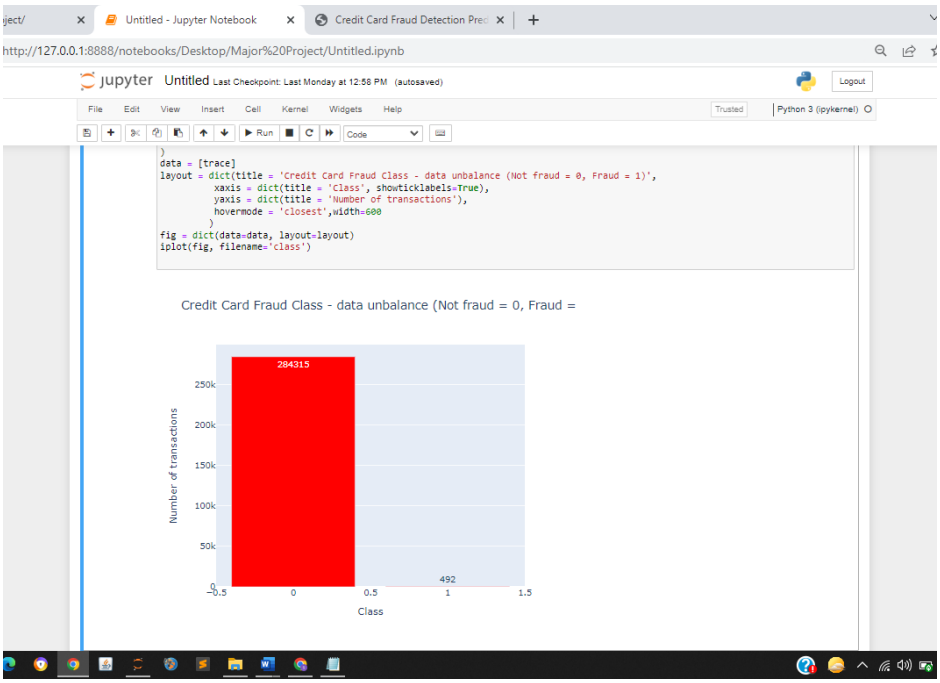


Figure 10. Target class with and without fraud.

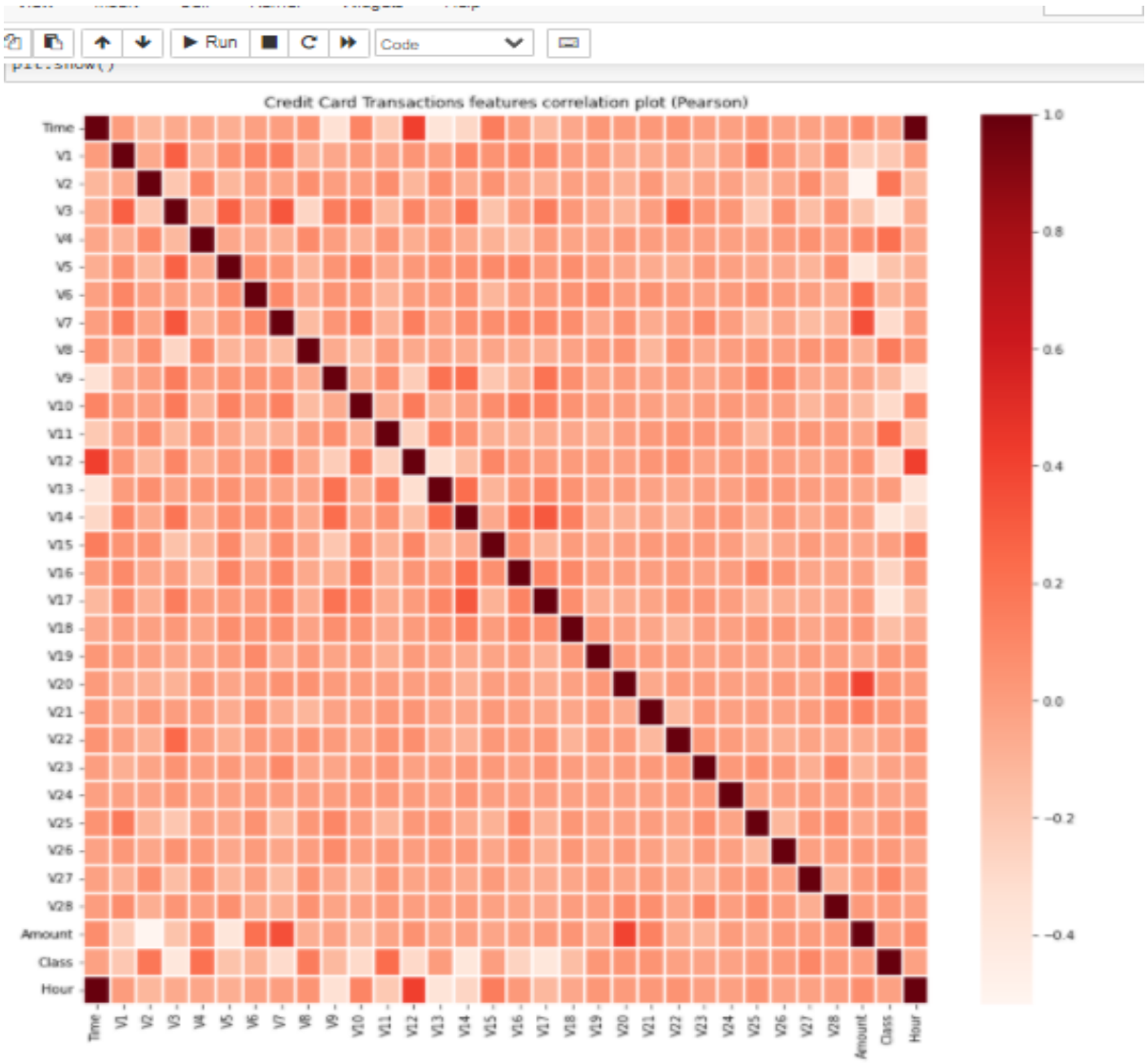


Figure 11. Correlation matrix using heatmap

FEATURE DENSITY PLOT

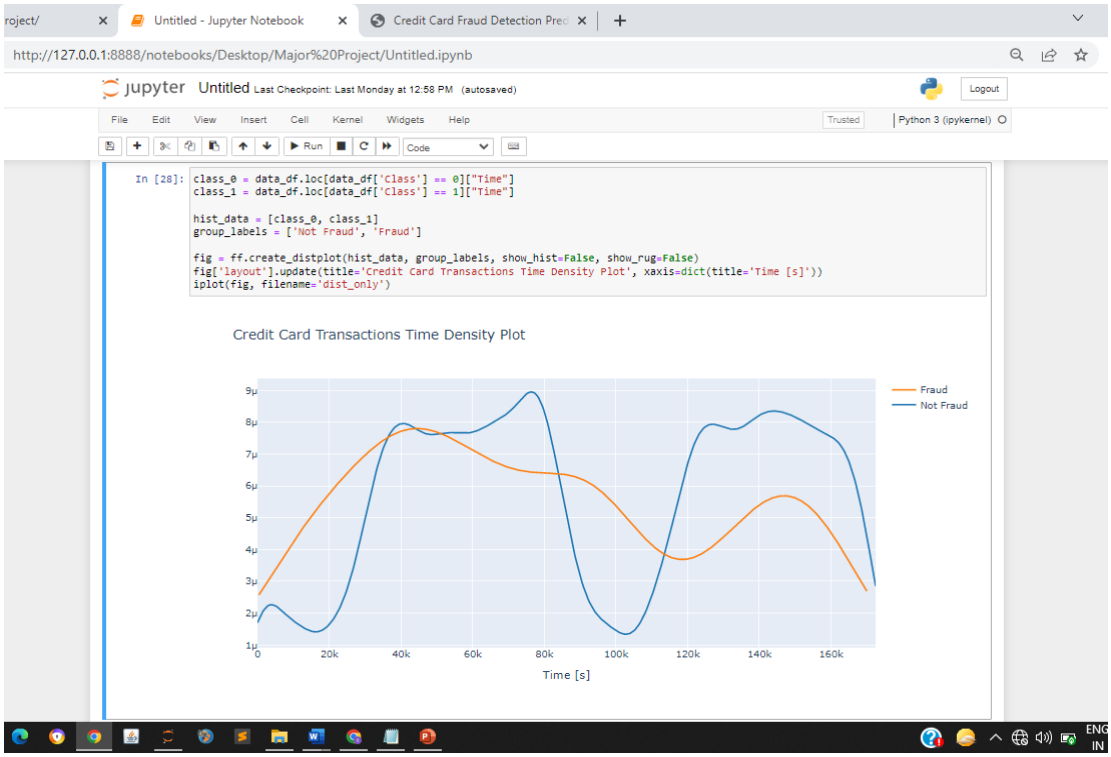


Figure 12. Feature density plot

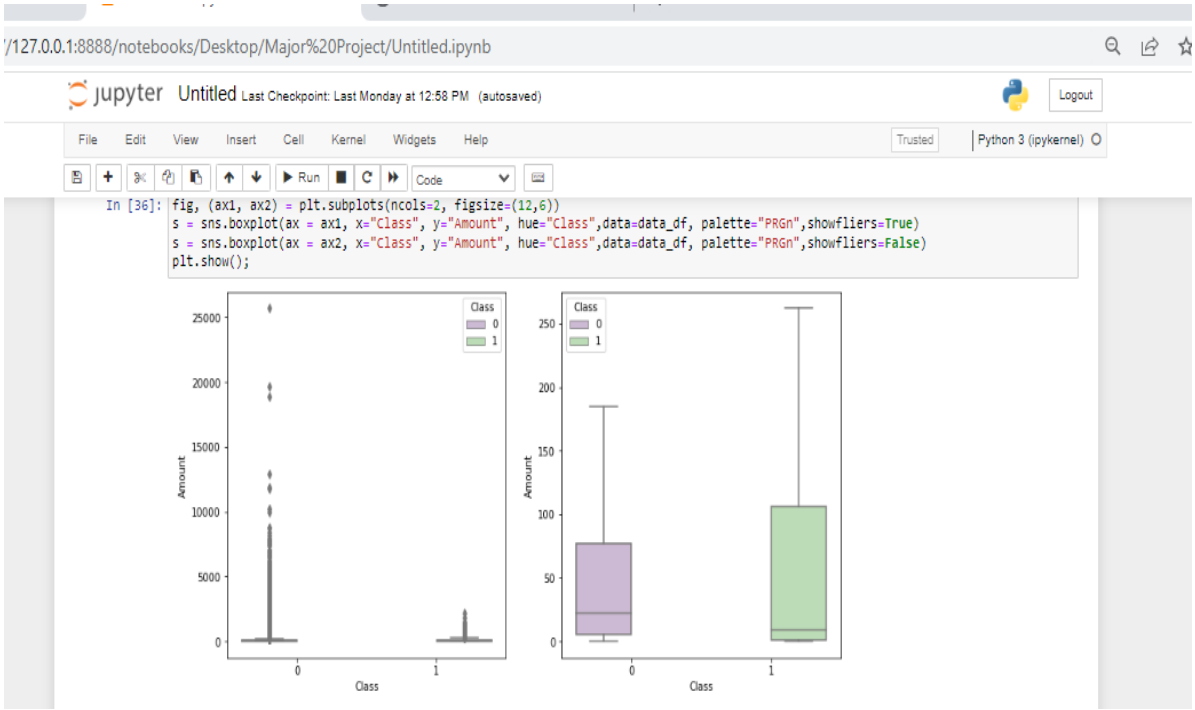


Figure 13. Box plot

```
In [29]: data_df['Hour'] = data_df['Time'].apply(lambda x: np.floor(x / 3600))

         tmp = data_df.groupby(['Hour', 'Class'])['Amount'].aggregate(['min', 'max', 'count', 'sum', 'mean', 'median', 'var']).reset_index
         df = pd.DataFrame(tmp)
         df.columns = ['Hour', 'Class', 'Min', 'Max', 'Transactions', 'Sum', 'Mean', 'Median', 'Var']
         df.head()
```

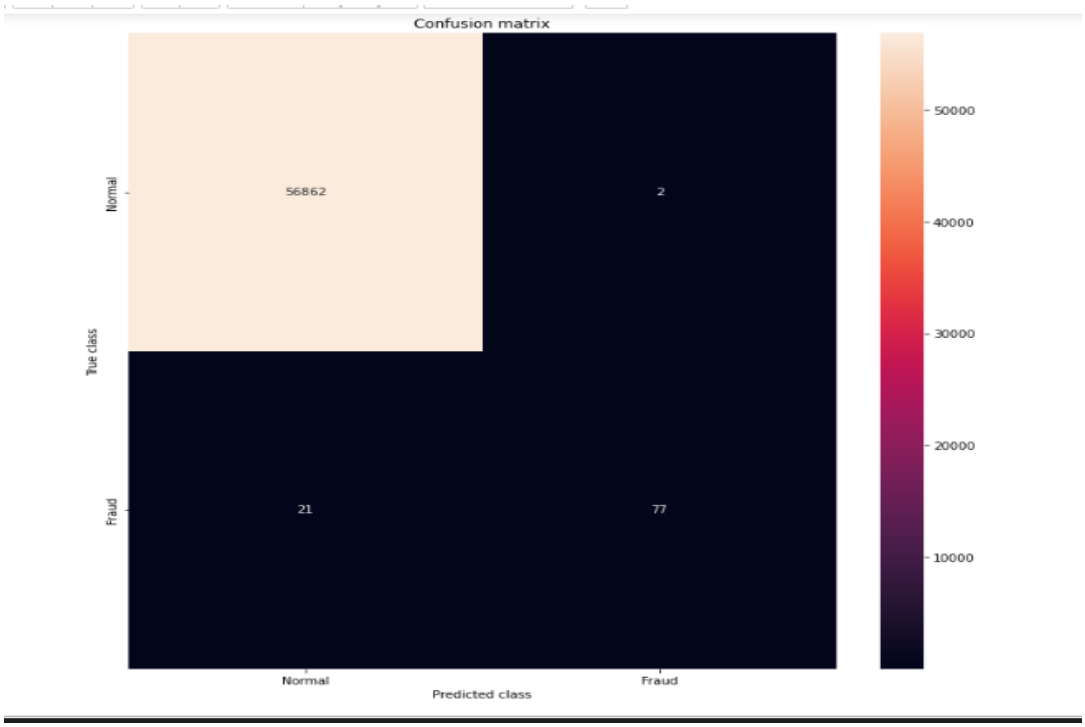
Out[29]:

	Hour	Class	Min	Max	Transactions	Sum	Mean	Median	Var
0	0.0	0	0.0	7712.43	3961	256572.87	64.774772	12.990	45615.821201
1	0.0	1	0.0	529.00	2	529.00	264.500000	264.500	139920.500000
2	1.0	0	0.0	1769.69	2215	145806.76	65.826980	22.820	20053.615770
3	1.0	1	59.0	239.93	2	298.93	149.465000	149.465	16367.832450
4	2.0	0	0.0	4002.88	1555	106989.39	68.803468	17.900	45355.430437

```
In [30]: fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18,6))
         s = sns.lineplot(ax = ax1, x="Hour", y="Sum", data=df.loc[df.Class==0])
         s = sns.lineplot(ax = ax2, x="Hour", y="Sum", data=df.loc[df.Class==1], color="red")
         plt.suptitle("Total Amount")
         plt.show();
```

Figure 14. Preparing data for model, scaling the data

RANDOM FOREST CLASSIFIER

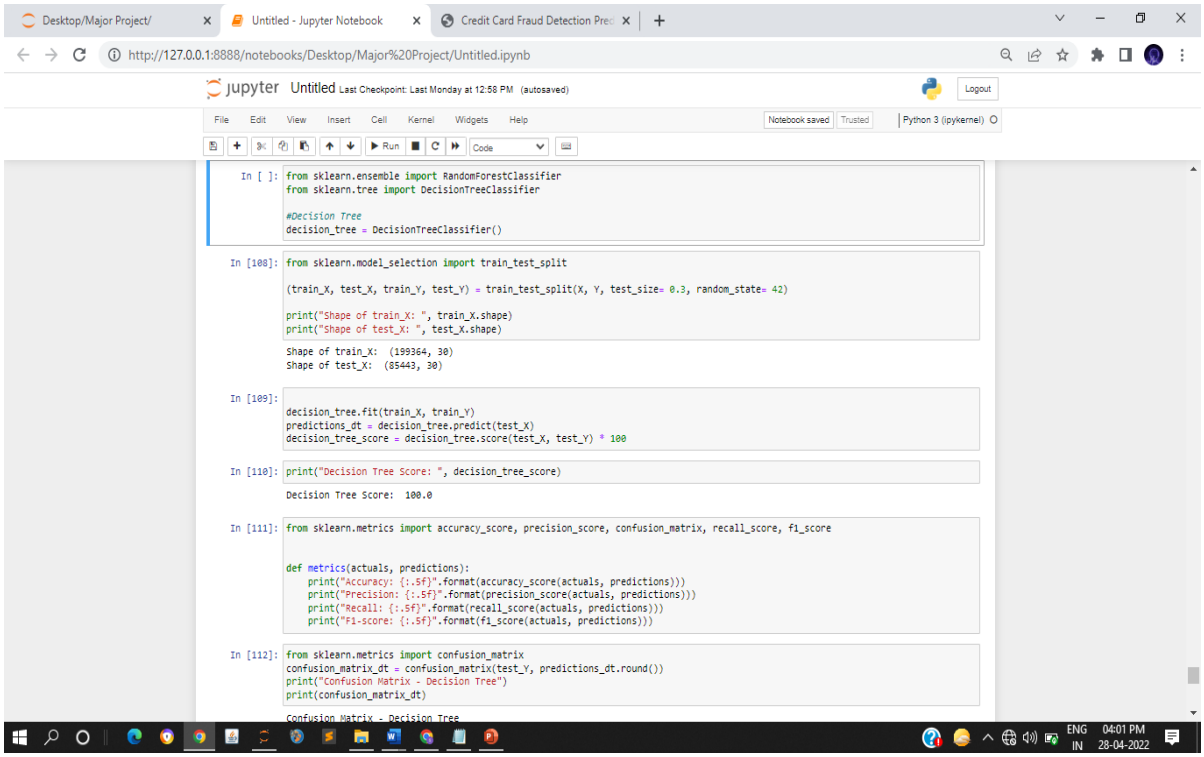


```
Random Forest: 23
0.9995962220427653
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.97      0.79      0.87        98

    accuracy                           1.00     56962
   macro avg       0.99      0.89      0.93     56962
weighted avg       1.00      1.00      1.00     56962


<Figure size 648x504 with 0 Axes>
```

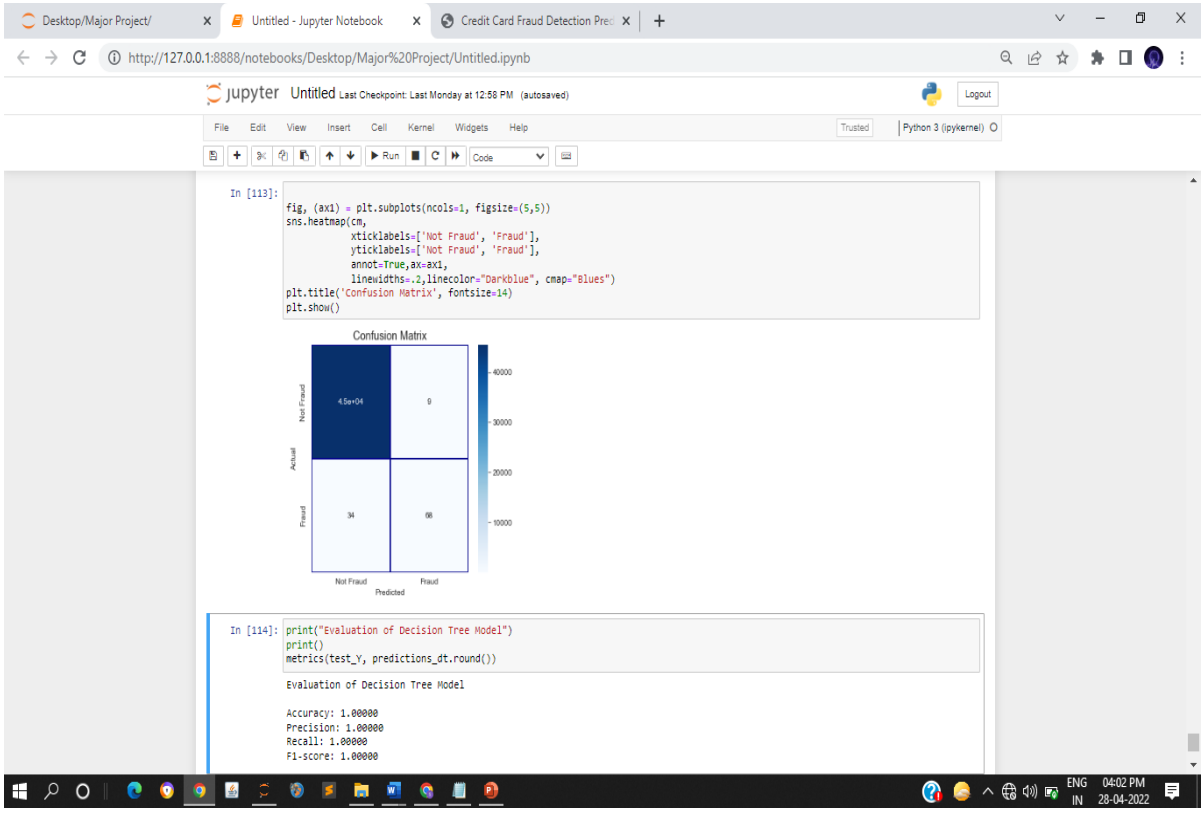
```
In [206]: roc_auc_score(valid_df[target].values, preds)

Out[206]: 0.8749250711823767
```

Figure 15. Random forest classifier

DECISION TREE



1



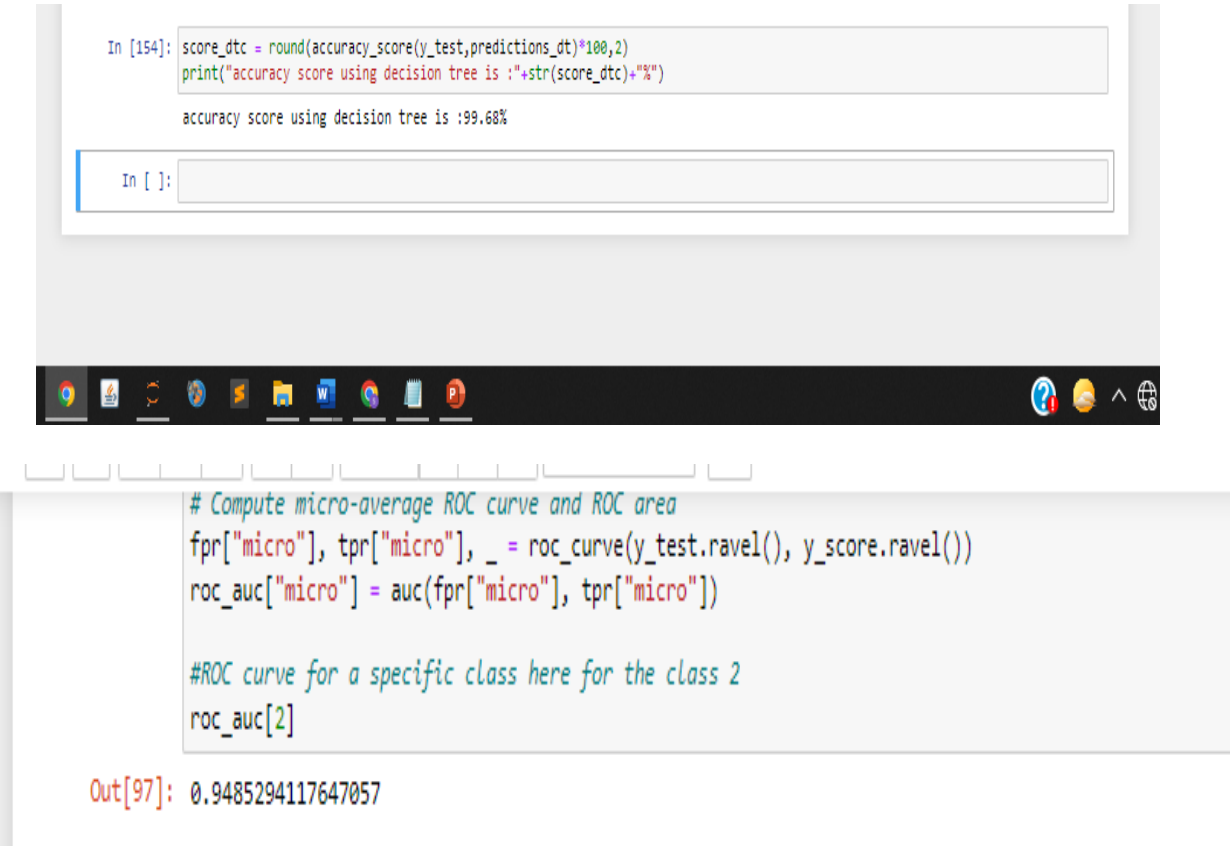
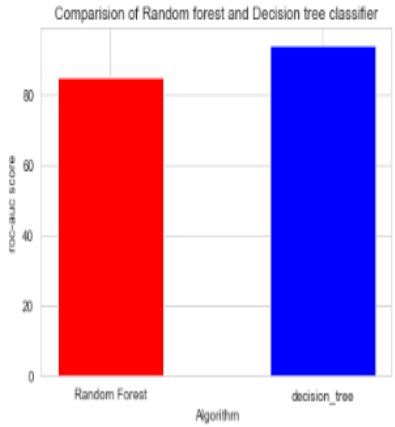


Figure 16. Decision Tree

VISUALIZATION

```
In [250]: from sklearn.metrics import roc_curve,roc_auc_score

In [257]: import matplotlib.pyplot as plt
          x= ['Random Forest', 'decision_tree']
          y= [85,94]
          plt.bar(x,y,color=['red','blue'],width=0.5)
          plt.title('Comparision of Random forest and Decision tree classifier')
          plt.xlabel('Algorithm')
          plt.ylabel('roc-auc score')
          plt.show()
```




```
In [ ]:
```

Figure 17.Bar plot of Random forest and Decision tree

SAMPLE CODING:

```
import pandas as pd

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import plotly.graph_objs as go

import plotly.figure_factory as ff

from sklearn import metrics


from plotly import tools

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

init_notebook_mode(connected=True)

!pip install plotly

import gc

from datetime import datetime

from sklearn.model_selection import train_test_split

from sklearn.metrics import roc_auc_score

from sklearn.ensemble import RandomForestClassifier

VALID_SIZE = 0.20

TEST_SIZE = 0.20

RANDOM_STATE = 2018
```

```
MAX_ROUNDS = 1000

EARLY_STOP = 50

OPT_ROUNDS = 1000

VERBOSE_EVAL = 50

IS_LOCAL = False

import os

data_df= pd.read_csv("creditcard1.csv")

data_df.head()

data_df.tail()

print("Credit Card Fraud Detection data - rows:",data_df.shape[0]," columns:", data_df.shape[1])

data_df.describe()

total = data_df.isnull().sum().sort_values(ascending = False)

percent = (data_df.isnull().sum()/data_df.isnull().count()*100).sort_values(ascending = False)

pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).transpose()

data_df=data_df.dropna()

data_df.isnull().sum()

data_df

data_df.shape

data_df['Class'].unique()

total = data_df.isnull().sum().sort_values(ascending = False)

percent = (data_df.isnull().sum()/data_df.isnull().count()*100).sort_values(ascending = False)

pd.concat([total, percent], axis=1, keys=['Total', 'Percent']).transpose()

data_df.info()
```

```
temp = data_df["Class"].value_counts()
df = pd.DataFrame({'Class': temp.index,'values': temp.values})

trace = go.Bar(
    x = df['Class'],y = df['values'],
    name="Credit Card Fraud Class - data unbalance (Not fraud = 0, Fraud = 1)",
    marker=dict(color="Red"),
    text=df['values']
)
data = [trace]
layout = dict(title = 'Credit Card Fraud Class - data unbalance (Not fraud = 0, Fraud = 1)',
          xaxis = dict(title = 'Class', showticklabels=True),
          yaxis = dict(title = 'Number of transactions'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='class')
class_0 = data_df.loc[data_df['Class'] == 0]["Time"]
class_1 = data_df.loc[data_df['Class'] == 1]["Time"]

hist_data = [class_0, class_1]
group_labels = ['Not Fraud', 'Fraud']

fig = ff.create_distplot(hist_data, group_labels, show_hist=False, show_rug=False)
```

```
fig['layout'].update(title='Credit Card Transactions Time Density Plot', xaxis=dict(title='Time [s]'))

iplot(fig, filename='dist_only')

data_df['Hour'] = data_df['Time'].apply(lambda x: np.floor(x / 3600))


tmp = data_df.groupby(['Hour', 'Class'])['Amount'].aggregate(['min', 'max', 'count', 'sum', 'mean', 'median', 'var']).reset_index()

df = pd.DataFrame(tmp)

df.columns = ['Hour', 'Class', 'Min', 'Max', 'Transactions', 'Sum', 'Mean', 'Median', 'Var']

df.head()

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12,6))

s = sns.boxplot(ax = ax1, x="Class", y="Amount", hue="Class",data=data_df, palette="PRGn",showfliers=True)

s = sns.boxplot(ax = ax2, x="Class", y="Amount", hue="Class",data=data_df, palette="PRGn",showfliers=False)

plt.show();

plt.figure(figsize = (14,14))

plt.title('Credit Card Transactions features correlation plot (Pearson)')

corr = data_df.corr()

sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,linewidths=.1,cmap="Reds")

plt.show()

target = 'Class'

predictors = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',\
    'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19',\
    'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',\
```

```
    'Amount']
X=data_df.iloc[:,0:30]
y=data_df['Class']
train_df, test_df = train_test_split(data_df, test_size=TEST_SIZE, random_state=RANDOM_STATE, shuffle=True )
train_df, valid_df = train_test_split(train_df, test_size=VALID_SIZE, random_state=RANDOM_STATE, shuffle=True)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size= 0.3)
clf = RandomForestClassifier(n_jobs=NO_JOBS,
                             random_state=RANDOM_STATE,
                             criterion=RFC_METRIC,
                             n_estimators=NUM_ESTIMATORS,
                             verbose=False)
clf.fit(train_df[predictors],train_df[target].values)
preds = clf.predict(valid_df[predictors])
preds
cm = pd.crosstab(valid_df[target].values, preds, rownames=['Actual'], colnames=['Predicted'])
fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))
sns.heatmap(cm,
            xticklabels=['Not Fraud', 'Fraud'],
            yticklabels=['Not Fraud', 'Fraud'],
            annot=True,ax=ax1,
            linewidths=.2,linecolor="Darkblue", cmap="Blues")
```

```
plt.title('Confusion Matrix', fontsize=14)
plt.show()
y_train_forest = clf.predict(X_train)
y_test_forest = clf.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, y_test_forest)
cnf_matrix
acc_train_forest = metrics.accuracy_score(y_train,y_train_forest)
acc_test_forest = metrics.accuracy_score(y_test,y_test_forest)
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))
print()

f1_score_train_forest = metrics.f1_score(y_train,y_train_forest)
f1_score_test_forest = metrics.f1_score(y_test,y_test_forest)
print("Random Forest : f1_score on training Data: {:.3f}".format(f1_score_train_forest))
print("Random Forest : f1_score on test Data: {:.3f}".format(f1_score_test_forest))
print()

recall_score_train_forest = metrics.recall_score(y_train,y_train_forest)
recall_score_test_forest = metrics.recall_score(y_test,y_test_forest)
print("Random Forest : Recall on training Data: {:.3f}".format(recall_score_train_forest))
print("Random Forest : Recall on test Data: {:.3f}".format(recall_score_test_forest))
print()
```

```
precision_score_train_forest = metrics.precision_score(y_train,y_train_forest)

precision_score_test_forest = metrics.precision_score(y_test,y_test_forest)

print("Random Forest : precision on training Data: {:.3f}".format(precision_score_train_forest))

print("Random Forest : precision on test Data: {:.3f}".format(precision_score_test_forest))

print(metrics.classification_report(y_test, y_test_forest))

training_accuracy = []

test_accuracy = []

# try max_depth from 1 to 20

depth = range(1,20)

for n in depth:

    forest_test =  RandomForestClassifier(n_estimators=n)

print("Accuracy:",metrics.accuracy_score(y_test, y_test_forest))

roc_auc_score(valid_df[target].values, preds)

score_dtc = round(accuracy_score(y_test,predictions_dt)*100,2)

print("accuracy score using random forest  is :"+str(score_dtc)+"%")

import numpy as np

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data_df["NormalizedAmount"] =  scaler.fit_transform(data_df["Amount"].values.reshape(-1, 1))

data_df.drop(["Amount", "Time"], inplace= True, axis= 1)

Y = data_df["Class"]

X = data_df.drop(["Class"], axis= 1)

from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.tree import DecisionTreeClassifier


#Decision Tree

decision_tree = DecisionTreeClassifier()

from sklearn.model_selection import train_test_split


(train_X, test_X, train_Y, test_Y) = train_test_split(X, Y, test_size= 0.3, random_state= 42)


print("Shape of train_X: ", train_X.shape)

print("Shape of test_X: ", test_X.shape)

decision_tree.fit(train_X, train_Y)

predictions_dt = decision_tree.predict(test_X) from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, recall_score, f1_score


fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))

sns.heatmap(cm,

        xticklabels=['Not Fraud', 'Fraud'],

        yticklabels=['Not Fraud', 'Fraud'],

        annot=True,ax=ax1,

        linewidths=.2,linecolor="Darkblue", cmap="Blues")

plt.title('Confusion Matrix', fontsize=14)

plt.show()def metrics(actuals, predictions):

    print("Accuracy: {:.5f}".format(accuracy_score(actuals, predictions)))

    print("Precision: {:.5f}".format(precision_score(actuals, predictions)))
```

```
    print("Recall: {:.5f}".format(recall_score(actuals, predictions)))
print("F1-score: {:.5f}".format(f1_score(actuals, predictions)))
decision_tree_score = decision_tree.score(test_X, test_Y) * 100
print("Decision Tree Score: ", decision_tree_score)
from sklearn.metrics import confusion_matrix
confusion_matrix_dt = confusion_matrix(test_Y, predictions_dt.round())
print("Confusion Matrix - Decision Tree")
print(confusion_matrix_dt)
print("Evaluation of Decision Tree Model")
print()
metrics(test_Y, predictions_dt.round())
score_dtc = round(accuracy_score(y_test,predictions_dt)*100,2)
print("accuracy score using decision tree is :"+str(score_dtc)+"%")
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.tree import DecisionTreeClassifier
from scipy import interp
iris = datasets.load_iris()
X = iris.data
y = iris.target
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5, random_state=0)
classifier = DecisionTreeClassifier()
```

```
y_score = classifier.fit(X_train, y_train).predict(X_test)


fpr = dict()
tpr =  dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])


# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])


#ROC curve for a specific class here for the class 2
roc_auc[2]
import matplotlib.pyplot as plt
x= ['Random Forest', 'decision_tree']
y= [85,94]
plt.bar(x,y,color=['red','blue'],width=0.5)
plt.title('Comparision of Random forest and Decision tree classifier')
plt.xlabel('Algorithm')
plt.ylabel('roc-auc score')
plt.show()
```