CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS

A Project

Report Submitted to the

Government Arts and Science College, Srivilliputtur

(Affiliated to Madurai Kamaraj University, Madurai)

In partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science

Submitted by

L.Abinaya

Reg.No.: C0S42601

Under the guidance of

Dr. Saravanan Suba, M.C.A., M.Phil., M.Tech., Ph.D.,

Department of Computer Science

Government Arts and Science College

Srivilliputtur 626125

APRIL – 2023

DECLARATION

I hereby declare that the project work entitled **"CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS"** is a bonafide work done by me for the partial fullfilment of the requirements for the degree of Bachelor of Science in Computer Science. I also hereby declare that this work done by me with my own effort and I assure that this project has not been submitted to any other university for the award of any degree.

Place : Srivilliputtur

Date :

Signature of Candidates

L.Abinaya

(Reg.No.:C0S42601)

Government Arts and Science College

Srivilliputtur-626125

Department of Computer Science

Certificate

This is to certify that the project work entitled **"CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS"** is a bonafide work done by L.Abinaya(Reg.No.C0S42601) for the partial fulfilment of the requirements for the award of the degree of Bachelor of Science in Computer Science during six semester of academic year 2022-2023.

Guide **Head of the Department**

Submitted for viva-voice examination held on……

Internal Examiner **External Examiner**

Acknowledgement

First of all, I render my humble prostration to the God Almighty for giving strength and will power that enables me to make this project successful.

I can hardly find words to express my gratitude to the help and warm encouragement that myself received from my beloved parents.

I express my sincere gratitude to **Dr. Saravanan Suba, M.C.A., M.Phil., M.Tech., Ph.D.,** Head of the Department of Computer Science, Government Arts And Science College, Srivilliputtur, for giving me this opportunity to carry out this project.

I would like to express my heartfelt gratitude and thanks to **Dr. Saravanan Suba, M.C.A., M.Phil., M.Tech., Ph.D.,** my project guide for his consistent help and suggestions. I express my gratitude to all our staff members of the Department of Computer Science for their valuable suggestions and advice throughout my project.

Finally, I would like to thank all those who directly or indirectly helped me in the successful completion of the project.

L.Abinaya

ABSTRACT

The project is mainly focussed on credit card fraud detection in real world.A phenomenal growth in the number of credit card transactions, has recently ledto a considerable rise in fraudulent activities. The purpose is to obtain goods without paying, or to obtain unauthorized funds from an account. Implementation of efficient fraud detection systems has become imperative for all credit card issuing banks to minimize their losses.

One of the most crucial challenges in making the business is that neither the card nor the cardholder needs to be present when the purchase is being made. This makes it impossible for the merchant to verify whether the customer making a purchase is the authentic cardholder or not. With the proposed scheme, using random forest algorithm the accuracy of detecting the fraud can be improved canbe improved.

Classification process of random forest algorithm to analyse data set and user current dataset. Finally optimize the accuracy of the result data. The performance of the techniques is evaluated based on accuracy ,sensitivity, and specificity, and precision. Then processing of some of the attributes provided identifies the fraud detection and provides the graphical model visualization. The performance of the techniques is evaluated based on accuracy..

LIST OF CONTENTS

LIST OF ABREVIATION

ML - Machine Learning

AI - Artificial Intelligence

SVM - Support Vector Machine

PCA - Principle Component Analysis

NumPY – Numerical Python

GSoC - Google summer of code

CSV -Comma separated value

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There are various fraudulent activities detection techniques has implemented in credit card transactions have been kept in researcher minds to methods to develop models based on artificial intelligence, data mining, fuzzy logic and machine learning. Credit card fraud detection is significantly difficult, but also popular problem to solve. In our proposed system we built the credit card fraud detection using Machine learning. With the advancement of machine learning techniques. Machine learning has been identified as a successful measure for fraud detection. A large amount of data is transferred during online transaction processes, resulting in a binary result: genuine or fraudulent.

Within the sample fraudulent datasets, features are constructed. These are data points namely the age and value of the customer account, as well as the origin of the credit card. There are hundreds of features and each contributes, to varying extents, towards the fraud probability We use supervised learning algorithm such as Random forest algorithm to classify the fraud card transaction in online or by offline. Random forest is advanced version of Decision tree. Random forest has better efficiency and accuracy than the other machine learning algorithms. Random forest aims to reduce the previously mentioned correlation issue by picking only a subsample of the feature space at each split.

1.1 MOTIVATION OF WORK

The use of machine learning in fraud detection has been an interesting topic now days. A credit card fraud detection algorithm consists in identifying those transactions with a high probability of being fraud ,based on historical fraud patterns. Machine learning, having three types, from that also the supervised and hybrid approach is more suitable for fraud detection

1.2 PROBLEM STATEMENT

Billions of dollars of loss are caused every year by the fraudulent credit card transactions. Fraud is old as humanity itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there is definitely an urge to solve the problem of credit card fraud detection. Moreover, the development of new technologies provides additional ways in which criminals may commit fraud. The use of credit cards is prevalent in modern day society and credit card fraud has been kept on growing in recent years. Hugh Financial losses has been fraudulent affects not only merchants and banks, but also individual person who are using the credits. Fraud may also affect the reputation and image of a merchant causing non-financial losses that, though difficult to quantify in the short term, may become visible in the long period. For example, if a cardholder is victim of fraud with a certain company, he may no longer trust their business and choose a contender

1.3 SCOPE OF THE PROJECT

In this proposed project we designed a protocol or a model to detect the fraud activity in credit card transactions. This system is capable of providing most of the essential features required to detect fraudulent and legitimate transactions. As technology changes, it becomes difficult to track the behaviour and pattern of fraudulent transactions. With the upsurge of machine learning, artificial intelligence and other relevant fields of information technology, it becomes feasible to automate the process and to save some of the effective amount of labor that is put into detecting credit card fraudulent activities.

CHAPTER 2

LITERATURE SURVEY

[1] The Use of Predictive Analytics Technology to Detect Credit Card Fraud in Canada. "Kosemani Temitayo Hafiz, Dr. Shaun Aghili, Dr. Pavol Zavarsky."

This research paper focuses on the creation of a scorecard from relevant evaluation criteria, features, and capabilities of predictive analytics vendor solutions currently being used to detect credit card fraud. The scorecard provides a side-byside comparison of five credit card predictive analytics vendor solutions adopted in Canada. From the ensuing research findings, a list of credit card fraud PAT vendor solution challenges, risks, and limitations was outlined.

[2] BLAST-SSAHA Hybridization for Credit Card Fraud Detection. "Amlan Kundu, Suvasini Panigrahi, Shamik Sural, Senior Member, IEEE, and Arun K. Majumdar"

This paper propose to use two-stage sequence alignment in which a profile Analyser (PA) first determines the similarity of an incoming sequence of transactions on a given credit card with the genuine cardholder's past spending sequences. The unusual transactions traced by the profile analyser are next passed on to a deviation analyser (DA) for possible alignment with past fraudulent behaviour. The final decision about the nature of a transaction is taken on the basis of the observations by these two analysers. In order to achieve online response time for both PA and DA, we suggest a new approach for combining two sequence alignment algorithms BLAST and SSAHA.

[3] Research on Credit Card Fraud Detection Model Based on Distance Sum."Wen-Fang YU, Na Wang".

Along with increasing credit cards and growing trade volume in China, credit card fraud rises sharply. How to enhance the detection and prevention of credit card fraud becomes the focus of risk control of banks. It proposes a credit card fraud detection model using outlier detection based on distance sum according to the infrequency and unconventionality of fraud in credit card

transaction data, applying outlier mining into credit card fraud detection. Experiments show that this model is feasible and accurate in detecting credit card fraud.

[4] Fraudulent Detection in Credit Card System Using SVM & Decision Tree. "Vijayshree B. Nipane, Poonam S. Kalinge, Dipali Vidhate, Kunal War, Bhagyashree P. Deshpande".

With growing advancement in the electronic commerce field, fraud is spreading all over the world, causing major financial losses. In current scenario, Major cause of financial losses is credit card fraud; it not only affects trades person but also individual clients. Decision tree, Genetic algorithm, Meta learning strategy, neural network, HMM are the presented methods used to detect credit card frauds. In contemplate system for fraudulent detection, artificial intelligence concept of Support Vector Machine (SVM) & decision tree is being used to solve the problem. Thus by implementation of this hybrid approach, financial losses can be reduced to greater extend.

[5] Supervised Machine (SVM) Learning for Credit Card Fraud Detection. "Sitaram patel, Sunita Gond".

This thesis propose the SVM (Support Vector Machine) based method with multiple kernel involvement which also includes several fields of user profile instead of only spending profile. The simulation result shows improvement in TP (true positive), TN (true negative) rate, & also decreases the FP (false positive) & FN (false negative) rate.

[6] Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. "Y. Sahin and E. Duman"

In this study, classification models based on decision trees and support vector machines (SVM) are developed and applied on credit card fraud detection problem. This study is one of the firsts to compare the performance of SVM and decision tree methods in credit card fraud detection with a Real data set.

CHAPTER 3
OVERVIEW OF SYSTEM

3.1 EXISTING SYSTEM

In existing System, a research about a case study involving credit card fraud detection, where data normalization is applied before Cluster Analysis and with results obtained from the use of Cluster Analysis and Artificial Neural Networks on fraud detection has shown that by clustering attributes neuronal inputs can be minimized. And promising results can be obtained by using normalized data and data should be MLP trained. This research was based on unsupervised learning. Significance of this paper was to find new methods for fraud detection and to increase the accuracy of results. The data set for this paper is based on real life transactional data by a large European company and personal details in data is kept confidential. Accuracy of an algorithm is around 50%. Significance of this paper was to find an algorithm and to reduce the cost measure. The result obtained was by 23% and the algorithm they find was Bayes minimum risk.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

1. In this paper a new collative comparison measure that reasonably represents the gains and losses due to fraud detection is proposed.
2. A cost sensitive method which is based on Bayes minimum risk is presented using the proposed cost measure.

3.2 PROPOSED SYSTEMS

In proposed System, we are applying random forest algorithm for classification of the credit card dataset. And its accuracy is more than other two algorithm SVM and Logistic Regression. Random Forest is an algorithm for

classification and regression. Summarily, it is a collection of decision tree classifiers. has advantage over other as it corrects the habit of over fitting 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 each tree is trained independently of the others. The Random Forest algorithm has been found to provide a good estimate of the generalization error and to be resistant to over fitting.

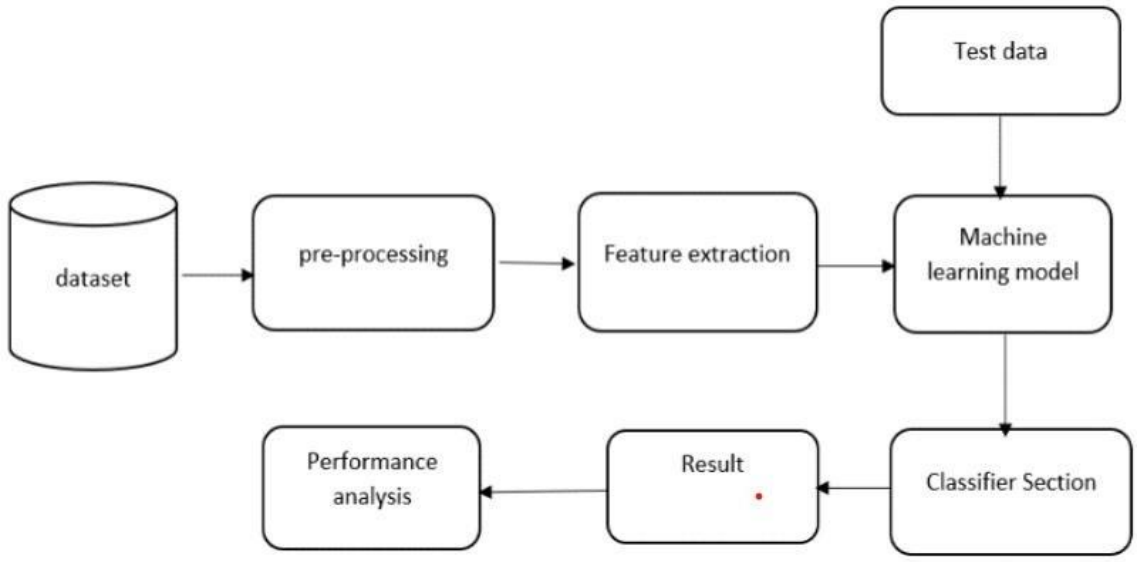


Figure 3.1 System Architecture of Proposed system

3.2.2 ADVANTAGES OF PROPOSED SYSTEM

- Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest.

- The 'amount' feature is the transaction amount. Feature 'class' is the target class for the binary classification and it takes value 1 for positive case (fraud) and 0 for negative case (not fraud).

CHAPTER -4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

- Processor - Intel
- RAM - 4 Gb
- Hard Disk - 260 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse

4.2 SOFTWARE REQUIREMENTS

- Python
- Anaconda
- OS - Windows 7, 8 and 10 (32 and 64 bit)

FUNCTIONAL REQUIREMENTS

- Functional requirements describe what the software should do (the functions). Think about the core operations.
- Because the “functions” are established before development, functional requirements should be written in the future tense. In developing the software for Stock Price Prediction, some of the functional requirements could include:

    - The software shall accept the data_set.csv dataset as input.
    - The software should shall do pre-processing (like verifying for missing data values) on input for model training.
    - The software shall use LSTM ARCHITECTURE as main component of the software.
    - It processes the given input data by producing the most possible outcomes of a CLOSING STOCK PRICE. Notice that each

requirement is directly related to what we expect the software to do. They represent some of the core functions.

NON-FUNCTIONAL REQUIREMENTS

- Product properties
    - **Usability**: It defines the user interface of the software in terms of simplicity of understanding the user interface of stock prediction software, for any kind of stock trader and other stakeholders in stock market.
    - **Efficiency**: maintaining the possible highest accuracy in the closing stock prices in shortest time with available data.
- Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

CHAPTER -5

TOOLS AND LIBRARIES

PYTHON:-

Python is an open source programming language. Python was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show. Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages. Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Some things that Python is often used for are:

- Web development
- Game programming
- Desktop GUIs
- Scientific programming

1.NUMPY:-

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project. Using NumPy, a developer can perform the following operations:

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

2.PANDAS:-

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice. Pandas deals with the following three data structures : • Series • DataFrame • Panel

3.MATPLOITLIB:-

It is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current

figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

4.SKLEARN:

Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k-means, random forests and DBSCAN. It is designed to work with Python Numpy and Scipy.The scikit-learn project kicked off as a Google Summer of Code (also known as GSoC) project by David Cournapeau as scikits.learn. It gets its name from “Scikit”, a separate third-party extension

CHAPTER 6

DATASET

DATASET USED - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

ABOUT DATASET

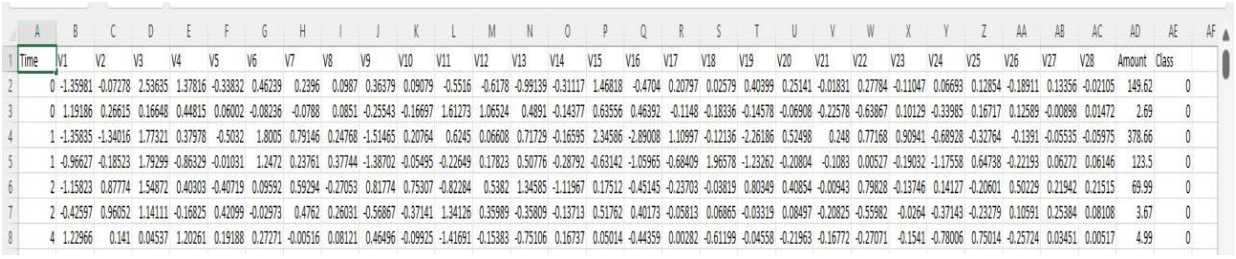
CONTEXT

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

CONTENT

The dataset contains transactions made by credit cards in by European cardholders.

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions..



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	Class
0	-1.35981	-0.07278	2.53635	1.37816	-0.33832	0.46239	0.2396	0.0987	0.36379	0.09079	-0.5516	-0.6178	-0.99139	-0.31117	1.46818	-0.4704	0.20797	0.02579	0.40399	0.25141	-0.01831	0.27784	-0.11047	0.06693	0.12854	-0.18911	0.13356	-0.02105	149.62	0
0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.08236	-0.0788	0.0851	-0.25543	-0.16697	1.61273	1.06524	0.4891	-0.14377	0.63556	0.46392	-0.1148	-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.10129	-0.33985	0.16717	0.12589	-0.00898	0.01472	2.69	0
1	-1.35835	-1.34016	1.77321	0.37978	-0.5032	1.8005	0.79146	0.24768	-1.51465	0.20764	0.6245	0.06608	0.71729	-0.16595	2.34586	-2.89008	1.10997	-0.12136	-2.26186	0.52498	0.248	0.77168	0.90941	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
1	-0.96627	-0.18523	1.79299	-0.86329	-0.01031	1.2472	0.23761	0.37744	-1.38702	-0.05495	-0.22649	0.17823	0.50776	-0.28792	-0.63142	-1.05965	-0.68409	1.96578	-1.23262	-0.20804	-0.1083	0.00527	-0.19032	-1.17558	0.64738	-0.22193	0.06272	0.06146	123.5	0
2	-1.15823	0.87774	1.54872	0.40303	-0.40719	0.09592	0.59294	-0.27053	0.81774	0.75307	-0.82284	0.5382	1.34585	-1.11967	0.17512	-0.45145	-0.23703	-0.03819	0.80349	0.40854	-0.00943	0.79828	-0.13746	0.14127	-0.20601	0.50229	0.21942	0.21515	69.99	0
2	-0.42597	0.96052	1.14111	-0.16825	0.42099	-0.02973	0.4762	0.26031	-0.56867	-0.37141	1.34126	0.35989	-0.35809	-0.13713	0.51762	0.40173	-0.05813	0.06865	-0.03319	0.08497	-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.10591	0.25384	0.08108	3.67	0
4	1.22966	0.141	0.04537	1.20261	0.19188	0.27271	-0.00516	0.08121	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.16737	0.05014	-0.44359	0.00282	-0.61199	-0.04558	-0.21963	-0.16772	-0.27071	-0.1541	-0.78006	0.75014	-0.25724	0.03451	0.00517	4.99	0

Figures 6.1 Data Fields

It contains only numerical input variables which are the result of a PCA transformation .Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data.

Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'.

Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning.

Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

DATASET TYPE – .CSV FILE

A CSV is a comma-separated values file, which allows data to be saved in a tabular format. CSVs look like a garden-variety spreadsheet but with a .csv extension.

CSV files can be used with most any spreadsheet program, such as Microsoft Excel or Google Spreadsheets. They differ from other spreadsheet file types because you can only have a single sheet in a file, they can not save cell, column, or row. Also, you cannot not save formulas in this format.

WHY ARE .CSV FILES USED?

These files serve a number of different business purposes. They help companies export a high volume of data to a more concentrated database, for instance.

They also serve two other primary business functions:

- CSV files are plain-text files, making them easier for the website developer to create
- Since they're plain text, they're easier to import into a spreadsheet or another storage database, regardless of the specific software you're using
- To better organize large amounts of data

HOW DO I SAVE CSV FILES?

Saving CSV files is relatively easy, you just need to know where to change the file type.

Under the "File name" section in the "Save As" tab, you can select "Save as type" and change it to "CSV (Comma delimited) (*.csv). Once that option is selected, you are on your way to quicker and easier data organization. This should be the same for both Apple and Microsoft operating systems.

CHAPTER 7

IMPLEMENTATION

7.1 ALGORI THM

7.1.1 LOGISTIC REGRESSION

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

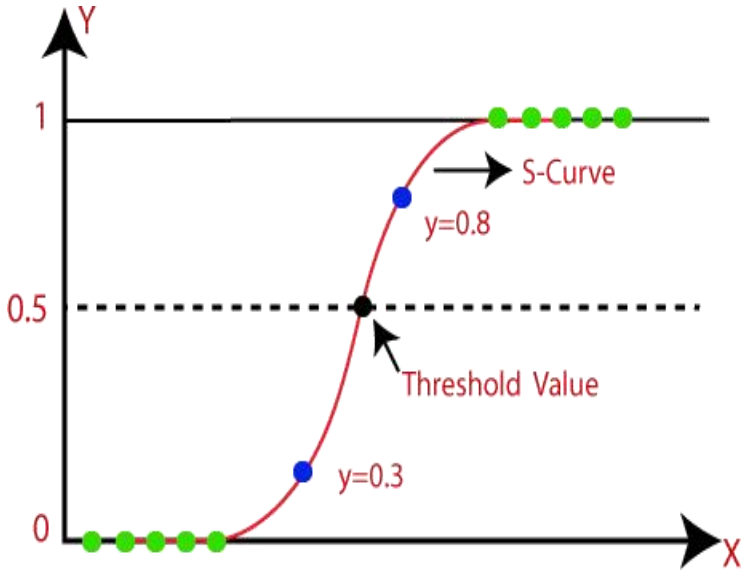


Figure7.1 Logistic Function

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Steps in Logistic Regression:

- Data Pre-processing step
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

7.1.2 SUPPORT VECTOR MACHINE

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

Hyperplane: There can be multiple lines/decision boundaries to segregate the c lasses in n-dimensional space, but we need to find out the best decision boundar y that helps to classify the data points. This best boundary is known as the hyper plane of SVM.

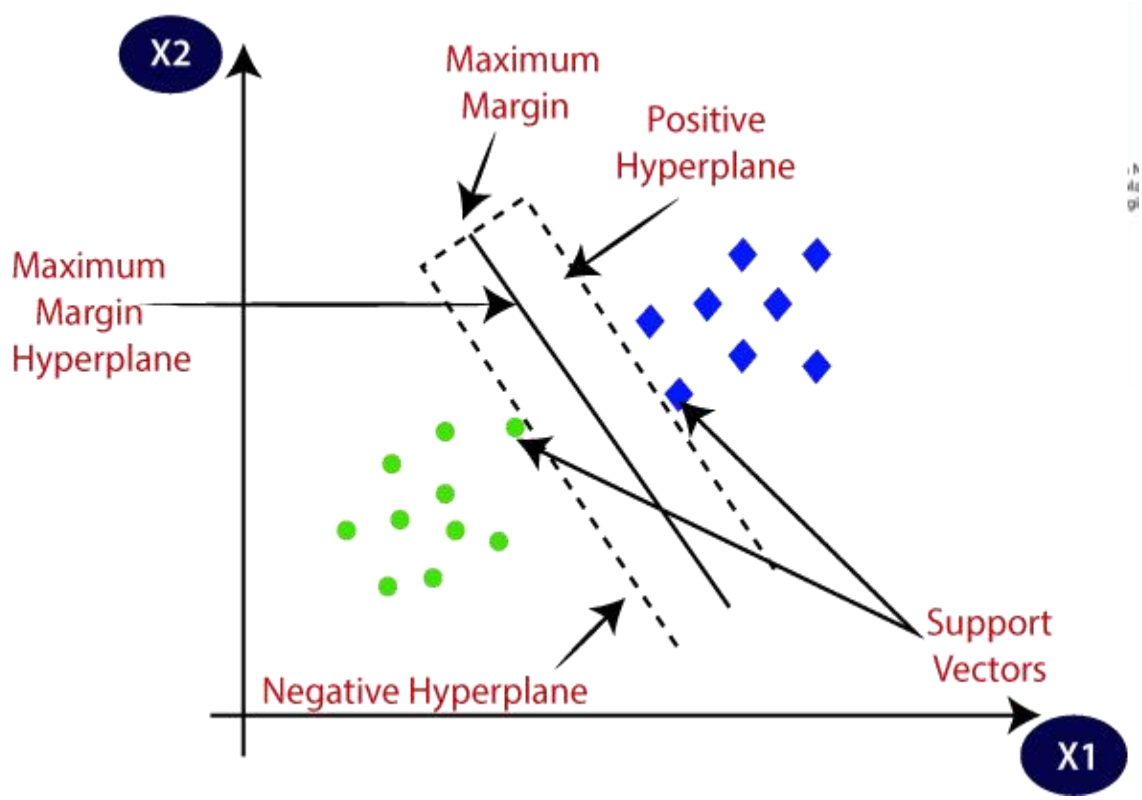


Figure7.2 Categories classified using hyperplane

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

SVM algorithm can be used for **Face detection, image classification, text cate gorization,** etc.

7.1.3 RANDOM FOREST

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.

WORKING OF RANDOM FOREST ALGORITHM

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes

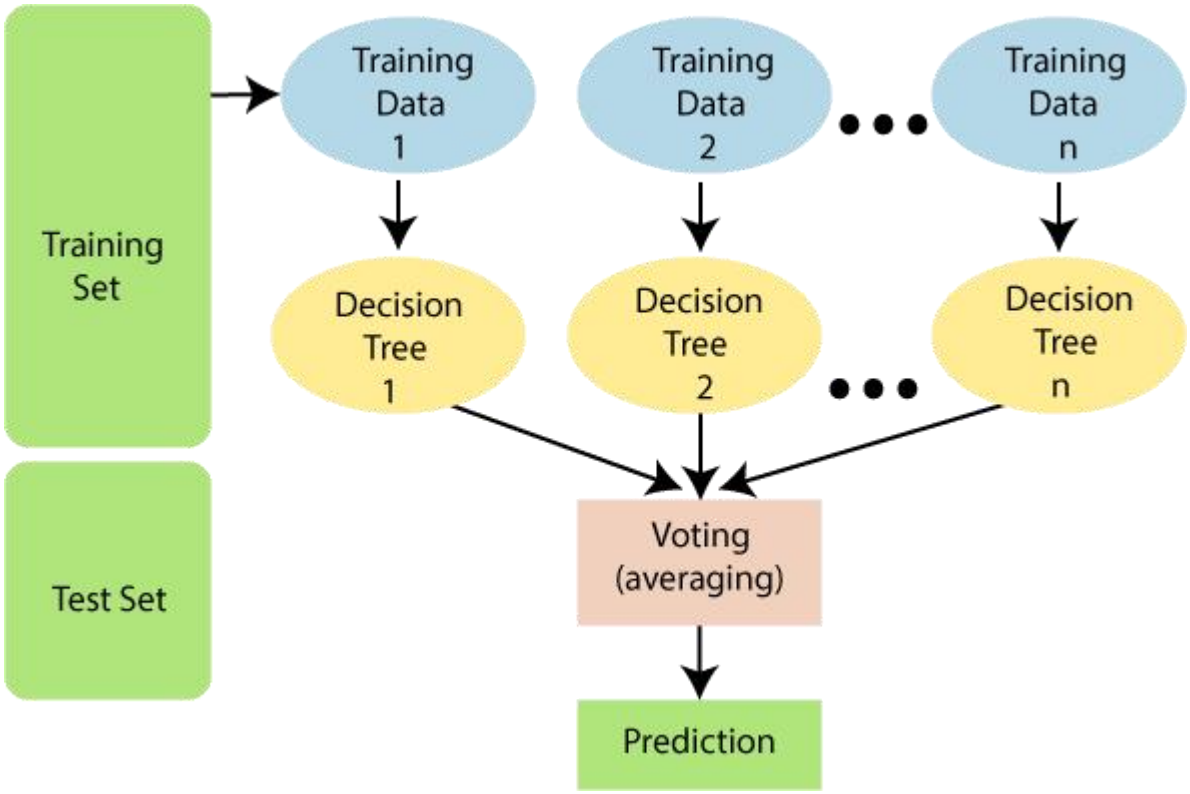


Figure 7.3 Working of Random Forest

ADVANTAGES OF RANDOM FOREST

- The random forest algorithm is not biased, since, there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore, the overall biasedness of the algorithm is reduced.

- This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees.

- The random forest algorithm works well when you have both categorical and numerical features

7.2 MODULE DESCRIPTION

7.2.1 MODULE 1: DATA COLLECTION

Data used in this paper is a set of product reviews collected from credit card transactions records. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called labelled data.

```
In [3]: import pandas as pd
        data = pd.read_csv(r"C:\Users\sound\svmpro\creditcard.csv")
```

Figure 7.4

7.2.1 MODULE 2: DATA VISUALIZATION

```
In [7]: # distribution of Time
        time = data['Time'].values
        sns.distplot(time)

C:\Users\sound\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

Out[7]: <AxesSubplot:ylabel='Density'>
```

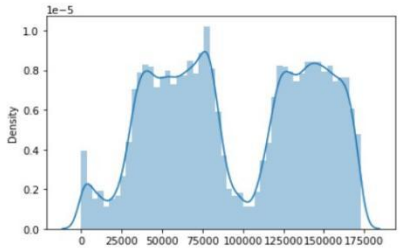


Figure 7.5

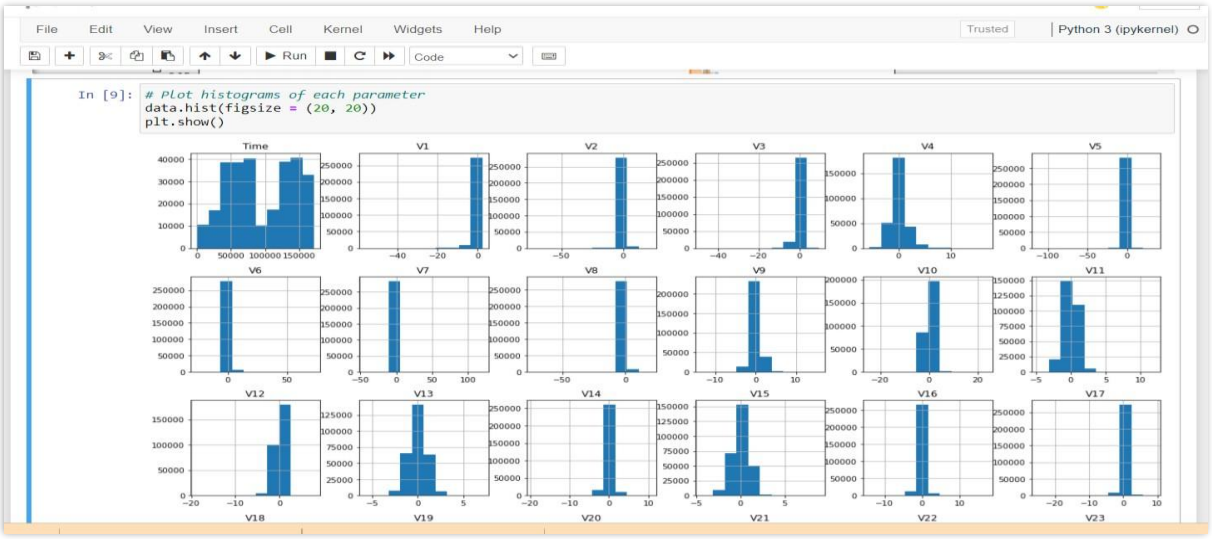


Figure 7.6

7.2.3 MODULE 2: DATA PRE-PROCESSING

Organize your selected data by formatting, cleaning and sampling from it. Three common data pre-processing steps are:

1) Formatting:

The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

2)Cleaning

Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be removed from the data entirely.

3)Sampling:

There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

```
In [55]: legit_sample = Valid.sample(n=492)

In [56]: new_dataset = pd.concat([legit_sample, Fraud], axis=0)

In [57]: new_dataset.head()

Out[57]:
```

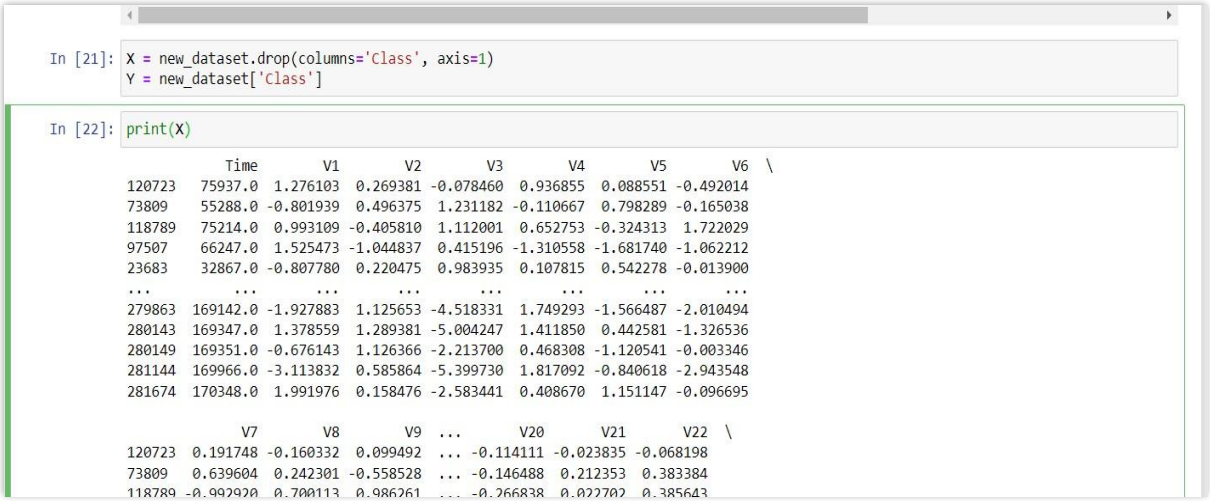
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V2
235660	148485.0	-3.489410	-1.494295	-4.679645	-0.716376	-12.131628	6.304623	9.590051	-3.001867	0.977404	...	0.268608	0.629566	1.370557	0.33772
60191	49202.0	1.159825	0.148771	0.241111	1.323326	0.007479	0.022971	0.082106	0.002104	0.314043	...	-0.115887	-0.129678	-0.120638	-0.27738
280231	169402.0	-1.908444	1.033402	2.007459	0.984352	-0.647625	1.097376	-0.316539	0.696990	1.306363	...	-0.222912	0.097988	-0.347013	0.66228
193023	129958.0	2.073751	0.030703	-1.297472	0.367036	-0.007684	-1.366135	0.285157	-0.429458	0.537215	...	0.250113	0.909418	-0.003635	0.09432
78960	57789.0	1.334397	0.452994	-0.716499	0.723441	0.948683	0.088408	0.475582	-0.154211	-0.258501	...	-0.056510	-0.037388	-0.293760	-1.29432

5 rows × 31 columns

Figure 7.7

7.2.4 MODULE 3: FEATURE EXTRATION

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.



```
In [21]: X = new_dataset.drop(columns='Class', axis=1)
         Y = new_dataset['Class']

In [22]: print(X)

             Time        V1        V2        V3        V4        V5        V6  \
120723   75937.0  1.276103  0.269381 -0.078460  0.936855  0.088551 -0.492014
73809    55288.0 -0.801939  0.496375  1.231182 -0.110667  0.798289 -0.165038
118789   75214.0  0.993109 -0.405810  1.112001  0.652753 -0.324313  1.722029
97507    66247.0  1.525473 -1.044837  0.415196 -1.310558 -1.681740 -1.062212
23683    32867.0 -0.807780  0.220475  0.983935  0.107815  0.542278 -0.013900
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

               V7        V8        V9  ...       V20       V21       V22  \
120723  0.191748 -0.160332  0.099492  ... -0.114111 -0.023835 -0.068198
73809   0.639604  0.242301 -0.558528  ... -0.146488  0.212353  0.383384
118789 -0.992920  0.700113  0.986261  ... -0.266838  0.022702  0.385643
```

Figure 7.8

7.2.5 MODULE 4: EVALUATION MODEL

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation.

To avoid over fitting, both methods use a test set (not seen by the model) to evaluate model performance. Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs. Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

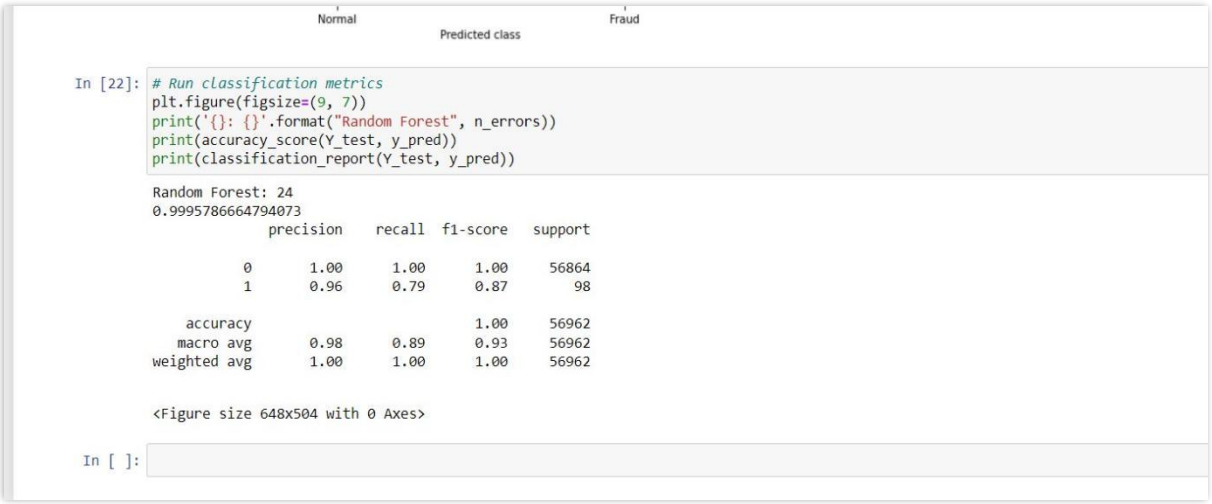

Normal
Fraud
Predicted class
In [22]:
# Run classification metrics
plt.figure(figsize=(9, 7))
print('{}: {}'.format("Random Forest", n_errors))
print(accuracy_score(Y_test, y_pred))
print(classification_report(Y_test, y_pred))
Random Forest: 24
0.9995786664794073
precision recall f1-score support
0 1.00 1.00 1.00 56864
1 0.96 0.79 0.87 98
accuracy 1.00 56962
macro avg 0.98 0.89 0.93 56962
weighted avg 1.00 1.00 1.00 56962
<Figure size 648x504 with 0 Axes>
In [ ]:

Figure 7.9

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

The proposed unsupervised random forest algorithm reduces the number of fraud transactions. There are several experiments performed using random forest algorithm. The obtained results ensured that the number of fraud transactions is greatly reduced. This improves more secure transactions through online and makes the system more accurate**.**

1. Random Forest = 99%

2. Logistic Regression =89%

3. Support Vector Machine =94%

8.1.1 FUTURE WORK

It is evident from the above review that several machine learning algorithms are used to detect fraud, but the findings are not satisfactory. As a result, we'd like to use deep learning algorithms to reliably detect credit card fraud.

CHAPTER 9

APPENDIX

9.1 RANDOM FOREST

9.1.1 CODE

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
data = pd.read_csv(r"C:\Users\sound\svmpro\creditcard.csv")
# Start exploring the dataset
print(data.columns)
data.head()
# Print the shape of the data
# data = data.sample(frac=0.1, random_state = 48)
print(data.shape)
print(data.describe())
# distribution of Amount
amount = [data['Amount'].values]
sns.distplot(amount)
# distribution of Time
```

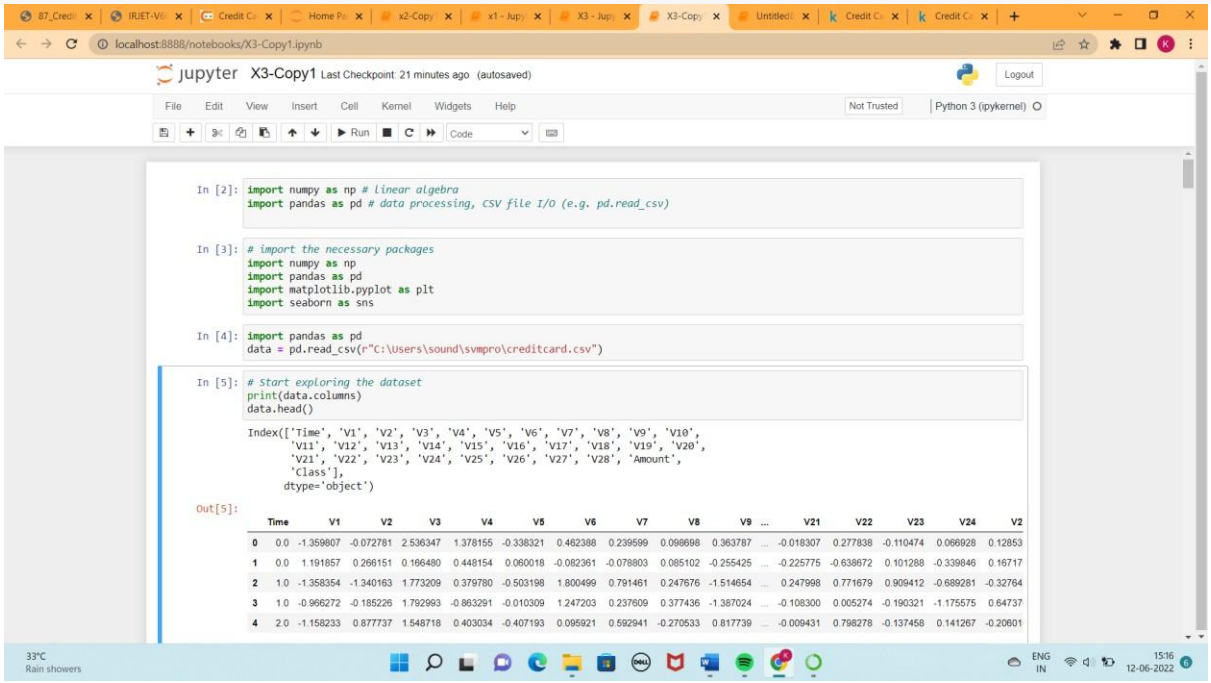
```
time = data['Time'].values
sns.distplot(time)
from matplotlib import gridspec
# distribution of anomalous features
features = data.iloc[:,0:28].columns

plt.figure(figsize=(12,28*4))
gs = gridspec.GridSpec(28, 1)
for i, c in enumerate(data[features]):
    ax = plt.subplot(gs[i])
    sns.distplot(data[c][data.Class == 1], bins=50)
    sns.distplot(data[c][data.Class == 0], bins=50)
    ax.set_xlabel('')
    ax.set_title('histogram of feature: ' + str(c))
plt.show
# Plot histograms of each parameter
data.hist(figsize = (20, 20))
plt.show()
# Determine number of fraud cases in dataset

Fraud = data[data['Class'] == 1]
Valid = data[data['Class'] == 0]

outlier_fraction = len(Fraud)/float(len(Valid))
print(outlier_fraction)

print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
```

```
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
print("Amount details of fradulent transacation")
Fraud.Amount.describe()
print("Amount details of valid transaction")
Valid.Amount.describe()
legit_sample = Valid.sample(n=492)
new_dataset = pd.concat([legit_sample, Fraud], axis=0)
new_dataset.head()
new_dataset.tail()
new_dataset['Class'].value_counts()
new_dataset.groupby('Class').mean()
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
print(X)
print(Y)
# Correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
#getting just the values for the sake of processing (its a numpy array with no columns)
X_data=X.values
Y_data=Y.values
X_data
# 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

X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size =
0.2, random_state = 42)

from sklearn.metrics import classification_report,
accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef

from sklearn.metrics import confusion_matrix

# Building the Random Forest Classifier (RANDOM FOREST)

from sklearn.ensemble import RandomForestClassifier

# random forest model creation

rfc = RandomForestClassifier()

rfc.fit(X_train,Y_train)

# predictions

y_pred = rfc.predict(X_test)

#Evaluating the classifier

#printing every score of the classifier

#scoring in any thing

from sklearn.metrics import classification_report,
accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef

from sklearn.metrics import confusion_matrix

n_outliers = len(Fraud)

n_errors = (y_pred != Y_test).sum()

print("The model used is Random Forest classifier")

acc= accuracy_score(Y_test,y_pred)

print("The accuracy is {}".format(acc))

prec= precision_score(Y_test,y_pred)

print("The precision is {}".format(prec))
```

```
rec= recall_score(Y_test,y_pred)
print("The recall is {}".format(rec))
f1= f1_score(Y_test,y_pred)
print("The F1-Score is {}".format(f1))
MCC=matthews_corrcoef(Y_test,y_pred)
print("The Matthews correlation coefficient is {}".format(MCC))
# Run classification metrics
plt.figure(figsize=(9, 7))
print('{}: {}'.format("Random Forest", n_errors))
print(accuracy_score(Y_test, y_pred))
print(classification_report(Y_test, y_pred))
```

9.1.2 OUTPUT



jupyter X3-Copy1 Last Checkpoint: 21 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

In [3]:
```
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
import pandas as pd
data = pd.read_csv(r"C:\Users\sound\svmpro\creditcard.csv")
```

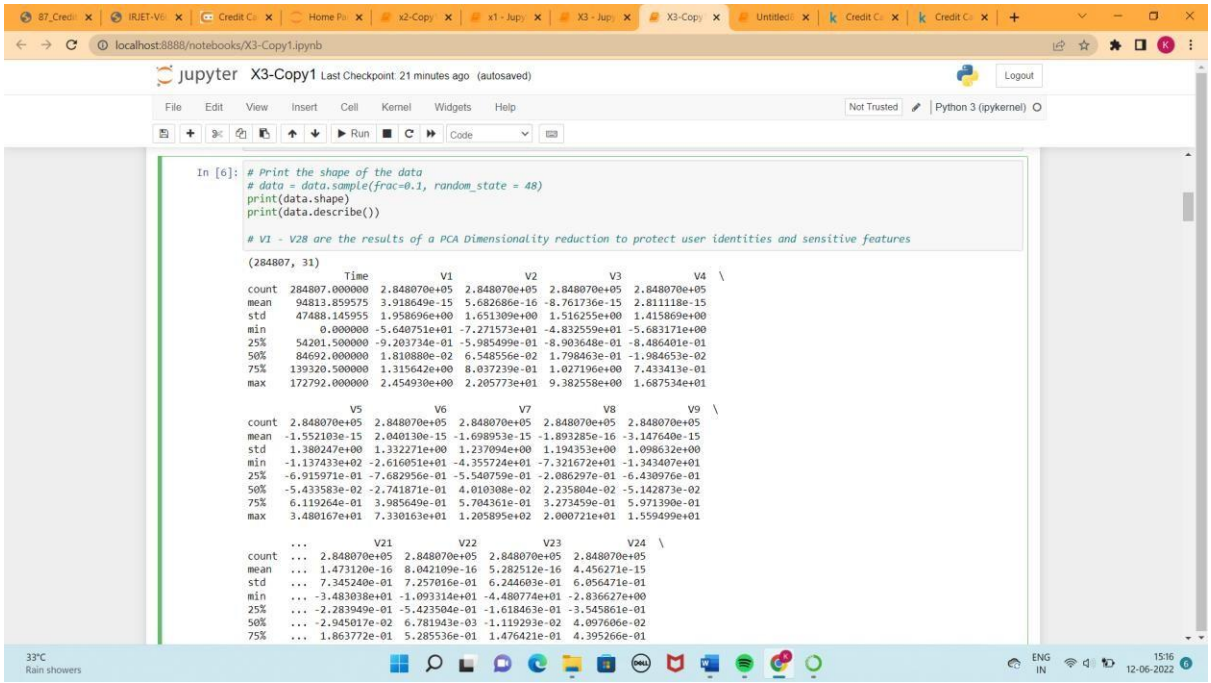
In [5]:
```
# Start exploring the dataset
print(data.columns)
data.head()
```

```
Index(['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',
       'Class'],
      dtype='object')
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601

33°C Rain showers | ENG IN | 15:16 12-06-2022



jupyter X3-Copy1 Last Checkpoint: 21 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

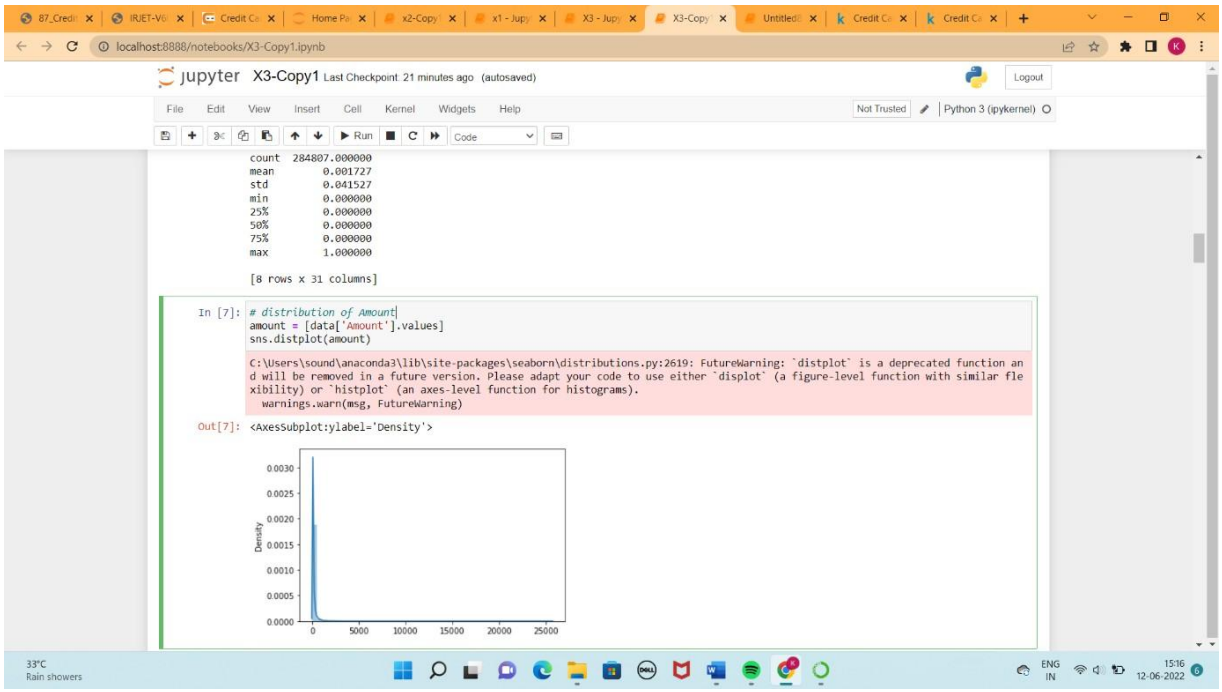
Not Trusted | Python 3 (ipykernel)

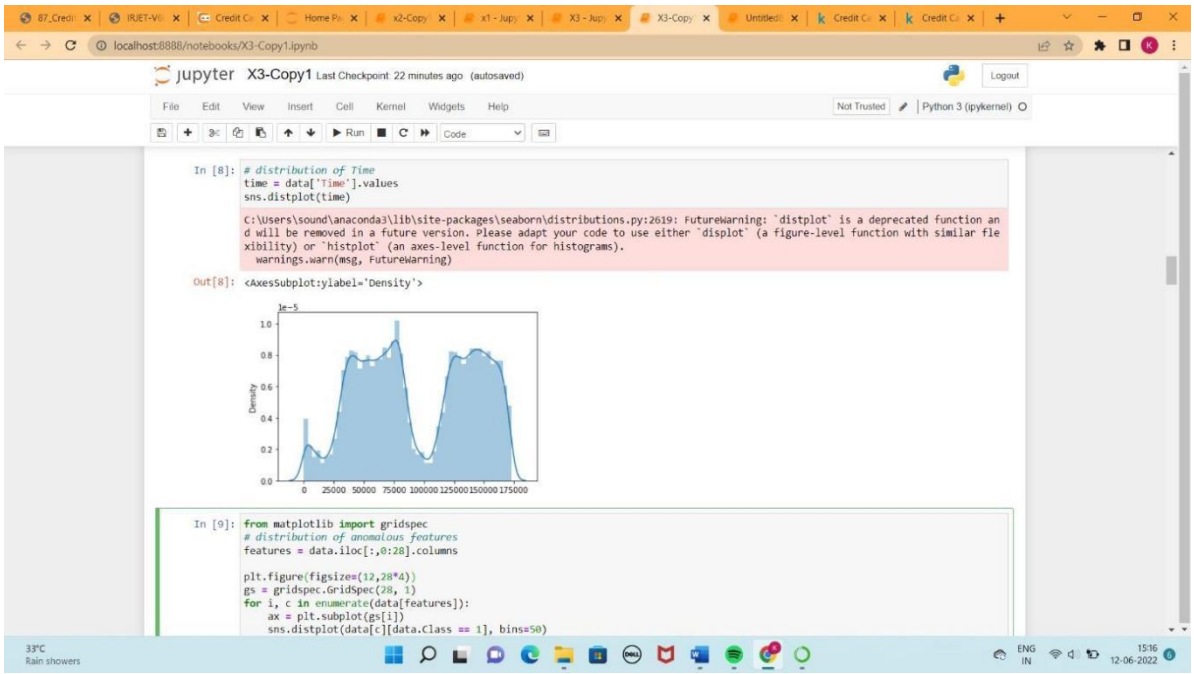
Run Code

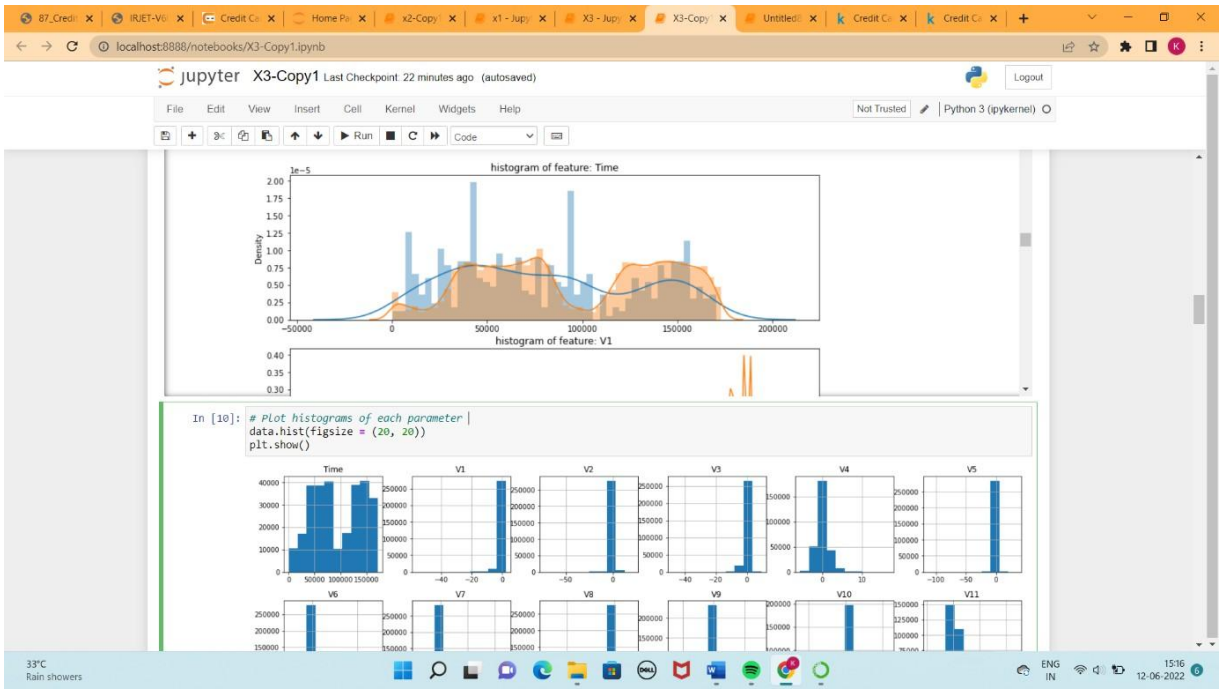
In [6]:
```
# Print the shape of the data
# data = data.sample(frac=0.1, random_state = 48)
print(data.shape)
print(data.describe())

# V1 - V28 are the results of a PCA Dimensionality reduction to protect user identities and sensitive features
```

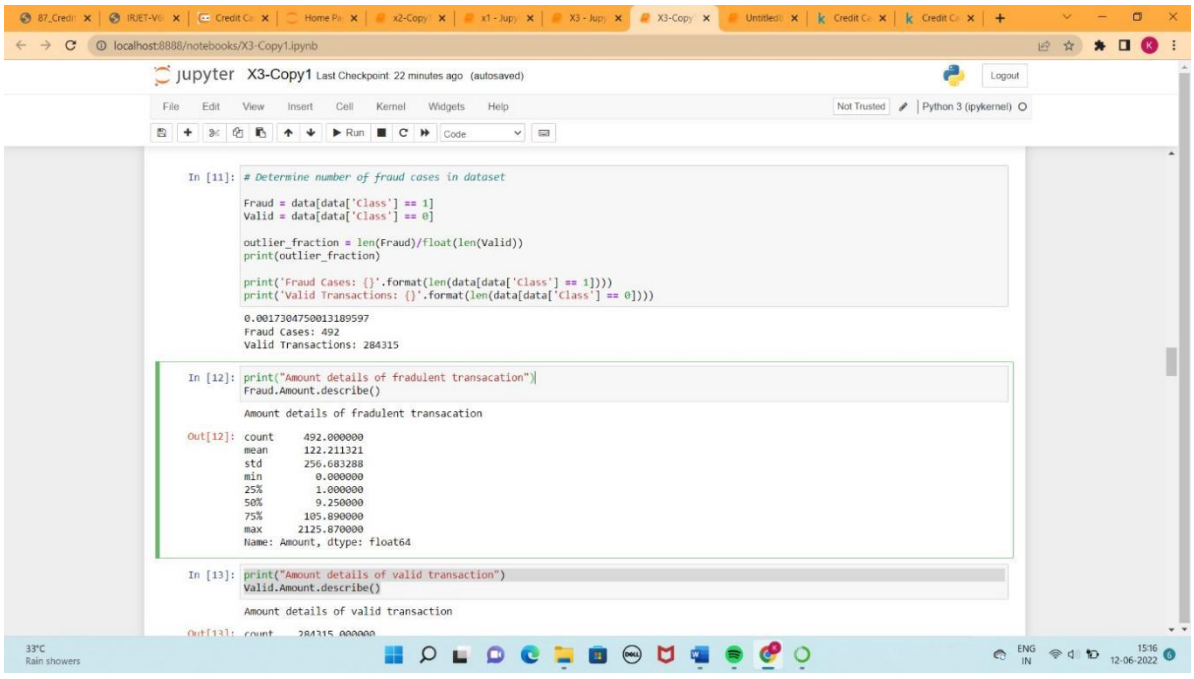
```
(284807, 31)
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  3.918649e-15  5.682686e-16 -8.761736e-15  2.811118e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean  -1.552103e-15  2.040130e-15 -1.698953e-15 -1.893285e-16 -3.147640e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.473120e-16  8.042109e-16  5.282512e-16  4.456271e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
```

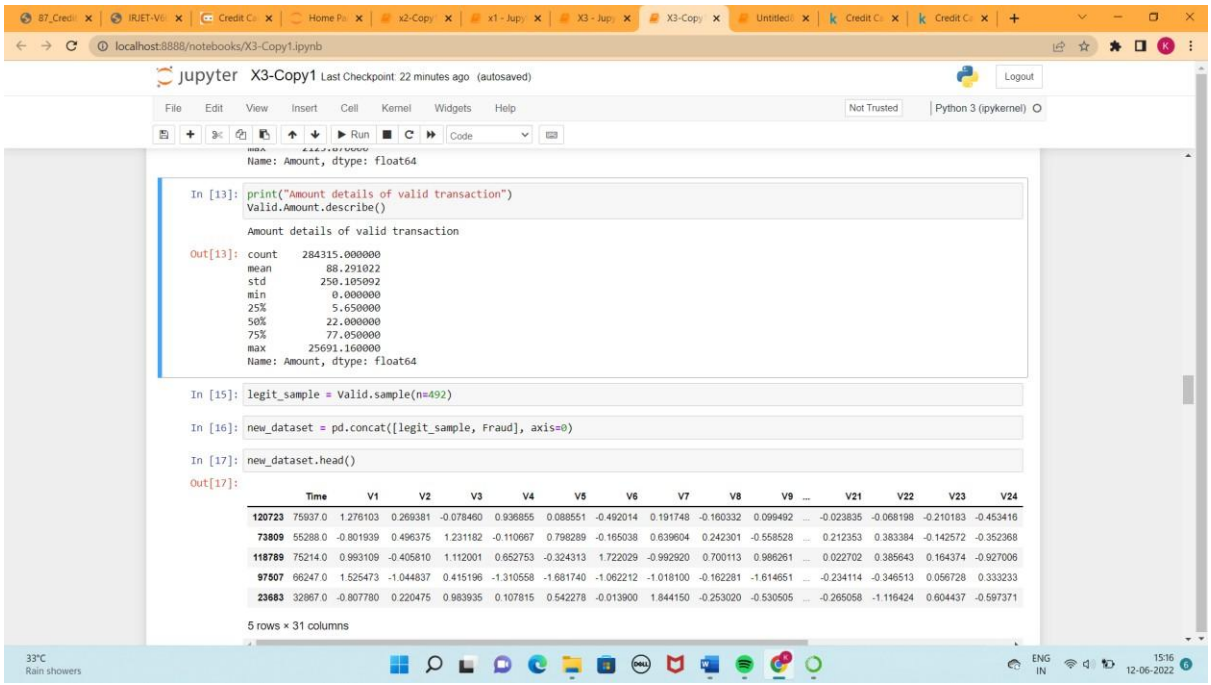
33°C Rain showers | ENG IN | 15:16 12-06-2022


jupyter X3-Copy1 Last Checkpoint: 21 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
count 284807.000000
mean 0.001727
std 0.041527
min 0.000000
25% 0.000000
50% 0.000000
75% 0.000000
max 1.000000
[8 rows x 31 columns]
In [7]:
# distribution of Amount
amount = [data['Amount'].values]
sns.distplot(amount)
C:\Users\sound\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
Out[7]: <AxesSubplot:ylabel='Density'>
Density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
5000
10000
15000
20000
25000
33°C
Rain showers
ENG
IN
15:16
12-06-2022


jupyter X3-Copy1 Last Checkpoint: 22 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [8]:
# distribution of Time
time = data['Time'].values
sns.distplot(time)
C:\Users\sound\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
Out[8]: <AxesSubplot:ylabel='Density'>
1e−5
Density
1.0
0.8
0.6
0.4
0.2
0.0
0
25000
50000
75000
100000
125000
150000
175000
In [9]:
from matplotlib import gridspec
# distribution of anomalous features
features = data.iloc[:,0:28].columns
plt.figure(figsize=(12,28*4))
gs = gridspec.GridSpec(28, 1)
for i, c in enumerate(data[features]):
ax = plt.subplot(gs[i])
sns.distplot(data[c][data.Class == 1], bins=50)
33°C
Rain showers
ENG
IN
15:16
12-06-2022


jupyter X3-Copy1 Last Checkpoint: 22 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
histogram of feature: Time
histogram of feature: V1
In [10]:
# Plot histograms of each parameter
data.hist(figsize = (20, 20))
plt.show()
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11


jupyter X3-Copy1 Last Checkpoint: 22 minutes ago (autosaved)
In [11]:
# Determine number of fraud cases in dataset
Fraud = data[data['Class'] == 1]
Valid = data[data['Class'] == 0]
outlier_fraction = len(Fraud)/float(len(Valid))
print(outlier_fraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
0.0017304750013189597
Fraud Cases: 492
Valid Transactions: 284315
In [12]:
print("Amount details of fradulent transacation")
Fraud.Amount.describe()
Amount details of fradulent transacation
Out[12]:
count 492.000000
mean 122.211321
std 256.683288
min 0.000000
25% 1.000000
50% 9.250000
75% 105.890000
max 2125.870000
Name: Amount, dtype: float64
In [13]:
print("Amount details of valid transaction")
Valid.Amount.describe()
Amount details of valid transaction



jupyter X3-Copy1 Last Checkpoint: 22 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

```
Name: Amount, dtype: float64
```

In [13]:
```
print("Amount details of valid transaction")
Valid.Amount.describe()
```

```
Amount details of valid transaction
```

Out[13]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

In [15]:
```
legit_sample = Valid.sample(n=492)
```

In [16]:
```
new_dataset = pd.concat([legit_sample, Fraud], axis=0)
```

In [17]:
```
new_dataset.head()
```

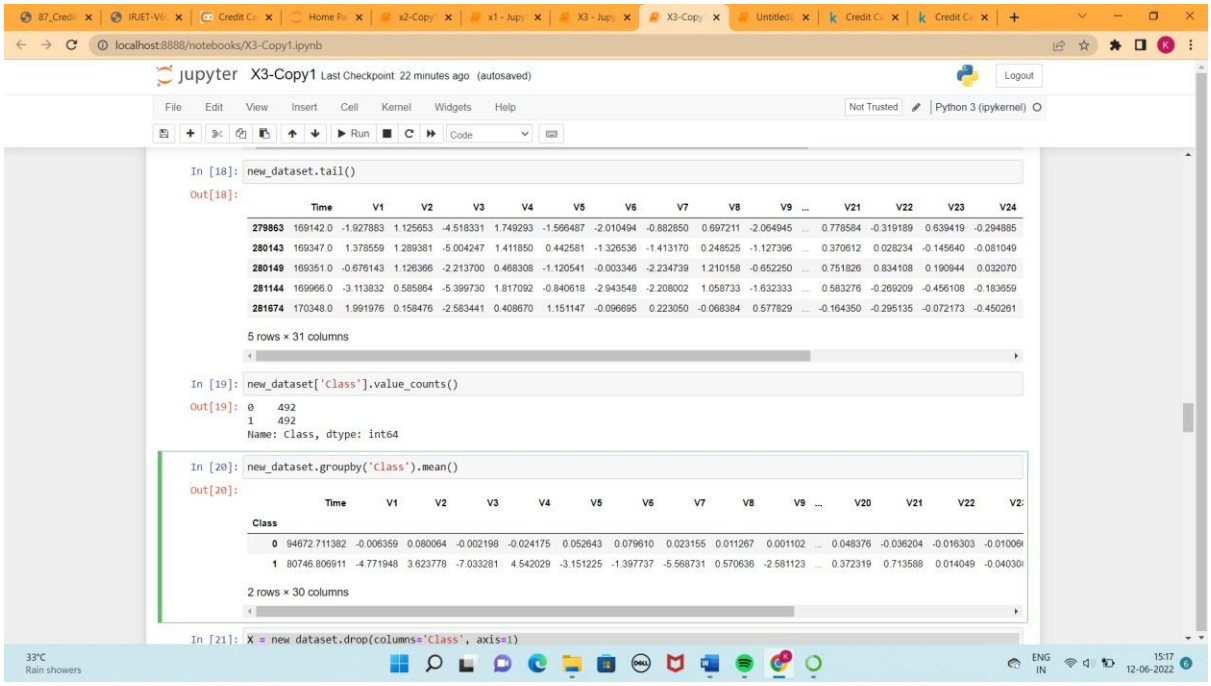
Out[17]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
120723	75937.0	1.276103	0.269381	-0.078460	0.936855	0.088551	-0.492014	0.191748	-0.160332	0.099492	...	-0.023835	-0.068198	-0.210183	-0.453416
73809	55288.0	-0.801939	0.496375	1.231182	-0.110667	0.798289	-0.165038	0.639604	0.242301	-0.558528	...	0.212353	0.383384	-0.142572	-0.352368
118789	75214.0	0.993109	-0.405810	1.112001	0.652753	-0.324313	1.722029	-0.992920	0.700113	0.986261	...	0.022702	0.385643	0.164374	-0.927006
97507	66247.0	1.525473	-1.044837	0.415196	-1.310558	-1.681740	-1.062212	-1.018100	-0.162281	-1.614651	...	-0.234114	-0.346513	0.056728	0.333233
23683	32867.0	-0.807780	0.220475	0.983935	0.107815	0.542278	-0.013900	1.844150	-0.253020	-0.530505	...	-0.265058	-1.116424	0.604437	-0.597371

5 rows × 31 columns

33°C Rain showers

ENG IN 15:16 12-06-2022



jupyter X3-Copy1 Last Checkpoint: 22 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

In [18]:
```
new_dataset.tail()
```

Out[18]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.319189	0.639419	-0.294885
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.028234	-0.145640	-0.081049
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.834108	0.190944	0.032070
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.269209	-0.456108	-0.183659
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.295135	-0.072173	-0.450261

5 rows × 31 columns

In [19]:
```
new_dataset['Class'].value_counts()
```

Out[19]:
```
0    492
1    492
Name: Class, dtype: int64
```

In [20]:
```
new_dataset.groupby('Class').mean()
```

Out[20]:

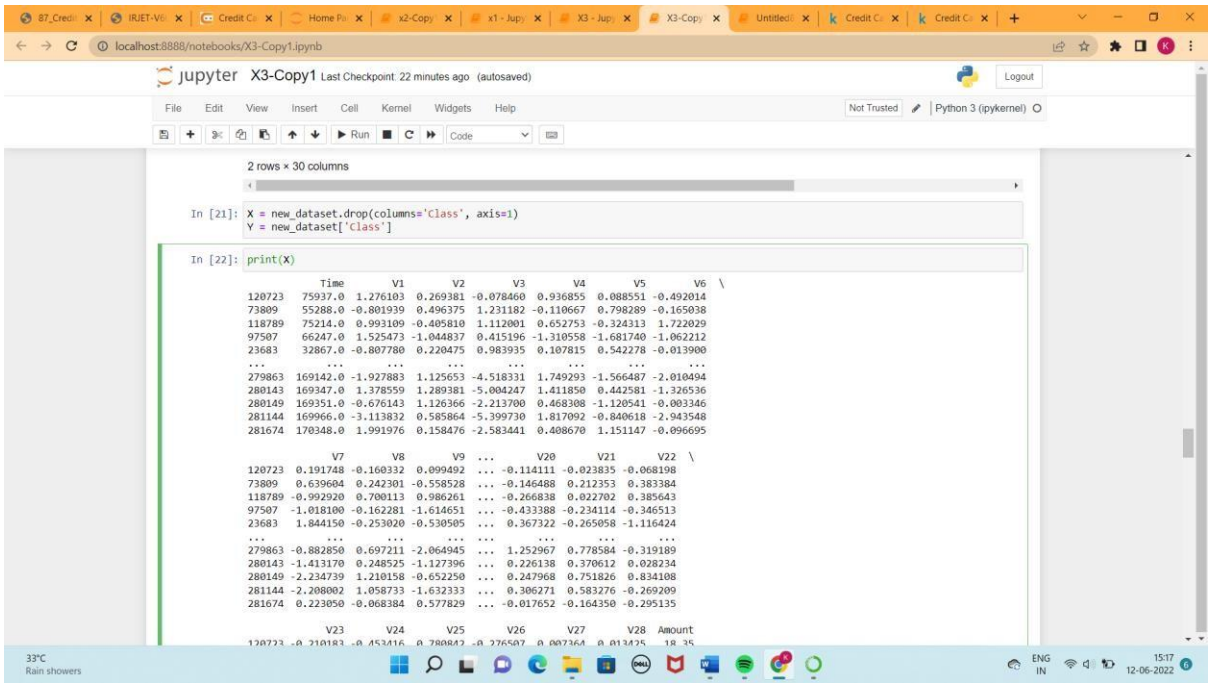
Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V2:
0	94672.711382	-0.006359	0.080064	-0.002198	-0.024175	0.052643	0.079610	0.023155	0.011267	0.001102	...	0.048376	-0.036204	-0.016303	-0.01006
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.04030

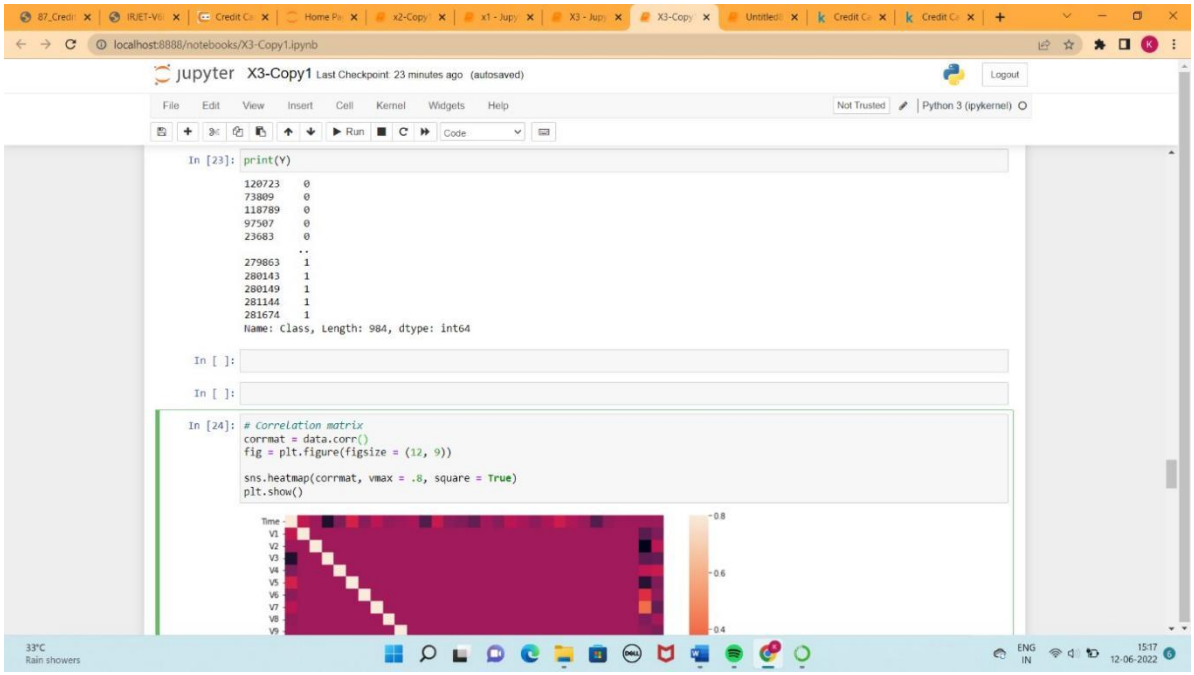
2 rows × 30 columns

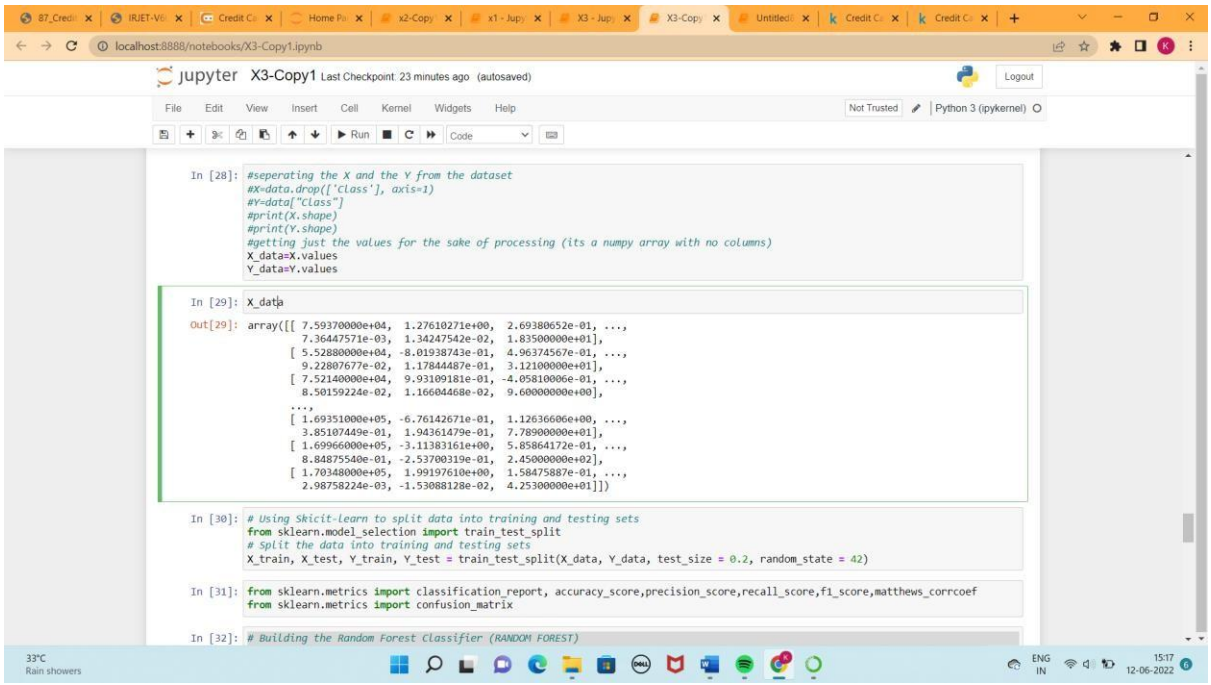
In [21]:
```
X = new_dataset.drop(columns='Class', axis=1)
```

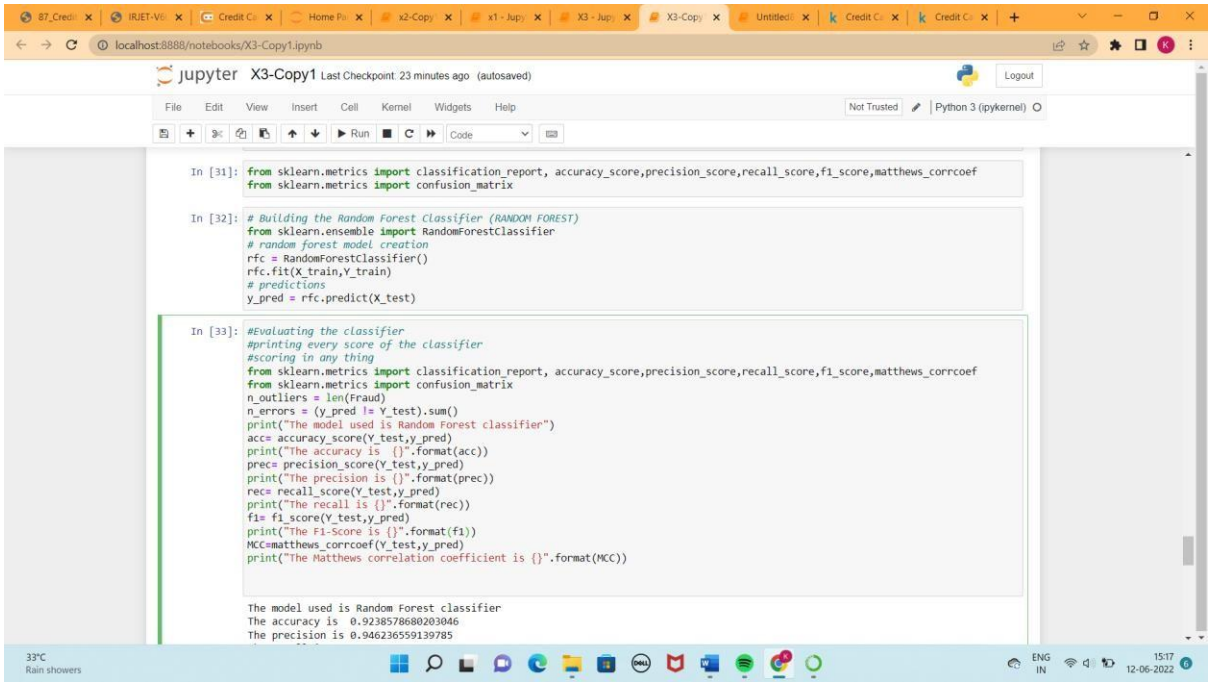
33°C Rain showers

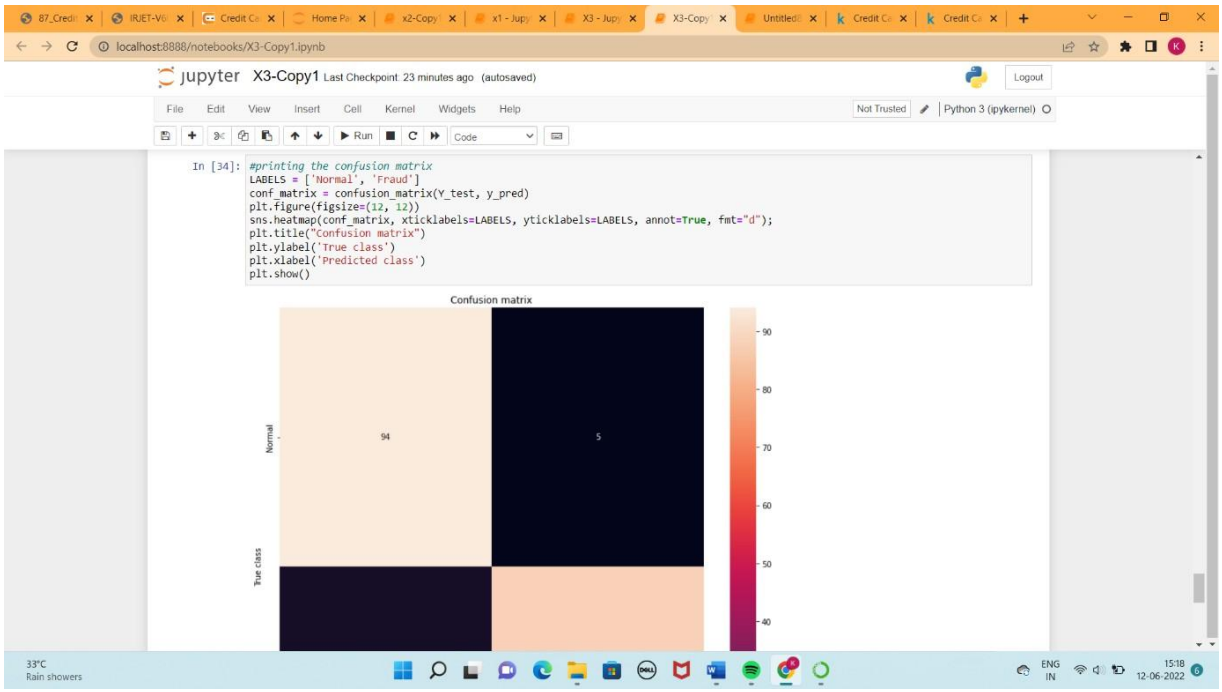
ENG IN 15:17 12-06-2022

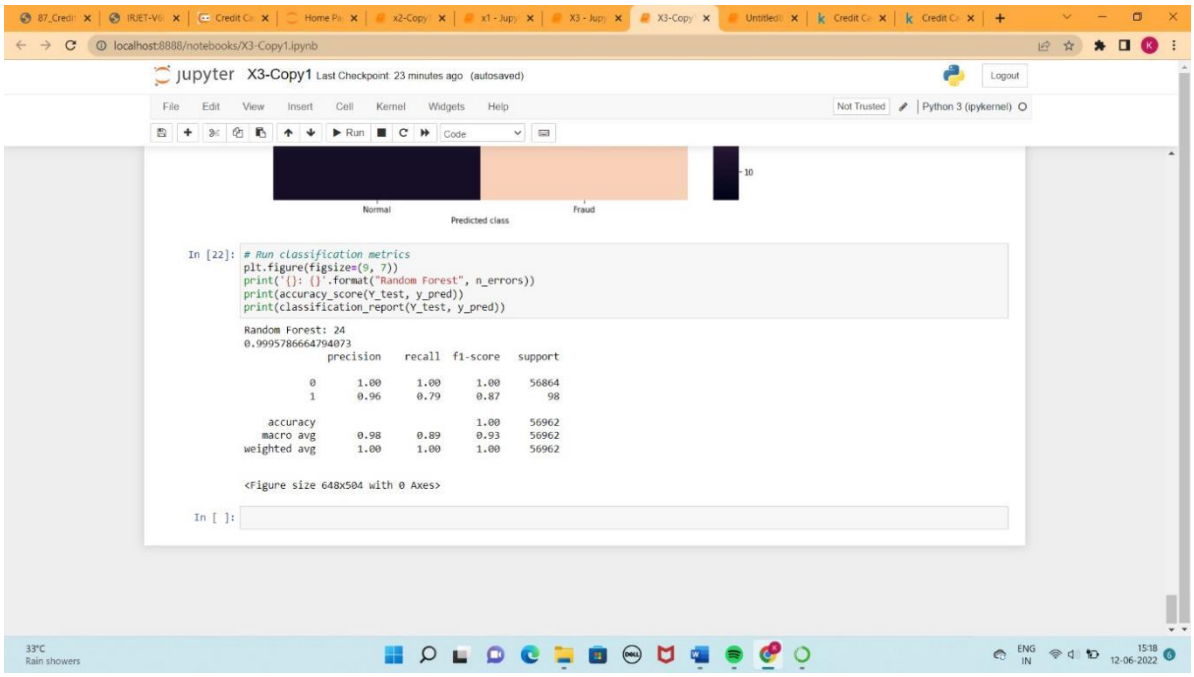

jupyter X3-Copy1 Last Checkpoint: 22 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
2 rows × 30 columns
In [21]:
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
In [22]:
print(X)
Time V1 V2 V3 V4 V5 V6 \
120723 75937.0 1.276103 0.269381 -0.078460 0.936855 0.088551 -0.492014
73809 55288.0 -0.801939 0.496375 1.231182 -0.110667 0.798289 -0.165038
118789 75214.0 0.993109 -0.405810 1.112001 0.652753 -0.324313 1.722029
97507 66247.0 1.525473 -1.044837 0.415196 -1.310558 -1.681740 -1.062212
23683 32867.0 -0.807780 0.220475 0.983935 0.107815 0.542278 -0.013900
... ...
279863 169142.0 -1.927883 1.125653 -4.518331 1.749293 -1.566487 -2.010494
280143 169347.0 1.378559 1.289381 -5.004247 1.411850 0.442581 -1.326536
280149 169351.0 -0.676143 1.126366 -2.213700 0.468308 -1.120541 -0.003346
281144 169966.0 -3.113832 0.585864 -5.399730 1.817092 -0.840618 -2.943548
281674 170348.0 1.991976 0.158476 -2.583441 0.408670 1.151147 -0.096695
V7 V8 V9 ... V20 V21 V22 \
120723 0.191748 -0.160332 0.099492 ... -0.114111 -0.023835 -0.068198
73809 0.639604 0.242301 -0.558528 ... -0.146488 0.212353 0.383384
118789 -0.992920 0.700113 0.986261 ... -0.266838 0.022702 0.385643
97507 -1.018100 -0.162281 -1.614651 ... -0.433388 -0.234114 -0.346513
23683 1.844150 -0.253020 -0.530505 ... 0.367322 -0.265058 -1.116424
... ...
279863 -0.882850 0.697211 -2.064945 ... 1.252967 0.778584 -0.319189
280143 -1.413170 0.248525 -1.127396 ... 0.226138 0.370612 0.028234
280149 -2.234739 1.210158 -0.652250 ... 0.247968 0.751826 0.834108
281144 -2.208002 1.058733 -1.632333 ... 0.306271 0.583276 -0.269209
281674 0.223050 -0.068384 0.577829 ... -0.017652 -0.164350 -0.295135
V23 V24 V25 V26 V27 V28 Amount
33°C
Rain showers
ENG
IN
15:17
12-06-2022


jupyter X3-Copy1 Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [23]:
print(Y)
120723 0
73809 0
118789 0
97507 0
23683 0
..
279863 1
280143 1
280149 1
281144 1
281674 1
Name: Class, Length: 984, dtype: int64
In [ ]:
In [ ]:
In [24]:
# Correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
0.8
0.6
0.4
33°C
Rain showers
ENG
IN
15:17
12-06-2022


jupyter X3-Copy1 Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [28]:
#seperating 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)
X_data=X.values
Y_data=Y.values
In [29]:
X_data
Out[29]:
array([[7.59370000e+04, 1.27610271e+00, 2.69380652e-01, ...,
7.36447571e-03, 1.34247542e-02, 1.83500000e+01],
[5.52880000e+04, -8.01938743e-01, 4.96374567e-01, ...,
9.22807677e-02, 1.17844487e-01, 3.12100000e+01],
[7.52140000e+04, 9.93109181e-01, -4.05810006e-01, ...,
8.50159224e-02, 1.16604468e-02, 9.60000000e+00],
...,
[1.69351000e+05, -6.76142671e-01, 1.12636606e+00, ...,
3.85107449e-01, 1.94361479e-01, 7.78900000e+01],
[1.69966000e+05, -3.11383161e+00, 5.85864172e-01, ...,
8.84875540e-01, -2.53700319e-01, 2.45000000e+02],
[1.70348000e+05, 1.99197610e+00, 1.58475887e-01, ...,
2.98758224e-03, -1.53088128e-02, 4.25300000e+01]])
In [30]:
# 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
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size = 0.2, random_state = 42)
In [31]:
from sklearn.metrics import classification_report, accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef
from sklearn.metrics import confusion_matrix
In [32]:
Building the Random Forest Classifier (RANDOM FOREST)
33°C
Rain showers
ENG
IN
15:17
12-06-2022


jupyter X3-Copy1 Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [31]:
from sklearn.metrics import classification_report, accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef
from sklearn.metrics import confusion_matrix
In [32]:
# Building the Random Forest Classifier (RANDOM FOREST)
from sklearn.ensemble import RandomForestClassifier
# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(X_train,Y_train)
# predictions
y_pred = rfc.predict(X_test)
In [33]:
#Evaluating the classifier
#printing every score of the classifier
#scoring in any thing
from sklearn.metrics import classification_report, accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef
from sklearn.metrics import confusion_matrix
n_outliers = len(Fraud)
n_errors = (y_pred != Y_test).sum()
print("The model used is Random Forest classifier")
acc= accuracy_score(Y_test,y_pred)
print("The accuracy is {}".format(acc))
prec= precision_score(Y_test,y_pred)
print("The precision is {}".format(prec))
rec= recall_score(Y_test,y_pred)
print("The recall is {}".format(rec))
f1= f1_score(Y_test,y_pred)
print("The F1-Score is {}".format(f1))
MCC=matthews_corrcoef(Y_test,y_pred)
print("The Matthews correlation coefficient is {}".format(MCC))
The model used is Random Forest classifier
The accuracy is 0.9238578680203046
The precision is 0.946236559139785
33°C
Rain showers
ENG
IN
15:17
12-06-2022


jupyter X3-Copy1 Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [34]:
#printing the confusion matrix
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(Y_test, y_pred)
plt.figure(figsize=(12, 12))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
Confusion matrix
Normal
Fraud
True class
94
5
90
80
70
60
50
40


jupyter X3-Copy1 Last Checkpoint: 23 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
10
Normal
Fraud
Predicted class
In [22]:
# Run classification metrics
plt.figure(figsize=(9, 7))
print('{}: {}'.format("Random Forest", n_errors))
print(accuracy_score(Y_test, y_pred))
print(classification_report(Y_test, y_pred))
Random Forest: 24
0.9995786664794073
precision recall f1-score support
0 1.00 1.00 1.00 56864
1 0.96 0.79 0.87 98
accuracy 1.00 56962
macro avg 0.98 0.89 0.93 56962
weighted avg 1.00 1.00 1.00 56962
<Figure size 648x504 with 0 Axes>
In [ ]:

9.2 LOGISTIC REGRESSION

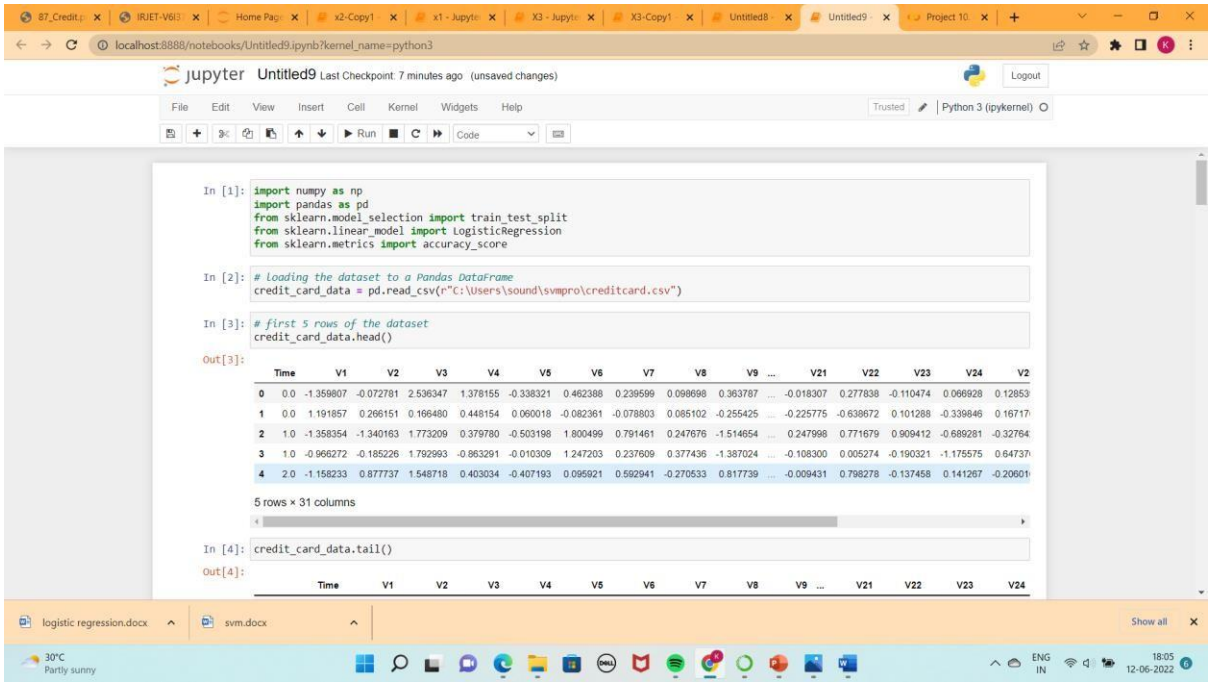
9.2.1 CODE

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('credit_data.csv')

# first 5 rows of the dataset
credit_card_data.head()

credit_card_data.tail()

# dataset informations
credit_card_data.info()

# checking the number of missing values in each column
credit_card_data.isnull().sum()

# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()

# separating the data for analysis
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]

print(legit.shape)
print(fraud.shape)

# statistical measures of the data
legit.Amount.describe()
```

```
fraud.Amount.describe()

# compare the values for both transactions
credit_card_data.groupby('Class').mean()

legit_sample = legit.sample(n=492)

new_dataset = pd.concat([legit_sample, fraud], axis=0)

new_dataset.head()

new_dataset.tail()

new_dataset['Class'].value_counts()

new_dataset.groupby('Class').mean()

X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']

print(X)

print(Y)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=
Y, random_state=2)

print(X.shape, X_train.shape, X_test.shape)

model = LogisticRegression()

# training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)

# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy on Training data : ', training_data_accuracy)

# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test

print('Accuracy score on Test Data : ', test_data_accuracy)
```

9.2.2 OUTPUT



jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score

In [2]: # loading the dataset to a Pandas DataFrame
        credit_card_data = pd.read_csv(r"C:\Users\sound\svmpro\creditcard.csv")

In [3]: # first 5 rows of the dataset
        credit_card_data.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601

5 rows × 31 columns

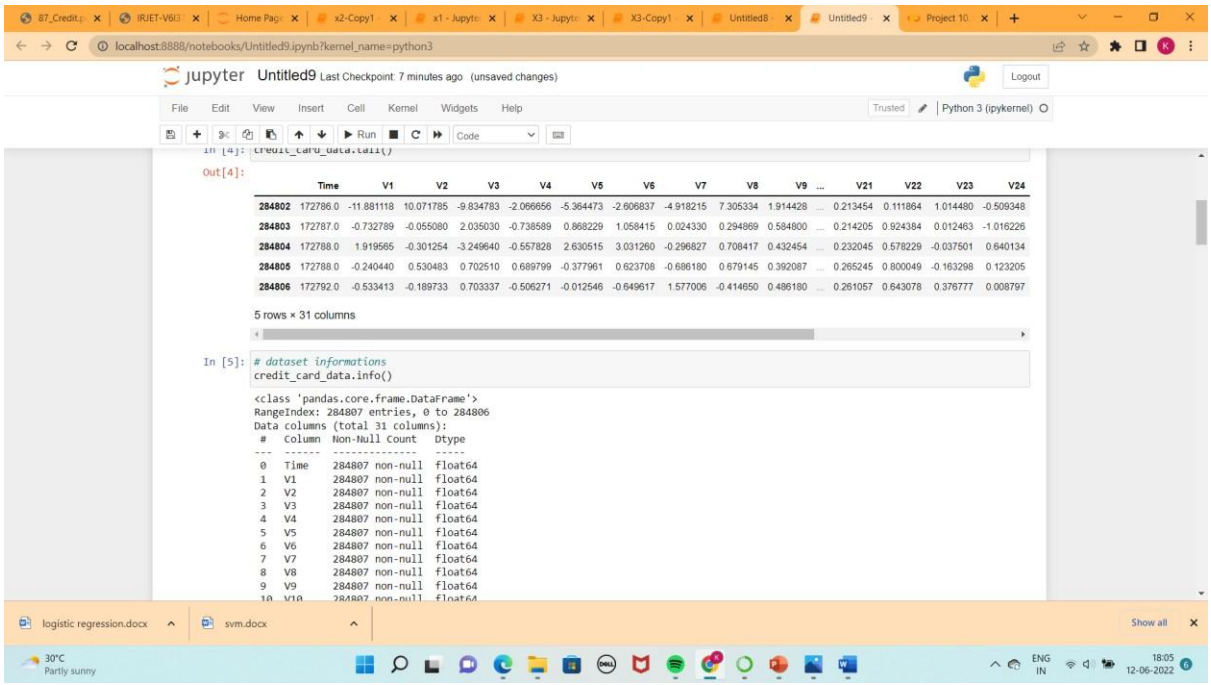
```
In [4]: credit_card_data.tail()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24

logistic regression.docx svm.docx Show all

30°C Partly sunny ENG IN 18:05 12-06-2022



jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797

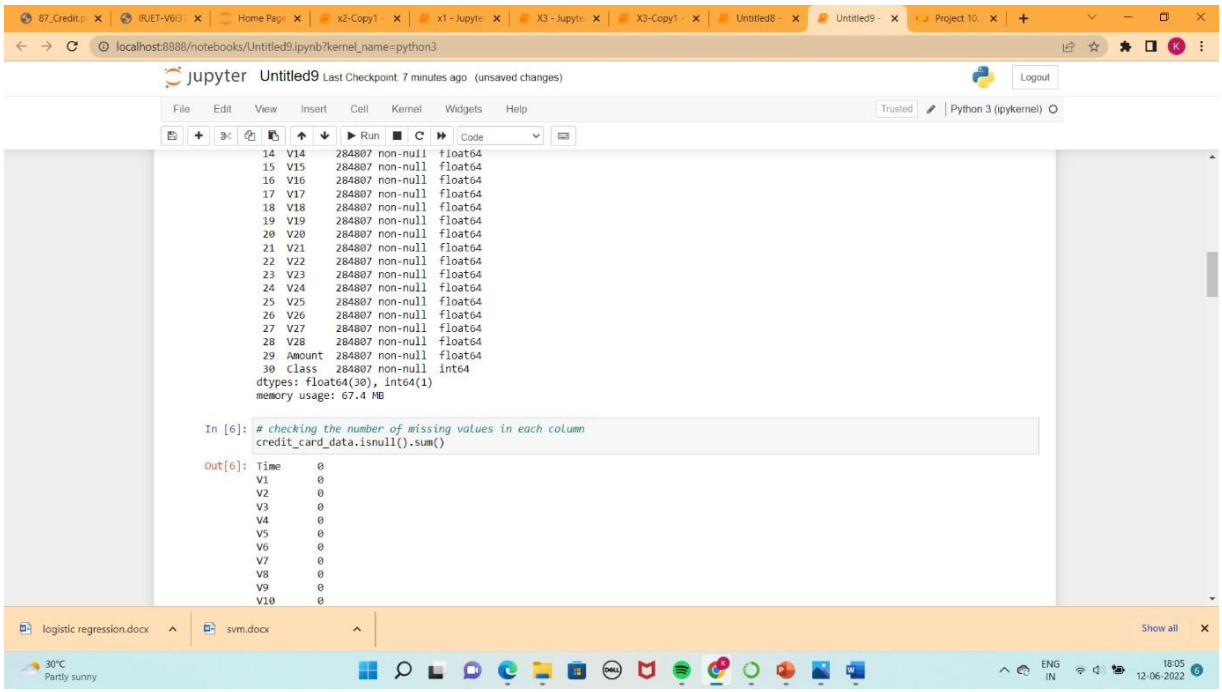
5 rows × 31 columns

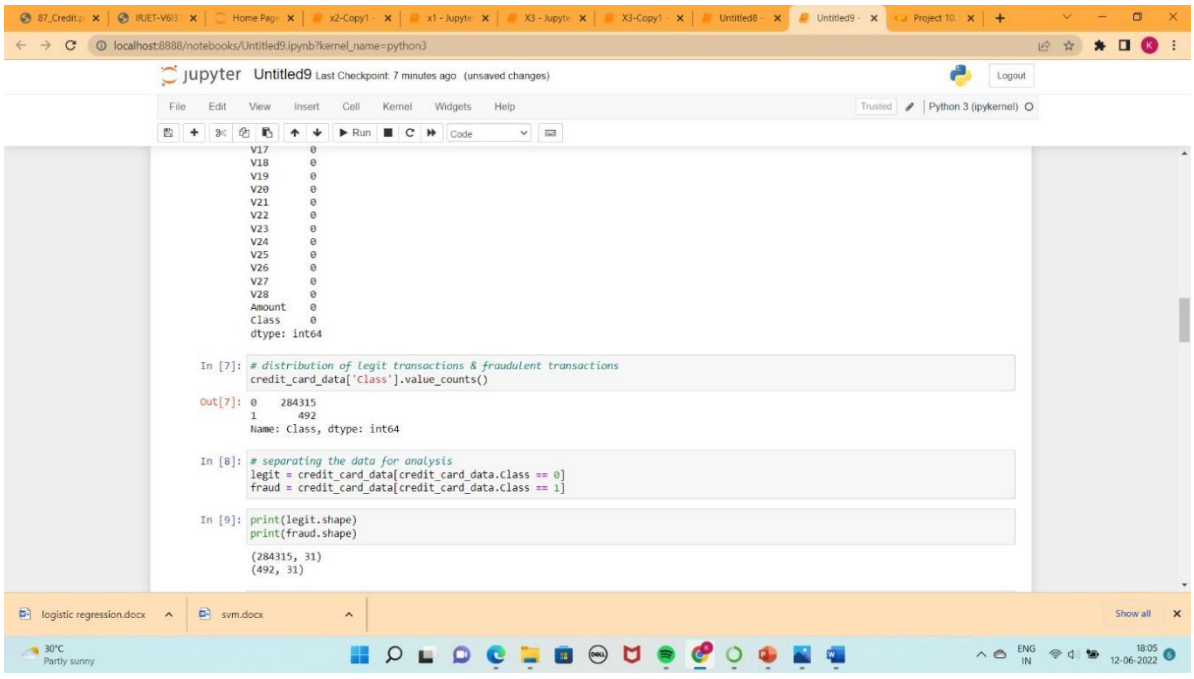
```
In [5]: # dataset informations
        credit_card_data.info()
```

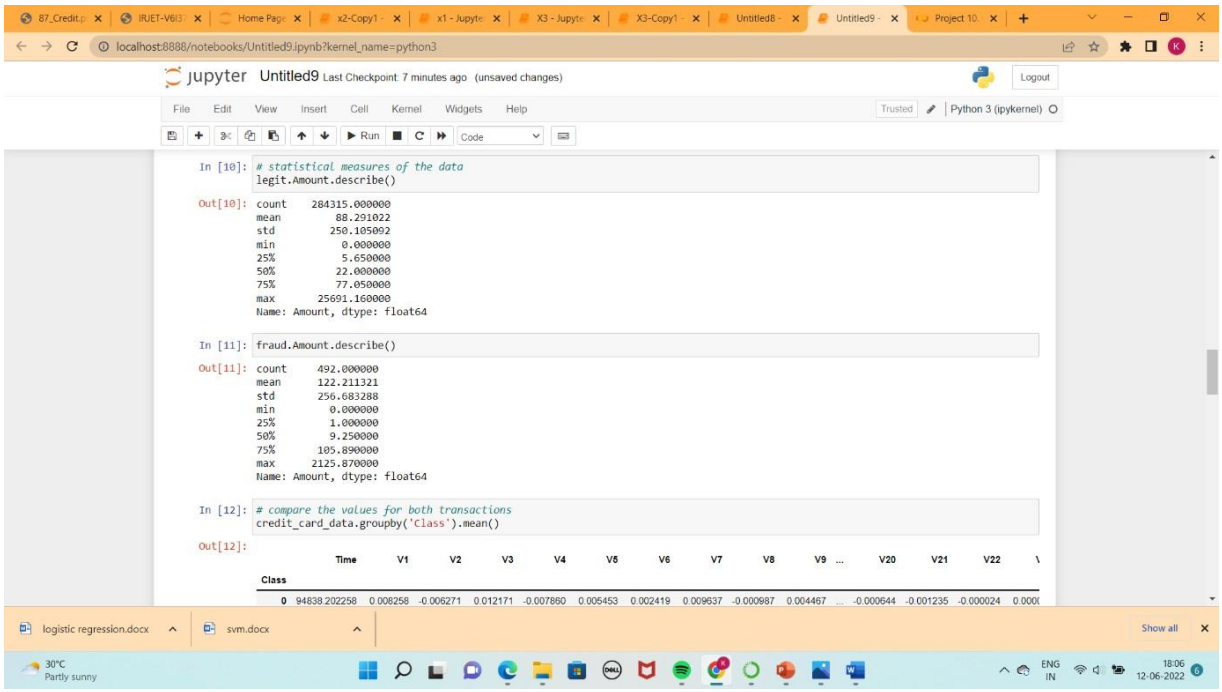
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
```

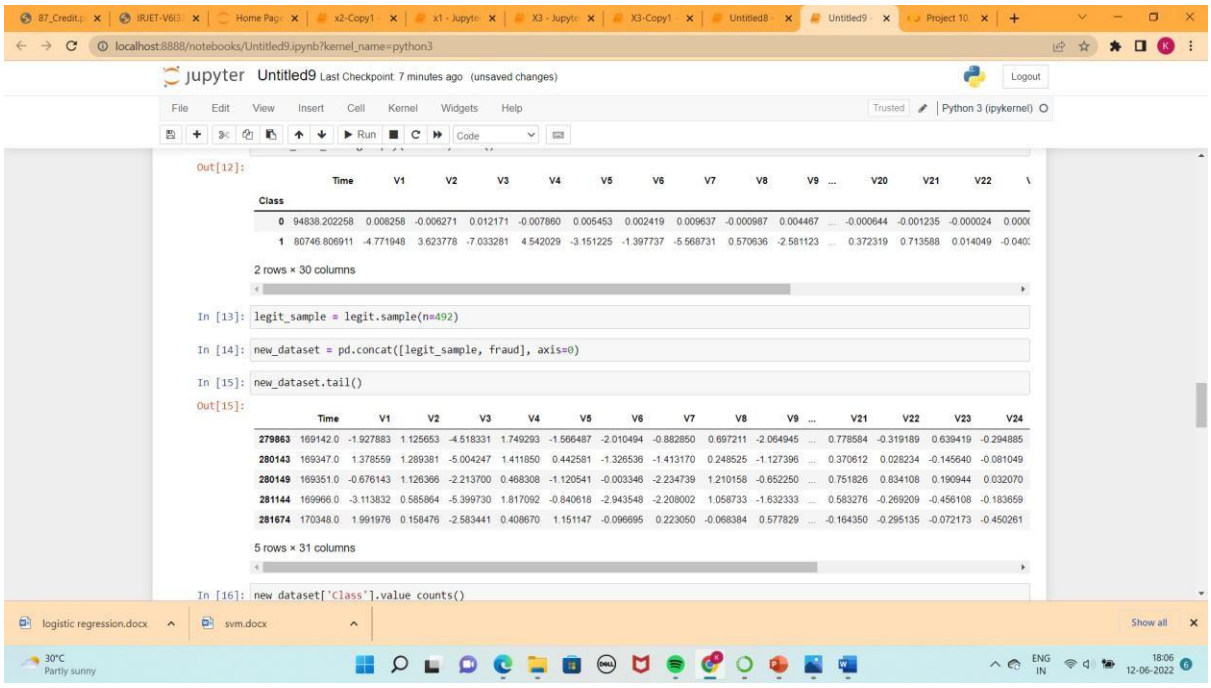
logistic regression.docx svm.docx Show all

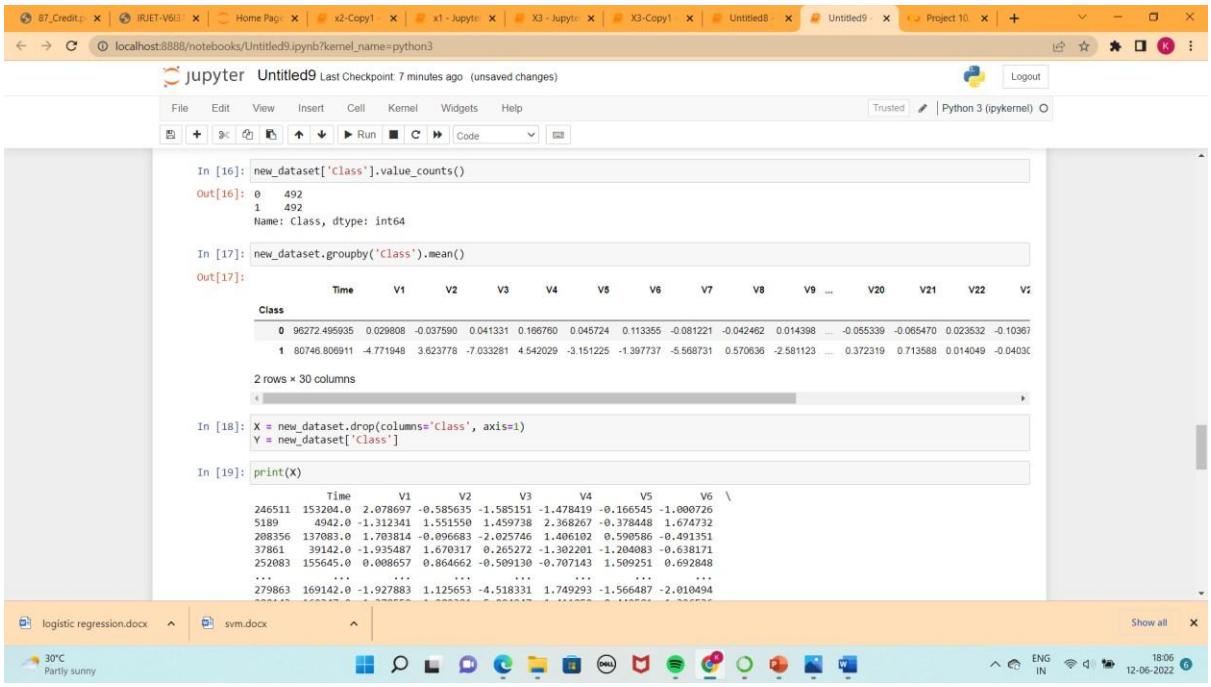
30°C Partly sunny ENG IN 18:05 12-06-2022


jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
14 V14 284807 non-null float64
15 V15 284807 non-null float64
16 V16 284807 non-null float64
17 V17 284807 non-null float64
18 V18 284807 non-null float64
19 V19 284807 non-null float64
20 V20 284807 non-null float64
21 V21 284807 non-null float64
22 V22 284807 non-null float64
23 V23 284807 non-null float64
24 V24 284807 non-null float64
25 V25 284807 non-null float64
26 V26 284807 non-null float64
27 V27 284807 non-null float64
28 V28 284807 non-null float64
29 Amount 284807 non-null float64
30 Class 284807 non-null int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
In [6]:
# checking the number of missing values in each column
credit_card_data.isnull().sum()
Out[6]:
Time 0
V1 0
V2 0
V3 0
V4 0
V5 0
V6 0
V7 0
V8 0
V9 0
V10 0
logistic regression.docx
svm.docx
Show all
30°C
Partly sunny
ENG
IN
18:05
12-06-2022


jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
V17 0
V18 0
V19 0
V20 0
V21 0
V22 0
V23 0
V24 0
V25 0
V26 0
V27 0
V28 0
Amount 0
Class 0
dtype: int64
In [7]:
distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
Out[7]:
0 284315
1 492
Name: Class, dtype: int64
In [8]:
# separating the data for analysis
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
In [9]:
print(legit.shape)
print(fraud.shape)
(284315, 31)
(492, 31)
logistic regression.docx
svm.docx
Show all
30°C
Partly sunny
ENG
IN
18:05
12-06-2022


localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
In [10]:
# statistical measures of the data
legit.Amount.describe()
Out[10]:
count 284315.000000
mean 88.291022
std 250.105092
min 0.000000
25% 5.650000
50% 22.000000
75% 77.050000
max 25691.160000
Name: Amount, dtype: float64
In [11]:
fraud.Amount.describe()
Out[11]:
count 492.000000
mean 122.211321
std 256.683288
min 0.000000
25% 1.000000
50% 9.250000
75% 105.890000
max 2125.870000
Name: Amount, dtype: float64
In [12]:
# compare the values for both transactions
credit_card_data.groupby('Class').mean()
Out[12]:
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V20 V21 V22
Class
0 94838.202258 0.008258 -0.006271 0.012171 -0.007860 0.005453 0.002419 0.009637 -0.000987 0.004467 ... -0.000644 -0.001235 -0.000024
logistic regression.docx
svm.docx
Show all
30°C
Partly sunny
ENG
IN
18:06
12-06-2022


localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
Out[12]:
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V20 V21 V22
Class
0 94838.202258 0.008258 -0.006271 0.012171 -0.007860 0.005453 0.002419 0.009637 -0.000987 0.004467 ... -0.000644 -0.001235 -0.000024
1 80746.806911 -4.771948 3.623778 -7.033281 4.542029 -3.151225 -1.397737 -5.568731 0.570636 -2.581123 ... 0.372319 0.713588 0.014049
2 rows × 30 columns
In [13]:
legit_sample = legit.sample(n=492)
In [14]:
new_dataset = pd.concat([legit_sample, fraud], axis=0)
In [15]:
new_dataset.tail()
Out[15]:
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V21 V22 V23 V24
279863 169142.0 -1.927883 1.125653 -4.518331 1.749293 -1.566487 -2.010494 -0.882850 0.697211 -2.064945 ... 0.778584 -0.319189 0.639419 -0.294885
280143 169347.0 1.378559 1.289381 -5.004247 1.411850 0.442581 -1.326536 -1.413170 0.248525 -1.127396 ... 0.370612 0.028234 -0.145640 -0.081049
280149 169351.0 -0.676143 1.126366 -2.213700 0.468308 -1.120541 -0.003346 -2.234739 1.210158 -0.652250 ... 0.751826 0.834108 0.190944 0.032070
281144 169966.0 -3.113832 0.585864 -5.399730 1.817092 -0.840618 -2.943548 -2.208002 1.058733 -1.632333 ... 0.583276 -0.269209 -0.456108 -0.183659
281674 170348.0 1.991976 0.158476 -2.583441 0.408670 1.151147 -0.096695 0.223050 -0.068384 0.577829 ... -0.164350 -0.295135 -0.072173 -0.450261
5 rows × 31 columns
In [16]:
new_dataset['Class'].value_counts()
logistic regression.docx
svm.docx
Show all
30°C
Partly sunny
ENG
IN
18:06
12-06-2022



jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Logout

Run Code

In [16]:
```
new_dataset['Class'].value_counts()
```

Out[16]:
```
0    492
1    492
Name: Class, dtype: int64
```

In [17]:
```
new_dataset.groupby('Class').mean()
```

Out[17]:

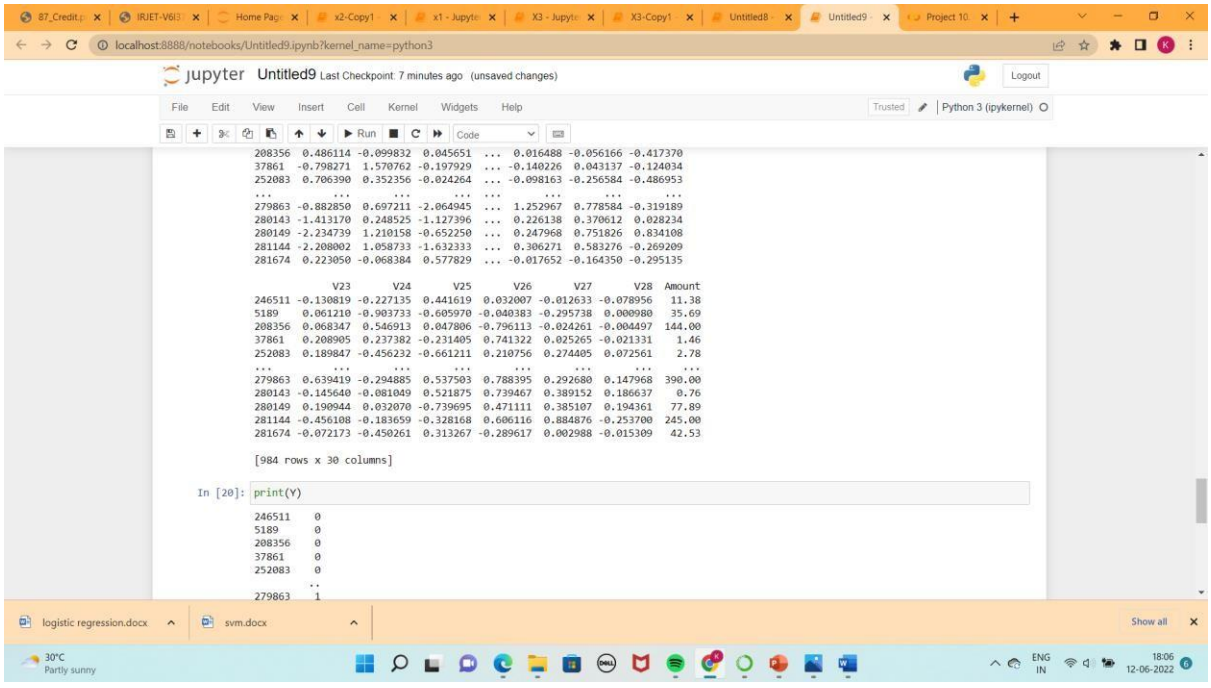
Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V2
0	96272.495935	0.029808	-0.037590	0.041331	0.166760	0.045724	0.113355	-0.081221	-0.042462	0.014398	...	-0.055339	-0.065470	0.023532	-0.10367
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.0403(

2 rows × 30 columns

In [18]:
```
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
```

In [19]:
```
print(X)
```

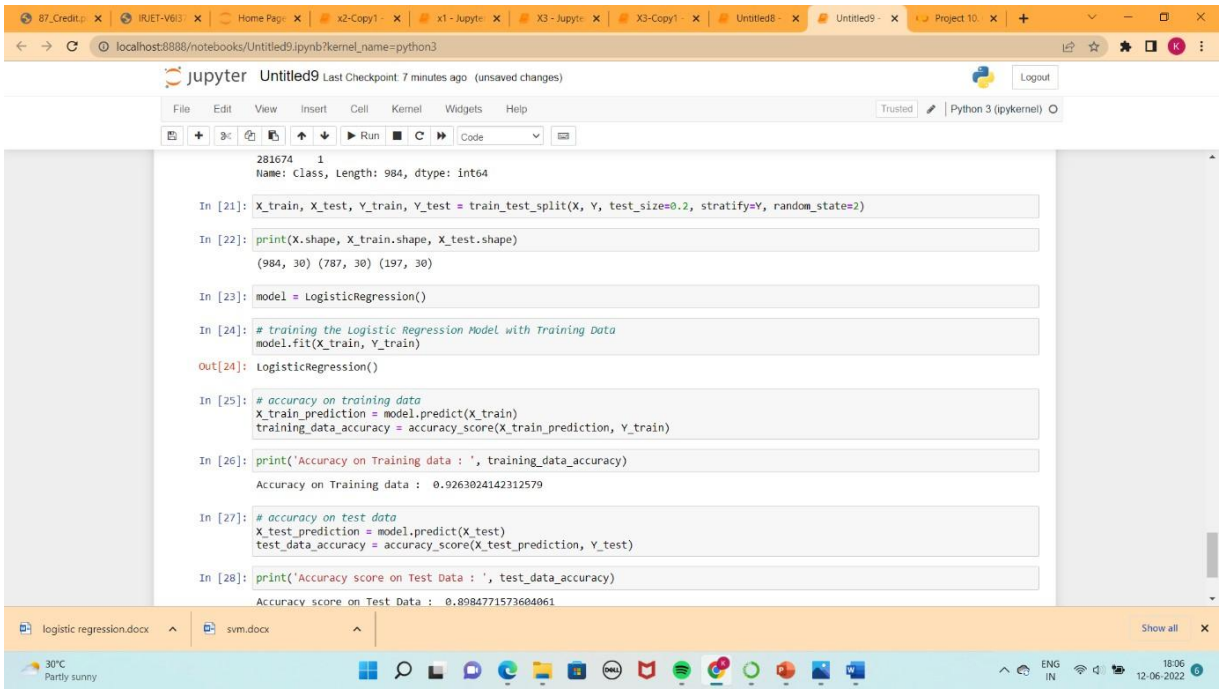
```
            Time        V1        V2        V3        V4        V5        V6  \
246511  153204.0  2.078697 -0.585635 -1.585151 -1.478419 -0.166545 -1.000726
5189      4942.0 -1.312341  1.551550  1.459738  2.368267 -0.378448  1.674732
208356  137083.0  1.703814 -0.096683 -2.025746  1.406102  0.590586 -0.491351
37861    39142.0 -1.935487  1.670317  0.265272 -1.302201 -1.204083 -0.638171
252083  155645.0  0.008657  0.864662 -0.509130 -0.707143  1.509251  0.692848
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
```

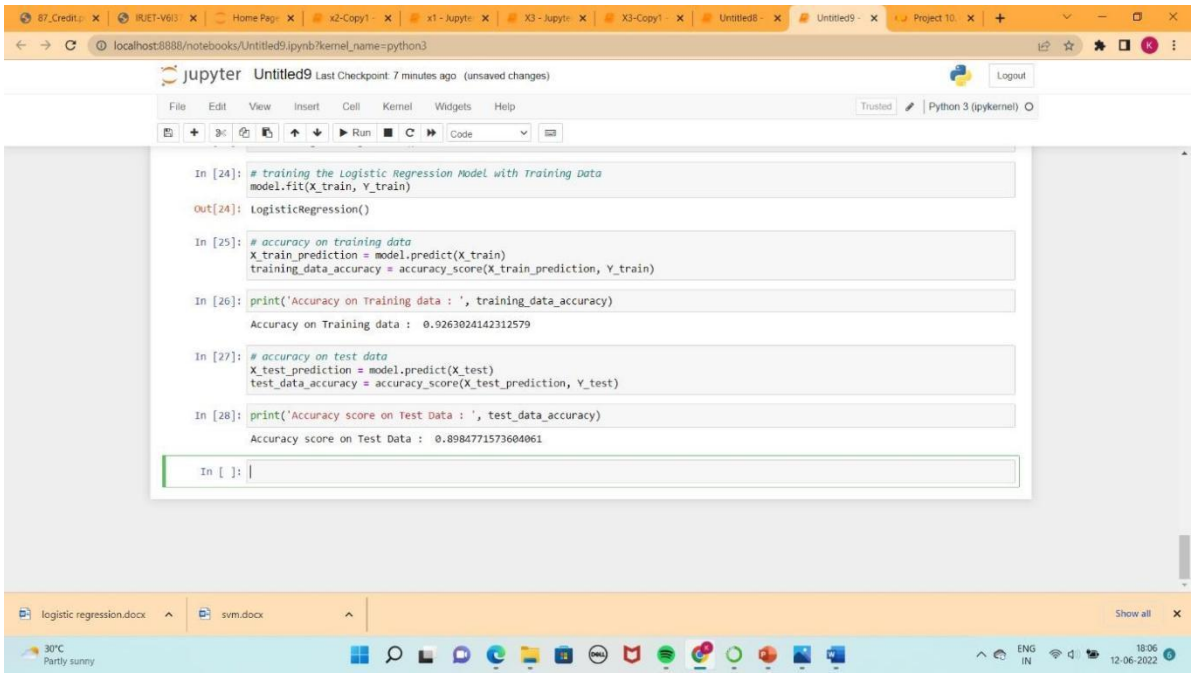




```
            208356  0.486114 -0.099832  0.045651  ...  0.016488 -0.056166 -0.417370
37861   -0.798271  1.570762 -0.197929  ... -0.140226  0.043137 -0.124034
252083  0.706390  0.352356 -0.024264  ... -0.098163 -0.256584 -0.486953
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135

                V23       V24       V25       V26       V27       V28  Amount
246511 -0.130819 -0.227135  0.441619  0.032007 -0.012633 -0.078956   11.38
5189    0.061210 -0.903733 -0.605970 -0.040383 -0.295738  0.000980   35.69
208356  0.068347  0.546913  0.047806 -0.796113 -0.024261 -0.004497  144.00
37861   0.208905  0.237382 -0.231405  0.741322  0.025265 -0.021331    1.46
252083  0.189847 -0.456232 -0.661211  0.210756  0.274405  0.072561    2.78
...          ...       ...       ...       ...       ...       ...     ...
279863  0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00
280143 -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76
280149  0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89
281144 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00
281674 -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53

[984 rows x 30 columns]
```

In [20]:
```
print(Y)
```

```
246511    0
5189      0
208356    0
37861     0
252083    0
         ..
279863    1
```


jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
281674 1
Name: Class, Length: 984, dtype: int64
In [21]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
In [22]: print(X.shape, X_train.shape, X_test.shape)
(984, 30) (787, 30) (197, 30)
In [23]: model = LogisticRegression()
In [24]: # training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)
Out[24]: LogisticRegression()
In [25]: # accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
In [26]: print('Accuracy on Training data : ', training_data_accuracy)
Accuracy on Training data : 0.9263024142312579
In [27]: # accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
In [28]: print('Accuracy score on Test Data : ', test_data_accuracy)
Accuracy score on Test Data : 0.8984771573604061
logistic regression.docx
svm.docx
Show all
30°C Partly sunny
ENG IN
18:06 12-06-2022


jupyter Untitled9 Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
In [24]: # training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)
Out[24]: LogisticRegression()
In [25]: # accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
In [26]: print('Accuracy on Training data : ', training_data_accuracy)
Accuracy on Training data : 0.9263024142312579
In [27]: # accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
In [28]: print('Accuracy score on Test Data : ', test_data_accuracy)
Accuracy score on Test Data : 0.8984771573604061
In [ ]:
logistic regression.docx
svm.docx
Show all
30°C Partly sunny
ENG IN
18:06 12-06-2022

9.3 SUPPORT VECTOR MACHINE

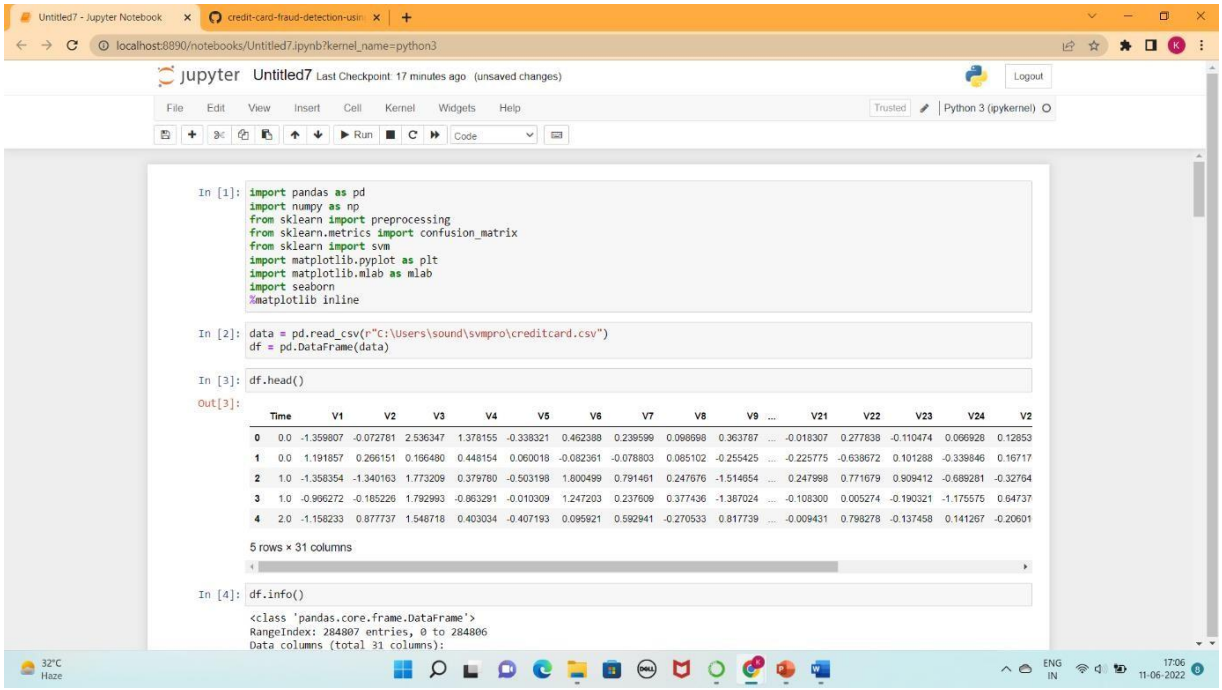
9.3.1 CODE

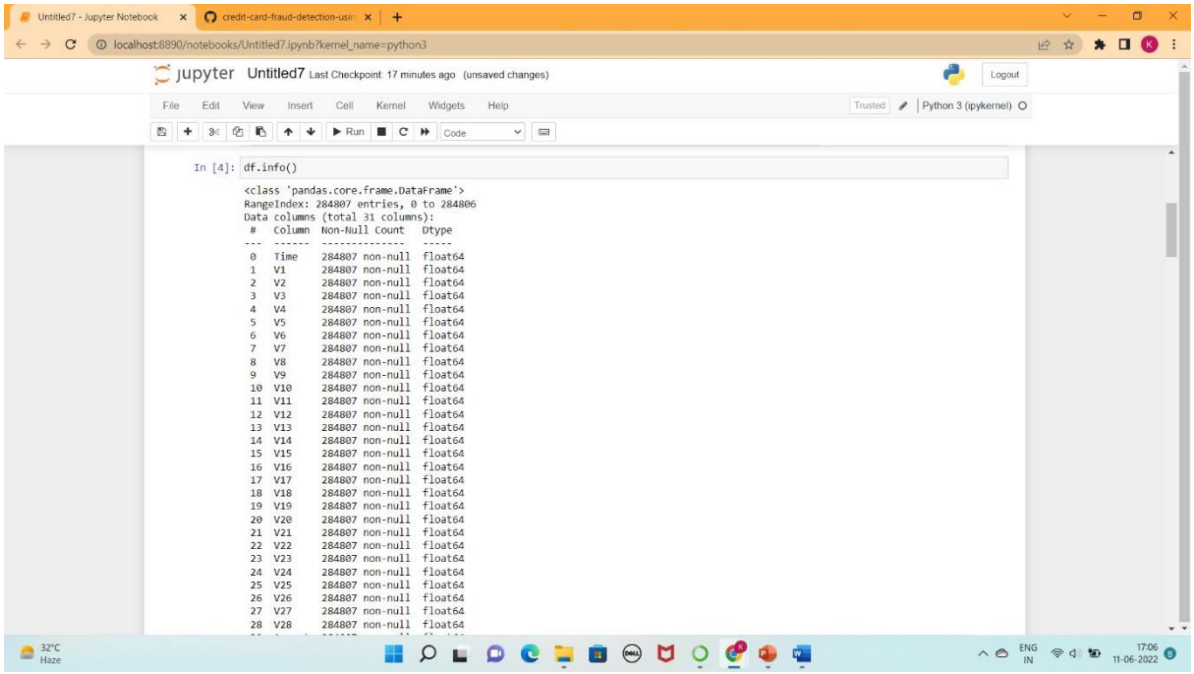
```
import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.metrics import confusion_matrix

from sklearn import svm
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import seaborn

%matplotlib inline

data = pd.read_csv()


df = pd.DataFrame(data)


df.info()


df.describe()


df_fraud = df[df['Class'] == 1] # Recovery of fraud data

plt.figure(figsize=(15,5))
plt.scatter(df_fraud['Time'], df_fraud['Amount']) # Display fraud
amountsaccording to their time


plt.title('Scratter plot amount fraud')
plt.xlabel('Time')
plt.ylabel('Amount')
plt.xlim([0,175000])
plt.ylim([0,2500])
plt.show()
```

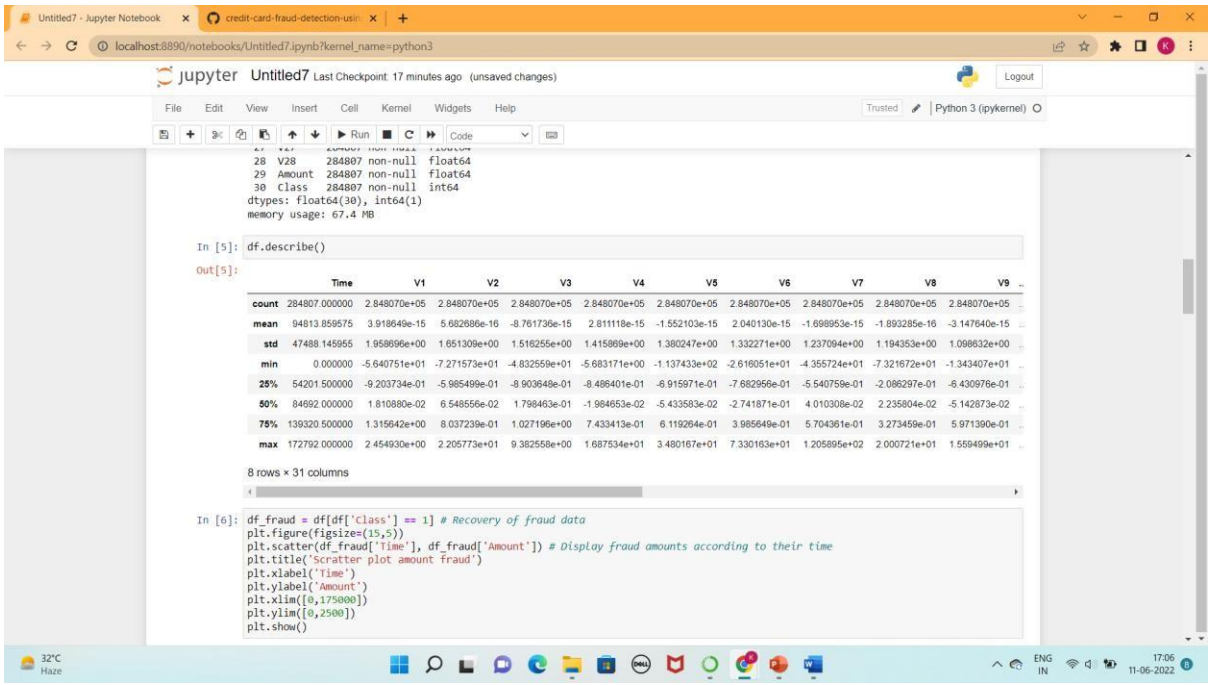
```
number_fraud = len(data[data.Class == 1])
number_no_fraud = len(data[data.Class == 0])
```

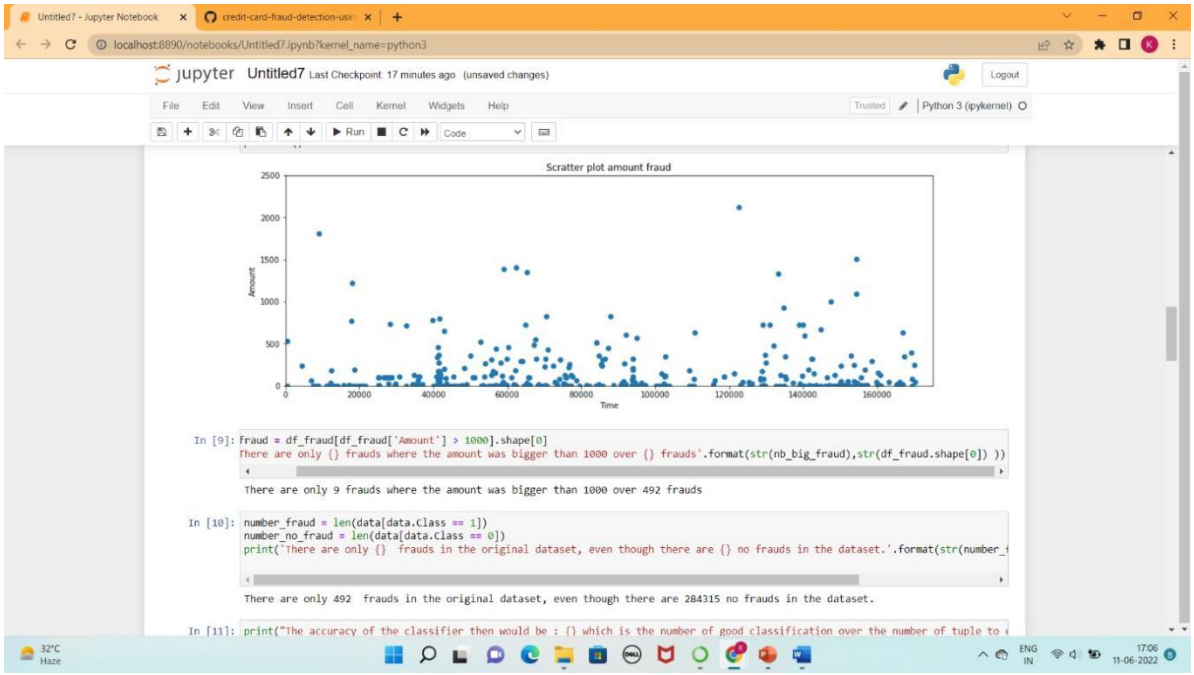
```
print('There are only {} frauds in the original dataset, even though there are {} no frauds in the dataset.'.format(str(number_fraud),str(number_no_fraud)


print("The accuracy of the classifier then would be : {} which is the number of good classification over the number of tuple to classify".format(str((284315-492)/284315)))

df_corr = df.corr()
plt.figure(figsize=(6,5))
seaborn.heatmap(df_corr, cmap='Blues')
seaborn.set(font_scale=2,style='white')

plt.title('Heatmap correlation')
plt.show()
```

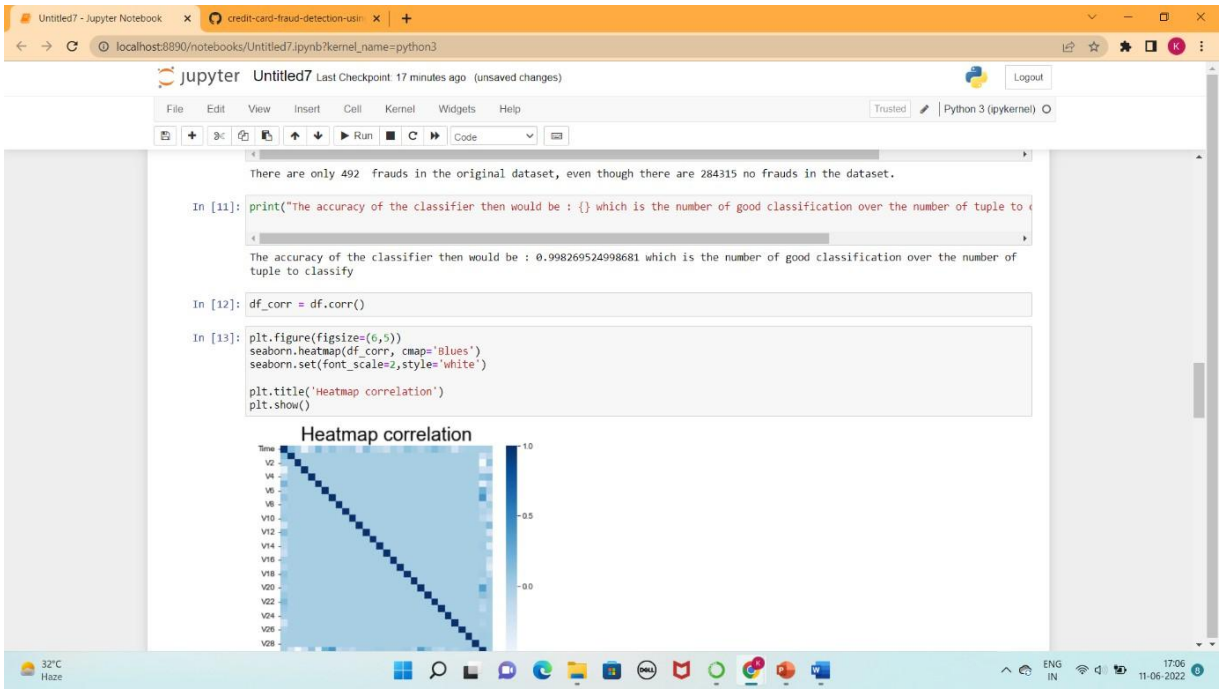
9.3.2 OUTPUT

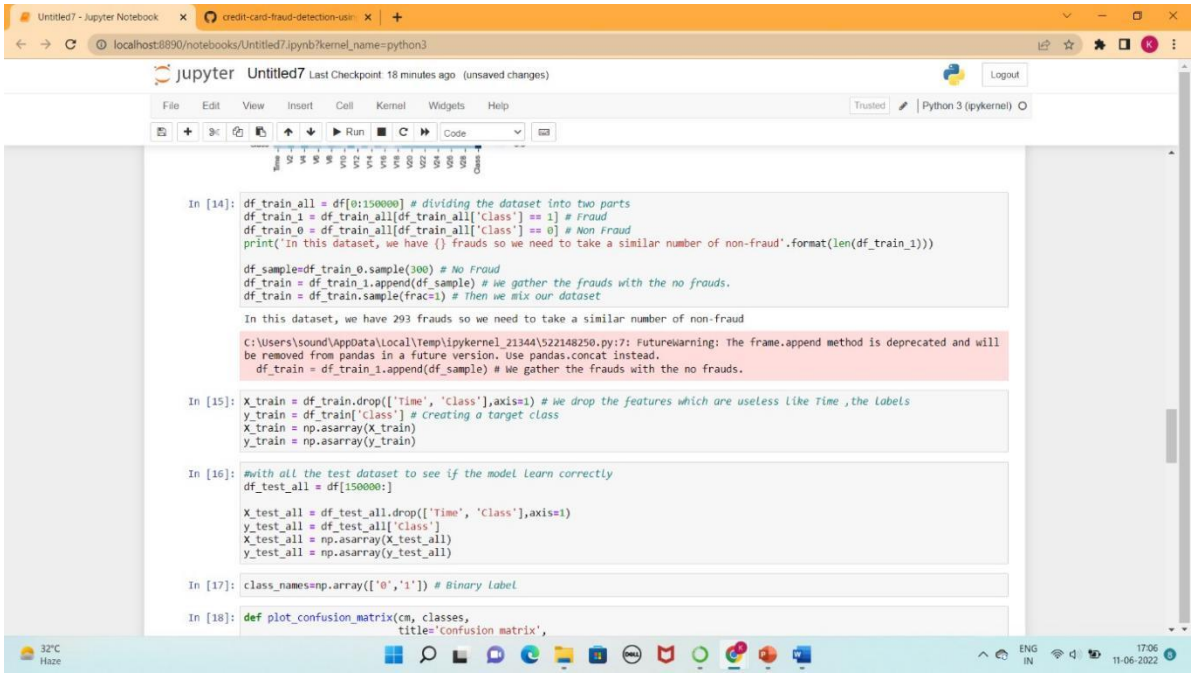

Untitled7 - Jupyter Notebook
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 17 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn import svm
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import seaborn
%matplotlib inline
In [2]:
data = pd.read_csv(r"C:\Users\sound\svmpro\creditcard.csv")
df = pd.DataFrame(data)
In [3]:
df.head()
Out[3]:
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V21 V22 V23 V24 V2
0 0.0 -1.359807 -0.072781 2.536347 1.378155 -0.338321 0.462388 0.239599 0.098698 0.363787 ... -0.018307 0.277838 -0.110474 0.066928 0.12853
1 0.0 1.191857 0.266151 0.166480 0.448154 0.060018 -0.082361 -0.078803 0.085102 -0.255425 ... -0.225775 -0.638672 0.101288 -0.339846 0.16717
2 1.0 -1.358354 -1.340163 1.773209 0.379780 -0.503198 1.800499 0.791461 0.247676 -1.514654 ... 0.247998 0.771679 0.909412 -0.689281 -0.32764
3 1.0 -0.966272 -0.185226 1.792993 -0.863291 -0.010309 1.247203 0.237609 0.377436 -1.387024 ... -0.108300 0.005274 -0.190321 -1.175575 0.64737
4 2.0 -1.158233 0.877737 1.548718 0.403034 -0.407193 0.095921 0.592941 -0.270533 0.817739 ... -0.009431 0.798278 -0.137458 0.141267 -0.20601
5 rows × 31 columns
In [4]:
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
32°C Haze
ENG IN
17:06 11-06-2022


Untitled7 - Jupyter Notebook
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 17 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [4]:
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 Time 284807 non-null float64
1 V1 284807 non-null float64
2 V2 284807 non-null float64
3 V3 284807 non-null float64
4 V4 284807 non-null float64
5 V5 284807 non-null float64
6 V6 284807 non-null float64
7 V7 284807 non-null float64
8 V8 284807 non-null float64
9 V9 284807 non-null float64
10 V10 284807 non-null float64
11 V11 284807 non-null float64
12 V12 284807 non-null float64
13 V13 284807 non-null float64
14 V14 284807 non-null float64
15 V15 284807 non-null float64
16 V16 284807 non-null float64
17 V17 284807 non-null float64
18 V18 284807 non-null float64
19 V19 284807 non-null float64
20 V20 284807 non-null float64
21 V21 284807 non-null float64
22 V22 284807 non-null float64
23 V23 284807 non-null float64
24 V24 284807 non-null float64
25 V25 284807 non-null float64
26 V26 284807 non-null float64
27 V27 284807 non-null float64
28 V28 284807 non-null float64
32°C Haze
ENG IN
17:06 11-06-2022


Untitled7 - Jupyter Notebook
credit-card-fraud-detection-usin
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 17 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
28 V28 284807 non-null float64
29 Amount 284807 non-null float64
30 Class 284807 non-null int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
In [5]: df.describe()
Out[5]:
Time V1 V2 V3 V4 V5 V6 V7 V8 V9
count 284807.000000 2.848070e+05 2.848070e+05 2.848070e+05 2.848070e+05 2.848070e+05 2.848070e+05 2.848070e+05 2.848070e+05 2.848070e+05
mean 94813.859575 3.918649e-15 5.682686e-16 -8.761736e-15 2.811118e-15 -1.552103e-15 2.040130e-15 -1.698953e-15 -1.893285e-16 -3.147640e-15
std 47488.145955 1.958696e+00 1.651309e+00 1.516255e+00 1.415869e+00 1.380247e+00 1.332271e+00 1.237094e+00 1.194353e+00 1.098632e+00
min 0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00 -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25% 54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01 -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50% 84692.000000 1.810880e-02 6.548556e-02 1.798463e-01 -1.984653e-02 -5.433583e-02 -2.741871e-01 4.010308e-02 2.235804e-02 -5.142873e-02
75% 139320.500000 1.315642e+00 8.037239e-01 1.027196e+00 7.433413e-01 6.119264e-01 3.985649e-01 5.704361e-01 3.273459e-01 5.971390e-01
max 172792.000000 2.454930e+00 2.205773e+01 9.382558e+00 1.687534e+01 3.480167e+01 7.330163e+01 1.205895e+02 2.000721e+01 1.559499e+01
8 rows × 31 columns
In [6]: df_fraud = df[df['Class'] == 1] # Recovery of fraud data
plt.figure(figsize=(15,5))
plt.scatter(df_fraud['Time'], df_fraud['Amount']) # Display fraud amounts according to their time
plt.title('Scratter plot amount fraud')
plt.xlabel('Time')
plt.ylabel('Amount')
plt.xlim([0,175000])
plt.ylim([0,2500])
plt.show()
32°C Haze
ENG IN
17:06 11-06-2022


Untitled7 - Jupyter Notebook
credit-card-fraud-detection-usin
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 17 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Scratter plot amount fraud
Amount
Time
0
500
1000
1500
2000
2500
0
20000
40000
60000
80000
100000
120000
140000
160000
In [9]: Fraud = df_fraud[df_fraud['Amount'] > 1000].shape[0]
There are only {} frauds where the amount was bigger than 1000 over {} frauds'.format(str(nb_big_fraud),str(df_fraud.shape[0])))
There are only 9 frauds where the amount was bigger than 1000 over 492 frauds
In [10]: number_fraud = len(data[data.Class == 1])
number_no_fraud = len(data[data.Class == 0])
print('There are only {} frauds in the original dataset, even though there are {} no frauds in the dataset.'.format(str(number_
There are only 492 frauds in the original dataset, even though there are 284315 no frauds in the dataset.
In [11]: print("The accuracy of the classifier then would be : {} which is the number of good classification over the number of tuple to
32°C Haze
ENG IN
17:06 11-06-2022


Untitled7 - Jupyter Notebook
credit-card-fraud-detection-usin
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 17 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
There are only 492 frauds in the original dataset, even though there are 284315 no frauds in the dataset.
In [11]: print("The accuracy of the classifier then would be : {} which is the number of good classification over the number of tuple to c
The accuracy of the classifier then would be : 0.998269524998681 which is the number of good classification over the number of tuple to classify
In [12]: df_corr = df.corr()
In [13]: plt.figure(figsize=(6,5))
seaborn.heatmap(df_corr, cmap='Blues')
seaborn.set(font_scale=2,style='White')
plt.title('Heatmap correlation')
plt.show()
Heatmap correlation
Time
V2
V4
V6
V8
V10
V12
V14
V16
V18
V20
V22
V24
V26
V28
1.0
0.5
0.0
32°C
Haze
ENG
IN
17:06
11-06-2022


Untitled7 - Jupyter Notebook
credit-card-fraud-detection-usin
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3
jupyter Untitled7 Last Checkpoint: 18 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [14]: df_train_all = df[0:150000] # dividing the dataset into two parts
df_train_1 = df_train_all[df_train_all['Class'] == 1] # Fraud
df_train_0 = df_train_all[df_train_all['Class'] == 0] # Non Fraud
print('In this dataset, we have {} frauds so we need to take a similar number of non-fraud'.format(len(df_train_1)))
df_sample=df_train_0.sample(300) # No Fraud
df_train = df_train_1.append(df_sample) # We gather the frauds with the no frauds.
df_train = df_train.sample(frac=1) # Then we mix our dataset
In this dataset, we have 293 frauds so we need to take a similar number of non-fraud
C:\Users\sound\AppData\Local\Temp\ipykernel_21344\522148250.py:7: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
df_train = df_train_1.append(df_sample) # We gather the frauds with the no frauds.
In [15]: X_train = df_train.drop(['Time', 'Class'],axis=1) # We drop the features which are useless like Time ,the labels
y_train = df_train['Class'] # Creating a target class
X_train = np.asarray(X_train)
y_train = np.asarray(y_train)
In [16]: #with all the test dataset to see if the model learn correctly
df_test_all = df[150000:]
X_test_all = df_test_all.drop(['Time', 'Class'],axis=1)
y_test_all = df_test_all['Class']
X_test_all = np.asarray(X_test_all)
y_test_all = np.asarray(y_test_all)
In [17]: class_names=np.array(['0','1']) # Binary label
In [18]: def plot_confusion_matrix(cm, classes,
title='Confusion matrix',
32°C
Haze
ENG
IN
17:06
11-06-2022

Untitled7 - Jupyter Notebook

localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3

jupyter Untitled7 Last Checkpoint: 18 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

In [18]:
```
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [19]:
```
classifier = svm.SVC(kernel='linear')
```

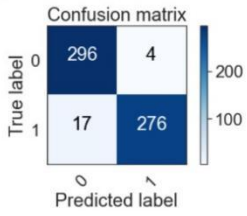
In [20]:
```
classifier.fit(X_train, y_train)
```

Out[20]:
```
SVC(kernel='linear')
```

In [21]:
```
prediction_SVM = classifier.predict(X_train)
```

In [23]:
```
import itertools
```

In [24]:
```
cm = confusion_matrix(y_train, prediction_SVM)
plot_confusion_matrix(cm,class_names)
```



In [25]:
```
print('We have detected {} frauds / {} total frauds.'.format(str(cm[1][1]),str(cm[1][1]+cm[1][0])))
print('\nSo, the probability to detect a fraud is {}'.format(str(cm[1][1]/(cm[1][1]+cm[1][0]))))
print("the accuracy is : {}".format(str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])))))
```

```
We have detected 276 frauds / 293 total frauds.

So, the probability to detect a fraud is 0.9419795221843004
the accuracy is : 0.9645868465430016
```

In [26]:
```
classifier = svm.SVC(kernel='linear')
```

In [27]:
```
classifier.fit(X_train,y_train)
```

Out[27]:
```
SVC(kernel='linear')
```

Untitled7 - Jupyter Notebook

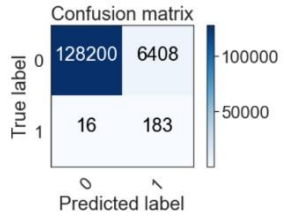
localhost:8890/notebooks/Untitled7.ipynb?kernel_name=python3

jupyter Untitled7 Last Checkpoint 18 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

```
In [27]: classifier.fit(X_train,y_train)
Out[27]: SVC(kernel='linear')

In [28]: prediction_SVM_all = classifier.predict(X_test_all)

In [29]: cm = confusion_matrix(y_test_all, prediction_SVM_all)
         plot_confusion_matrix(cm,class_names)
```




```
In [30]: print('We have detected {} frauds / {} total frauds.'.format(str(cm[1][1]),str(cm[1][1]+cm[1][0])))
         print('\nSo, the probability to detect a fraud is {}'.format(str(cm[1][1]/(cm[1][1]+cm[1][0]))))
         print("the accuracy is : {}".format(str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])))))
```

```
We have detected 183 frauds / 199 total frauds.

So, the probability to detect a fraud is 0.9195979899497487
the accuracy is : 0.9523466882283561
```

```
In [ ]:
```

32°C Haze ENG IN 17:07 11-06-2022

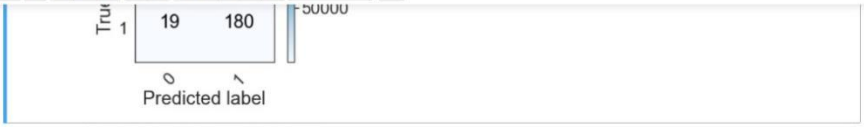
87_Credit.pdf | IRJET-V6I371 | Home Page - | x2-Copy1 - Ju | x1 - Jupyter N | X3 - Jupyter N | X3-Copy1 - Ju | support vecto | support vecto

localhost:8888/notebooks/support%20vector%20machine.ipynb

jupyter support vector machine Last Checkpoint 6 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)




```
In [26]: print('Our criterion give a result of '
               + str( ( (cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1])) + 4 * cm[1][1]/(cm[1][0]+cm[1][1])) / 5))
```

```
Our criterion give a result of 0.9120815975401723
```

```
In [27]: print('We have detected ' + str(cm[1][1]) + ' frauds / ' + str(cm[1][1]+cm[1][0]) + ' total frauds.')
         print('\nSo, the probability to detect a fraud is ' + str(cm[1][1]/(cm[1][1]+cm[1][0])))
         print("the accuracy is : "+str((cm[0][0]+cm[1][1]) / (sum(cm[0]) + sum(cm[1]))))
```

```
We have detected 180 frauds / 199 total frauds.

So, the probability to detect a fraud is 0.9045226130653267
the accuracy is : 0.9423175354395543
```

```
In [ ]:
```

logistic regression.docx svm.docx Show all

28°C Haze ENG IN 18:31 12-06-2022

REFERENCES

1]Credit card fraud detection using machine learning and python

Link:

https://towardsdatascience.com/credit-card-fraud-detection-using-machine-learning-python-5b098d4a8edc

2]Machine learning based credit card fraud detection using GA algorithm

Link:

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00573-8

3]Credit card fraud detection-kaggle

Link:

https://www.kaggle.com/mlg-ulb/creditcardfraud

4]Credit card fraud detection using fraud detection

Link:

https://www.kaggle.com/hassanamin/credit-card-fraud-detection-using-random-forest

5]Random forest in credit card fraud analysis

Link:

https://www.researchgate.net/publication/347441357_THE_ROLE_OF_RANDOM_FOREST_IN_CREDIT_CARD_FRAUD_ANALYSIS

6]Fraud detection using random forest, neural autoencoder

Link:

https://www.infoq.com/articles/fraud-detection-random-forest/

7. U. Fiore, A. De Santis, F. Perla, P. Zanetti, F. Palmieri, Using generative adversarial networksfor improving classification effectiveness in credit card fraud detection. Inf. Sci. 479, 448–455(2019)

8. N. Carneiro, G. Figueira, M. Costa, A data mining based system for credit-card fraud detectionin e-tail. Decis. Supp. Syst. 95, 91–101 (2017)

9. A.C. Bahnsen, D. Aouada, A. Stojanovic, B. Ottersten, Feature engineeringstrategies for creditcard fraud detection. Exp. Syst. Appl. 51, 134–142 (2016)

10. M. Zareapoor, P. Shamsolmoali, Application of credit card fraud detection:based on baggingand ensemble classifier. Proc. Comput. Sci. 48, 679–685 (2015)

11. K. Randhawa, C.K. Loo, M. Seera, C.P. Lim, A.K. Nandi, Credit card frauddetection usingAdaBoost and majority voting. IEEE Access 6, 14277–14284 (2017)

12. P. Save, P. Tiwarekar, K.N. Jain, N. Mahyavanshi, A novel idea for credit card fraud detectionusing decision tree. Int. J. Comput. Appl. 161(13), 0975– 8887 (2017)

13. S. Sorournejad, Z. Zojaji, R.E. Atani, A.H. Monadjemi, A survey of credit card fraud detectiontechniques: data and technique oriented perspective. ArXiv(2016)

14. A. Singh, A. Jain, Adaptive credit card fraud detection techniques based onfeature selectionmethod. Adv. Comput. Commun. Comput. Sci., 167–178 (2019)

15. Z. Li, G. Liu, S. Wang,S. Xuan, C. Jiang, Credit card fraud detection viakernel-based supervisedhashing, in 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & TrustedComputing, Scalable Computing & Communications, Cloud & Big Data Computing, Internetof People and Smart City Innovation (2018)

16. S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, C. Jiang, Random forest forcredit card frauddetection, in 2018 IEEE 15th International Conference on Networking, Sensing and Control(ICNSC) (2018)