**Credit Card Fraud Detection Using Machine Learning**

**PROJECT REPORT**

**Submitted in partial fulfillment of the Requirements**

**For the award of Master of Computer Application Degree**

**LNCT UNIVERSITY, BHOPAL (M.P.)**



**MINOR PROJECT REPORT**

**Submitted by -**

**APARNA THAKUR**

**Enrollment no.- LNCCMCA11108**

**Under the Guidance of –**

**LNCT UNIVERSITY, BHOPAL (M.P.)**

**MASTER OF COMPUTER APPLICATION**

**CERTIFICATE**

This is to certify that the work embodied in this Minor project “**STOCK TSLA PREDICTION & CSV Analysis**” has been satisfactorily completed by **APARNA THAKUR** of MCA(AIML) Sec A Enrollment No. LNCCMCA11108. It is a bonafide piece of work, carried out under my guidance in the **Master of Computer Application, LNCT University, Bhopal** for the partial fulfillment of the **Master of Computer Application** degree during the academic session July-December, 2022.

Guided By

**< Prof. Sonal Saklle >**

**Approved By**

**Head of Department**

**Forwarded by**

**Director**

**LNCT UNIVERSITY, BHOPAL (M.P.)**

**MASTER OF COMPUTER APPLICATION**

**CERTIFICATE OF APPROVAL**

This foregoing minor project is hereby approved as a creditable study of a Computer Application Subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only for the purpose for which it has been submitted.

Internal Examiner External Examiner

**LNCT UNIVERSITY, BHOPAL (M.P.)**

**MASTER OF COMPUTER APPLICATION**

**DECLARATION**

I, Aparna Thakur student of Master of Computer Application, LNCT University, Bhopal, hereby declare that the work presented in this Minor Project is outcome of my own work, is bonafide, correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any University for the award of any degree or any professional diploma.

**Aparna Thakur (LNCCMCA11108)**

**Date: 04/07/2023**

**Chapters**

**1. INTRODUCTION**

**2. HARDWARE & SOFTWARE REQUIREMENT**

**4. DESIGN / GRAPH / METHODOLOGY**

**6. CODE/IMPLEMENTATION**

**7. SCREENSHOTS**

**8. BIBLIOGRAPHY**

**Introduction**

**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 datasets contains transactions made by credit cards in September 2013 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.

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-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

**Inspiration**

Identify fraudulent credit card transactions.

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.

**Acknowledgements**

The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group ([http://mlg.ulb.ac.be](http://mlg.ulb.ac.be/)) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. More details on current and past projects on related topics are available on <https://www.researchgate.net/project/Fraud-detection-5> and the page of the DefeatFraud project

**Model Prediction**

Now it is time to start building the model .The types of algorithms we are going to use to try to do anomaly detection on this dataset are as follows

**Isolation Forest Algorithm :**

One of the newest techniques to detect anomalies is called Isolation Forests. The algorithm is based on the fact that anomalies are data points that are few and different. As a result of these properties, anomalies are susceptible to a mechanism called isolation.

This method is highly useful and is fundamentally different from all existing methods. It introduces the use of isolation as a more effective and efficient means to detect anomalies than the commonly used basic distance and density measures. Moreover, this method is an algorithm with a low linear time complexity and a small memory requirement. It builds a good performing model with a small number of trees using small sub-samples of fixed size, regardless of the size of a data set.

Typical machine learning methods tend to work better when the patterns they try to learn are balanced, meaning the same amount of good and bad behaviors are present in the dataset.

How Isolation Forests Work The Isolation Forest algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The logic argument goes: isolating anomaly observations is easier because only a few conditions are needed to separate those cases from the normal observations. On the other hand, isolating normal observations require more conditions. Therefore, an anomaly score can be calculated as the number of conditions required to separate a given observation.

The way that the algorithm constructs the separation is by first creating isolation trees, or random decision trees. Then, the score is calculated as the path length to isolate the observation.

**Local Outlier Factor(LOF) Algorithm**

The LOF algorithm is an unsupervised outlier detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outlier samples that have a substantially lower density than their neighbors.

The number of neighbors considered, (parameter n\_neighbors) is typically chosen 1) greater than the minimum number of objects a cluster has to contain, so that other objects can be local outliers relative to this cluster, and 2) smaller than the maximum number of close by objects that can potentially be local outliers. In practice, such informations are generally not available, and taking n\_neighbors=20 appears to work well in general.

**HARDWARE & SOFTWARE REQUIREMENT:**

**SOFTWARE REQUIREMENT:**

**1) Operating System : Microsof windows 10 board system**

**2) Language : Python**

**3) Environment : Jupyter Notebook**

**HARDWARE REQUIREMENT:**

**1) Operating System : windows OS**

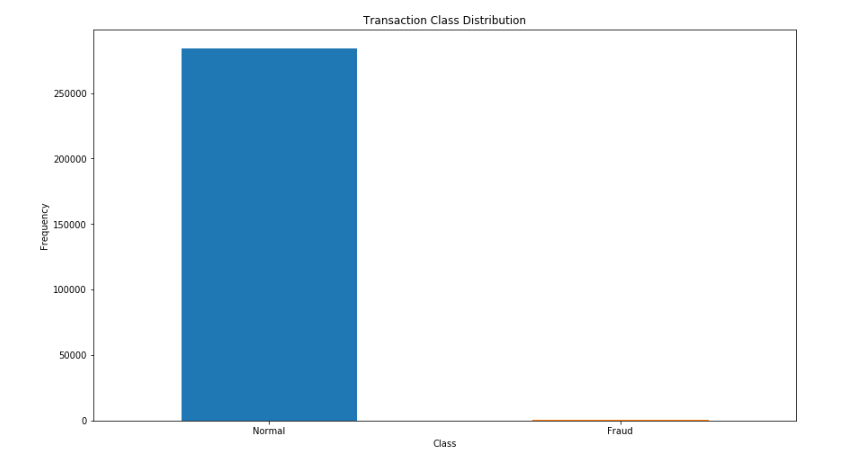
**2) MEMORY : 256mb RAM**

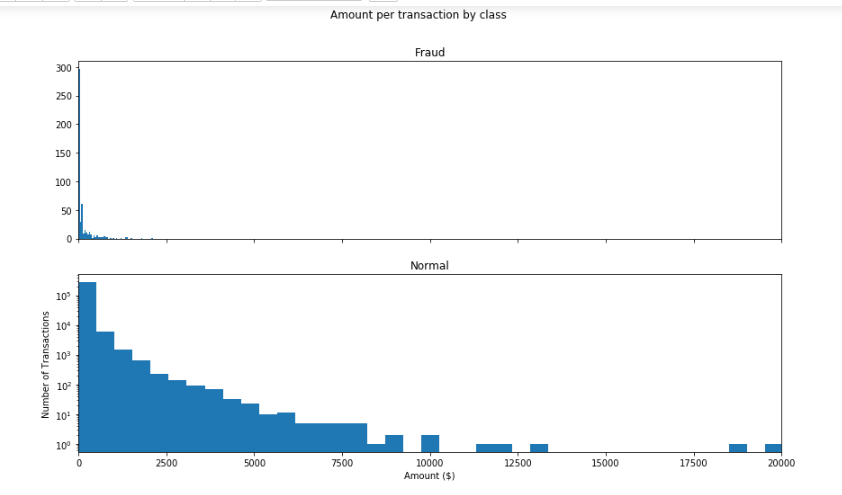
**3) MISCELLANEOUS : Keyboard,Mouse**

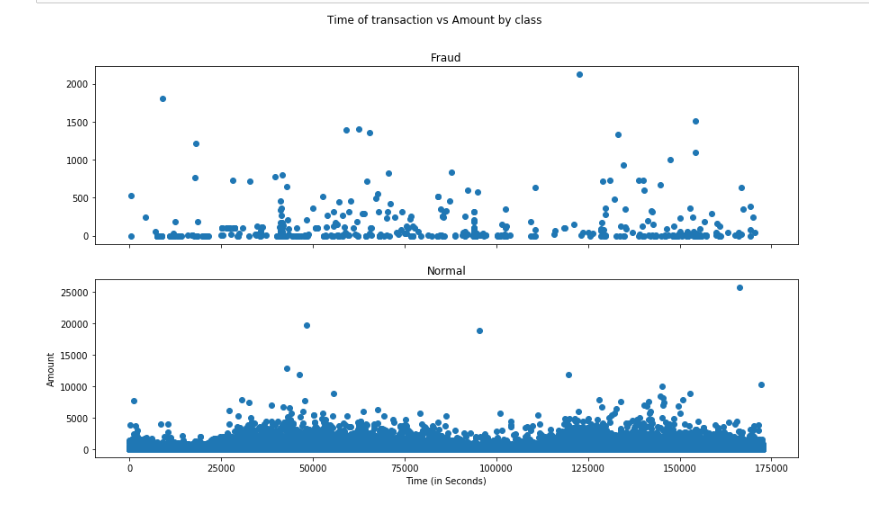
**4) PROCESSOR : intel 3 personal computer with minimum**

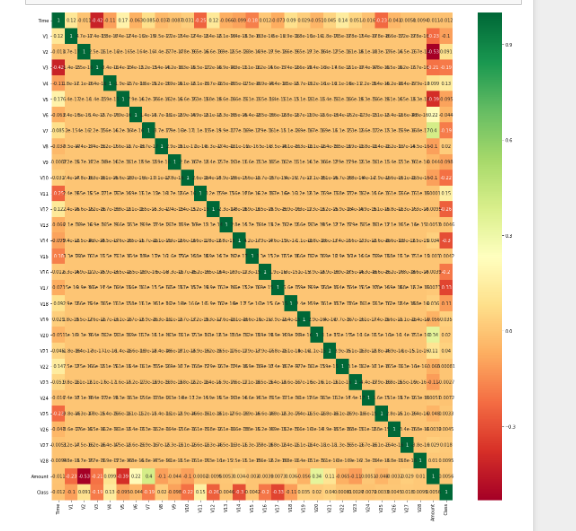
**90MHz or Higher processor**

**Graph**

****

****





**Source code**

**import numpy as np**

**import pandas as pd**

**import sklearn**

**import scipy**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.metrics import classification\_report,accuracy\_score**

**from sklearn.ensemble import IsolationForest**

**from sklearn.neighbors import LocalOutlierFactor**

**from sklearn.svm import OneClassSVM**

**from pylab import rcParams**

**rcParams['figure.figsize'] = 14, 8**

**RANDOM\_SEED = 42**

**LABELS = ["Normal", "Fraud"]**

**data = pd.read\_csv('creditcard.csv',sep=',')**

**data.head()**

**data.info()**

**data.isnull().values.any()**

**count\_classes = pd.value\_counts(data['Class'], sort = True)**

**count\_classes.plot(kind = 'bar', rot=0)**

**plt.title("Transaction Class Distribution")**

**plt.xticks(range(2), LABELS)**

**plt.xlabel("Class")**

**plt.ylabel("Frequency")**

**## Get the Fraud and the normal dataset**

**fraud = data[data['Class']==1]**

**normal = data[data['Class']==0]**

**print(fraud.shape,normal.shape)**

**## We need to analyze more amount of information from the transaction data**

**#How different are the amount of money used in different transaction classes?**

**fraud.Amount.describe()**

**normal.Amount.describe()**

**f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)**

**f.suptitle('Amount per transaction by class')**

**bins = 50**

**ax1.hist(fraud.Amount, bins = bins)**

**ax1.set\_title('Fraud')**

**ax2.hist(normal.Amount, bins = bins)**

**ax2.set\_title('Normal')**

**plt.xlabel('Amount ($)')**

**plt.ylabel('Number of Transactions')**

**plt.xlim((0, 20000))**

**plt.yscale('log')**

**plt.show();**

**# We Will check Do fraudulent transactions occur more often during certain time frame ? Let us find out with a visual representation.**

**f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)**

**f.suptitle('Time of transaction vs Amount by class')**

**ax1.scatter(Fraud.Time, Fraud.Amount)**

**ax1.set\_title('Fraud')**

**ax2.scatter(Normal.Time, Normal.Amount)**

**ax2.set\_title('Normal')**

**plt.xlabel('Time (in Seconds)')**

**plt.ylabel('Amount')**

**plt.show()**

**## Take some sample of the data**

**data1= data.sample(frac = 0.1,random\_state=1)**

**data1.shape**

**data.shape**

**#Determine the number of fraud and valid transactions in the dataset**

**Fraud = data1[data1['Class']==1]**

**Valid = data1[data1['Class']==0]**

**outlier\_fraction = len(Fraud)/float(len(Valid))**

**print(outlier\_fraction)**

**print("Fraud Cases : {}".format(len(Fraud)))**

**print("Valid Cases : {}".format(len(Valid)))**

**## Correlation**

**import seaborn as sns**

**#get correlations of each features in dataset**

**corrmat = data1.corr()**

**top\_corr\_features = corrmat.index**

**plt.figure(figsize=(20,20))**

**#plot heat map**

**g=sns.heatmap(data[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")**

**#Create independent and Dependent Features**

**columns = data1.columns.tolist()**

**# Filter the columns to remove data we do not want**

**columns = [c for c in columns if c not in ["Class"]]**

**# Store the variable we are predicting**

**target = "Class"**

**# Define a random state**

**state = np.random.RandomState(42)**

**X = data1[columns]**

**Y = data1[target]**

**X\_outliers = state.uniform(low=0, high=1, size=(X.shape[0], X.shape[1]))**

**# Print the shapes of X & Y**

**print(X.shape)**

**print(Y.shape)**

**##Define the outlier detection methods**

**classifiers ={**

**"Isolation Forest":IsolationForest(n\_estimators=100, max\_samples=len(X),**

**contamination=outlier\_fraction,random\_state=state, verbose=0),**

**"Local Outlier Factor":LocalOutlierFactor(n\_neighbors=20, algorithm='auto',**

**leaf\_size=30, metric='minkowski',**

**p=2, metric\_params=None, contamination=outlier\_fraction),**

**"Support Vector Machine":OneClassSVM(kernel='rbf', degree=3, gamma=0.1,nu=0.05,**

**max\_iter=-1, random\_state=state)**

**}**

**type(classifiers)**

**n\_outliers = len(Fraud)**

**for i, (clf\_name,clf) in enumerate(classifiers.items()):**

**#Fit the data and tag outliers**

**if clf\_name == "Local Outlier Factor":**

**y\_pred = clf.fit\_predict(X)**

**scores\_prediction = clf.negative\_outlier\_factor\_**

**elif clf\_name == "Support Vector Machine":**

**clf.fit(X)**

**y\_pred = clf.predict(X)**

**else:**

**clf.fit(X)**

**scores\_prediction = clf.decision\_function(X)**

**y\_pred = clf.predict(X)**

**#Reshape the prediction values to 0 for Valid transactions , 1 for Fraud transactions**

**y\_pred[y\_pred == 1] = 0**

**y\_pred[y\_pred == -1] = 1**

**n\_errors = (y\_pred != Y).sum()**

**# Run Classification Metrics**

**print("{}: {}".format(clf\_name,n\_errors))**

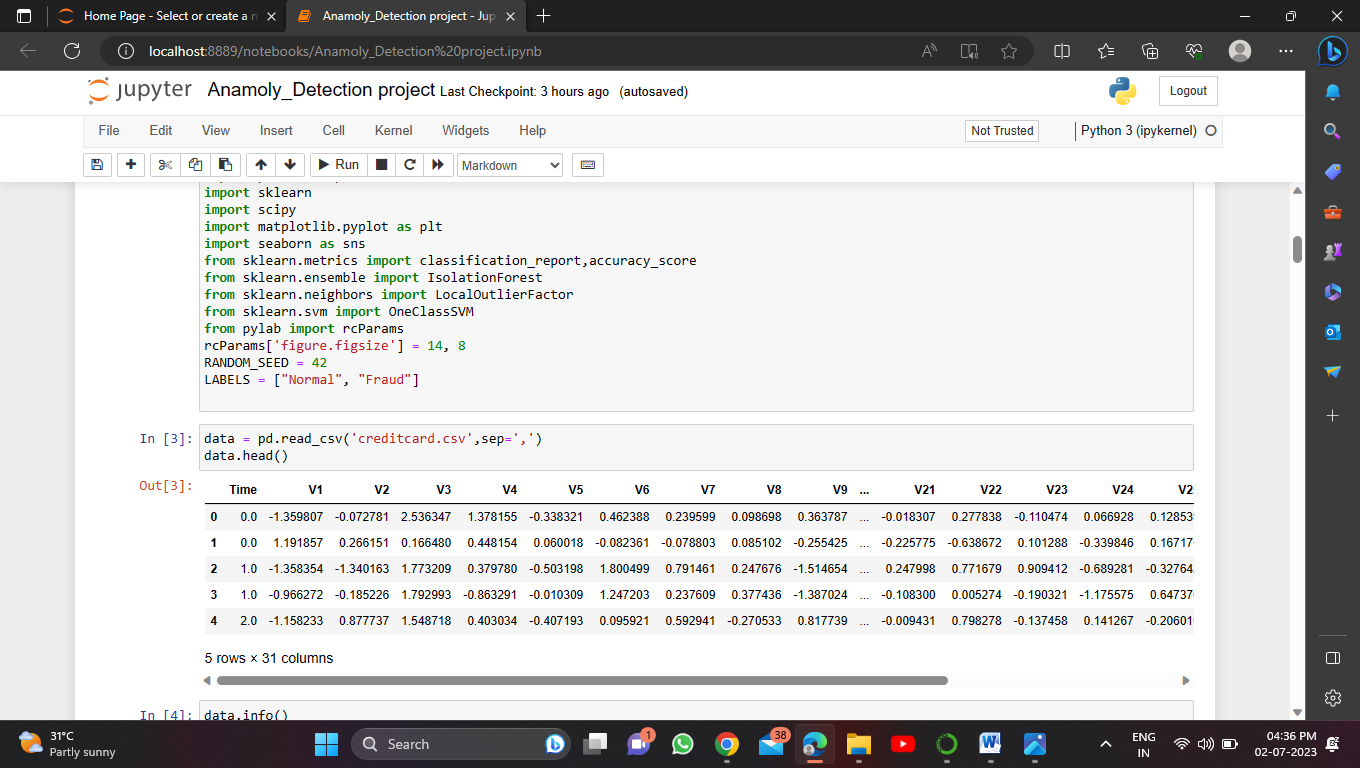
**print("Accuracy Score :")**

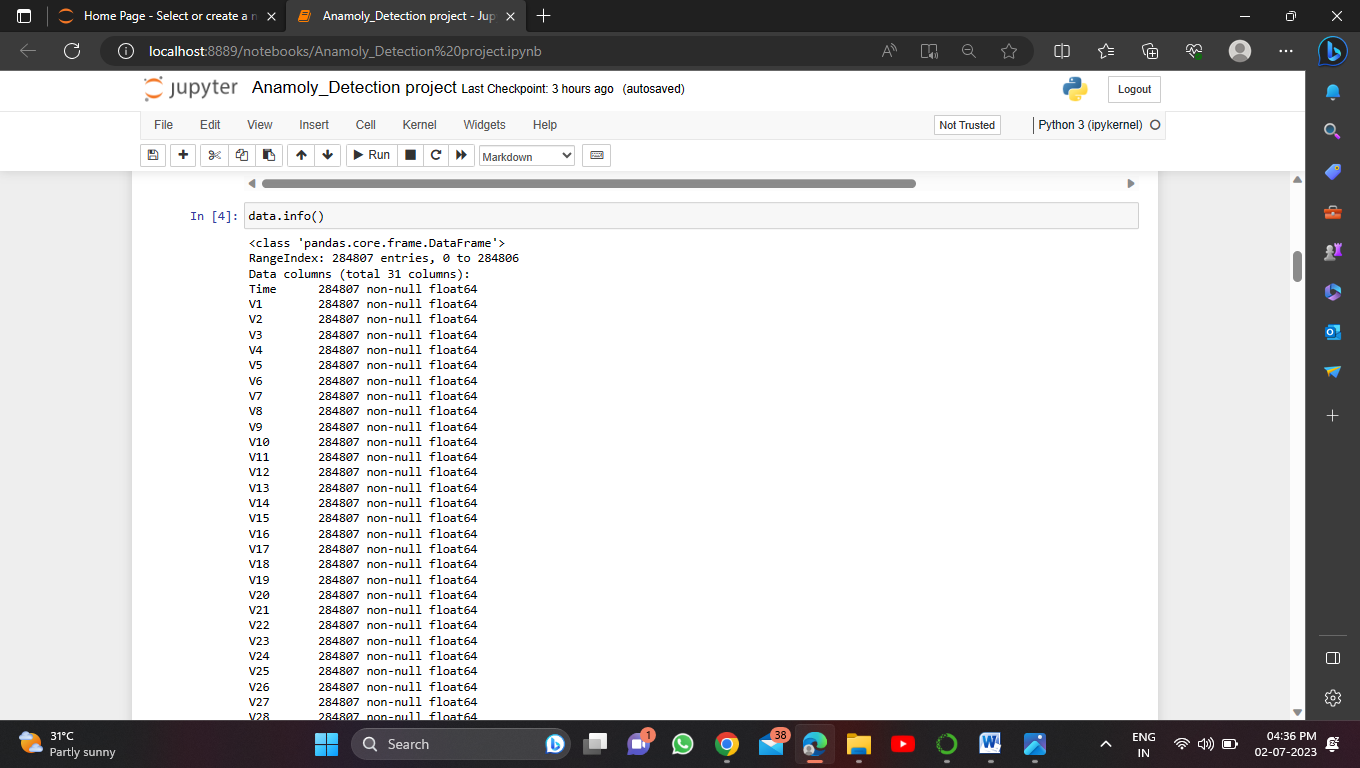
**print(accuracy\_score(Y,y\_pred))**

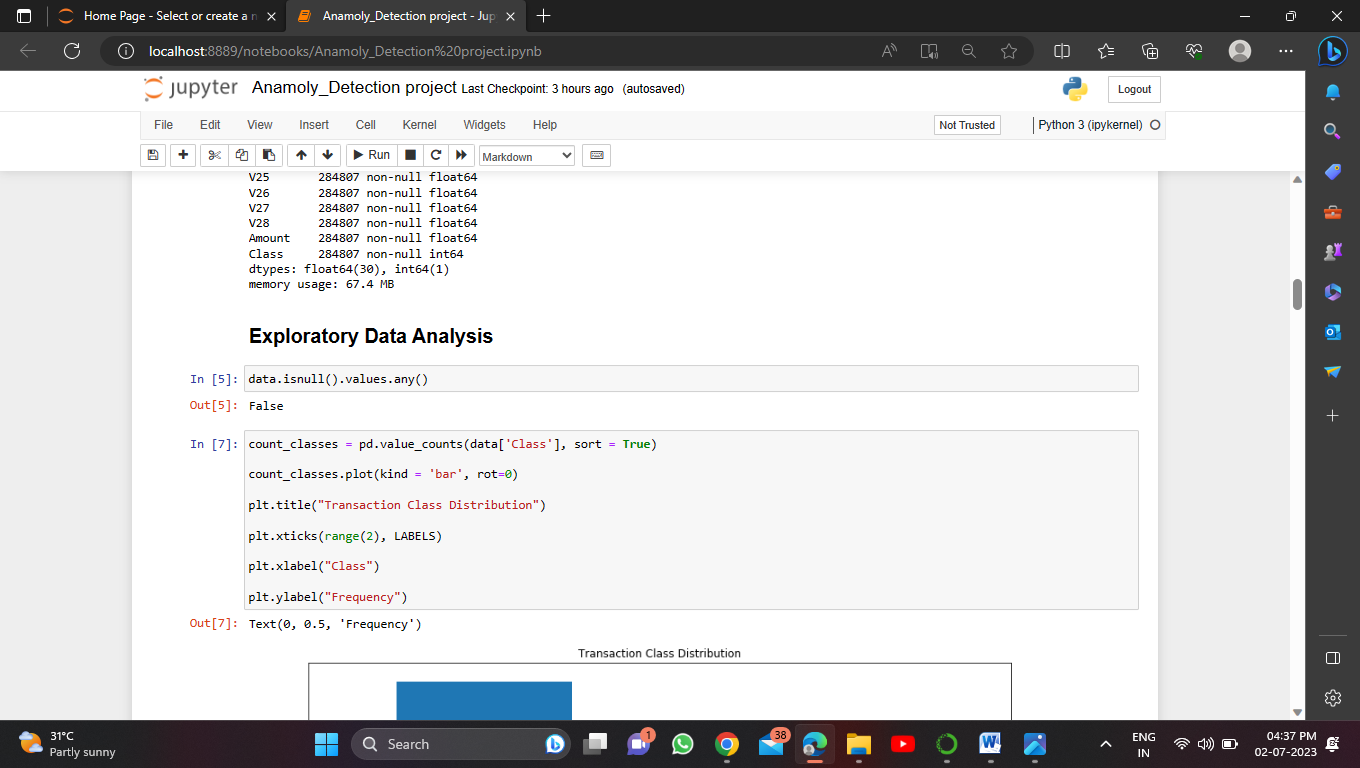
**print("Classification Report :")**

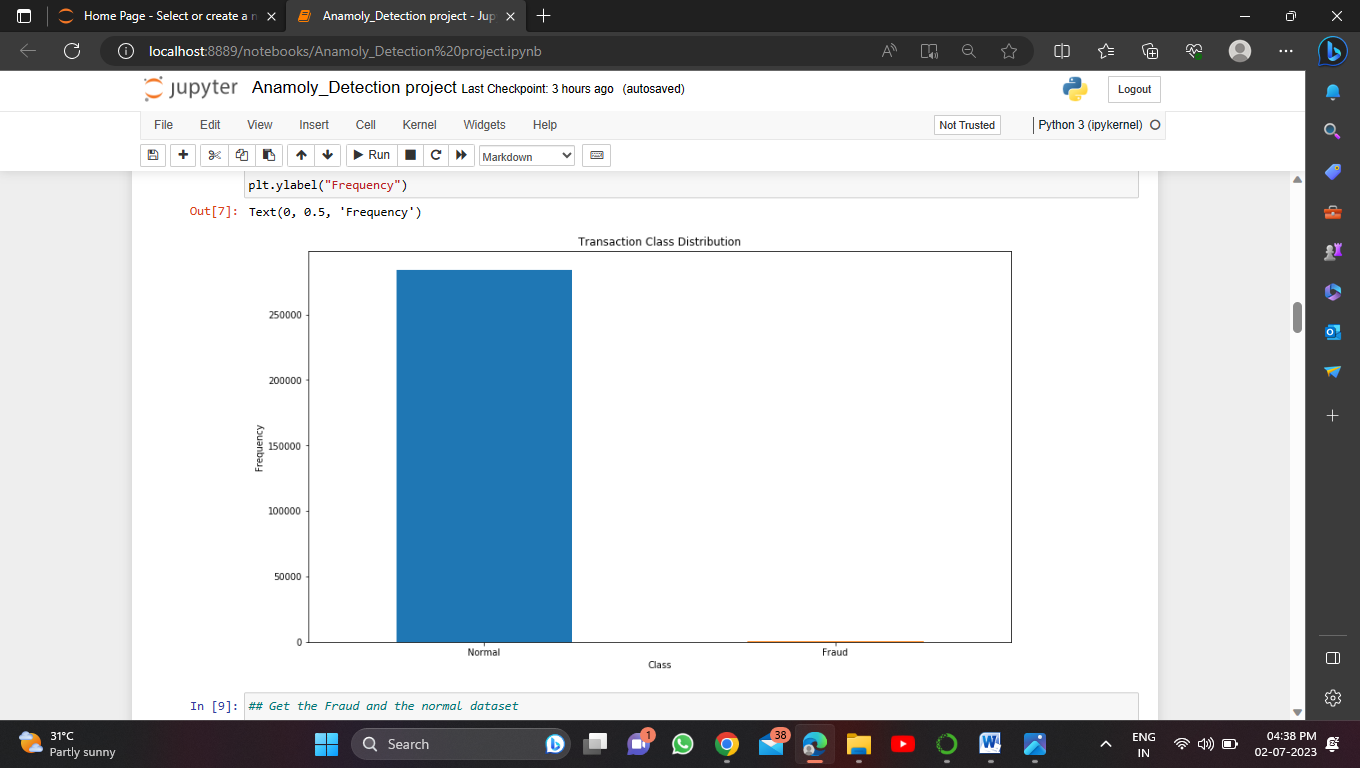
**print(classification\_report(Y,y\_pred))**

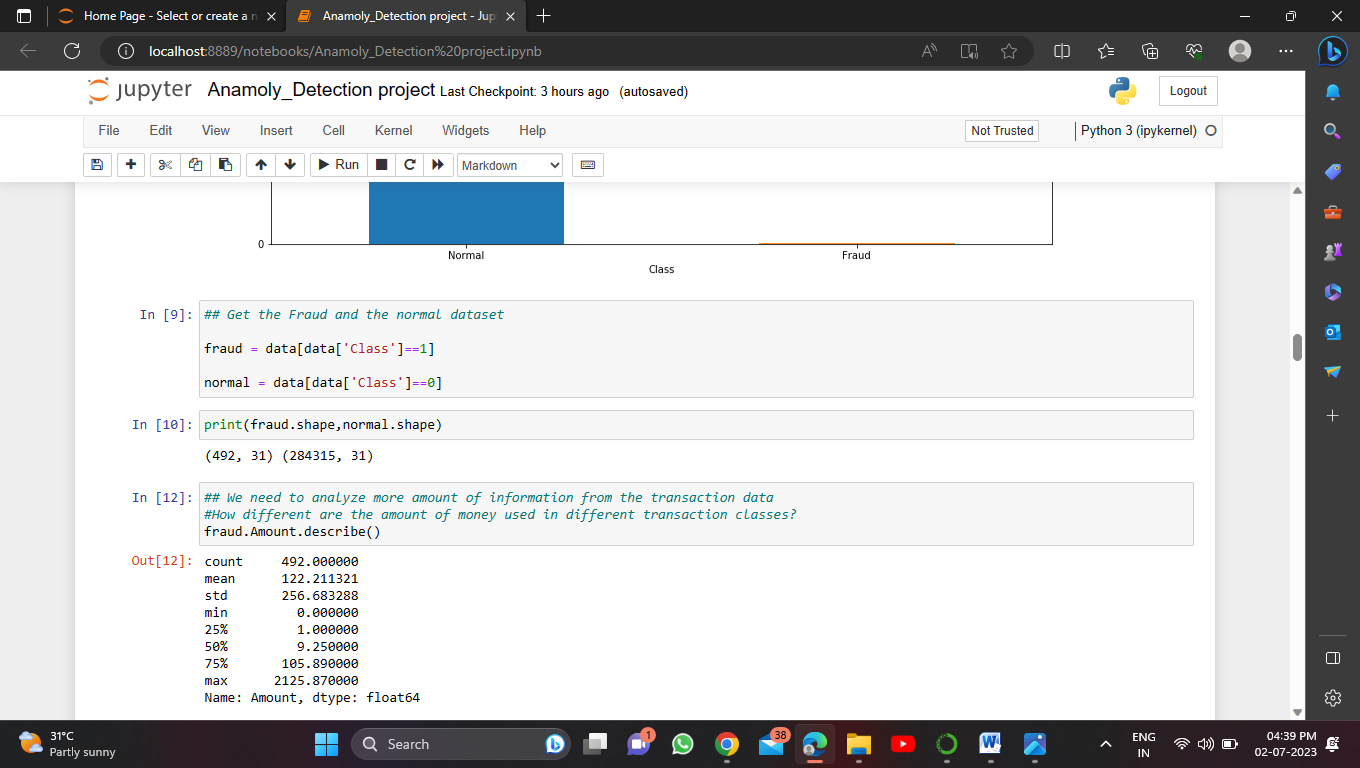
**Screenshots**

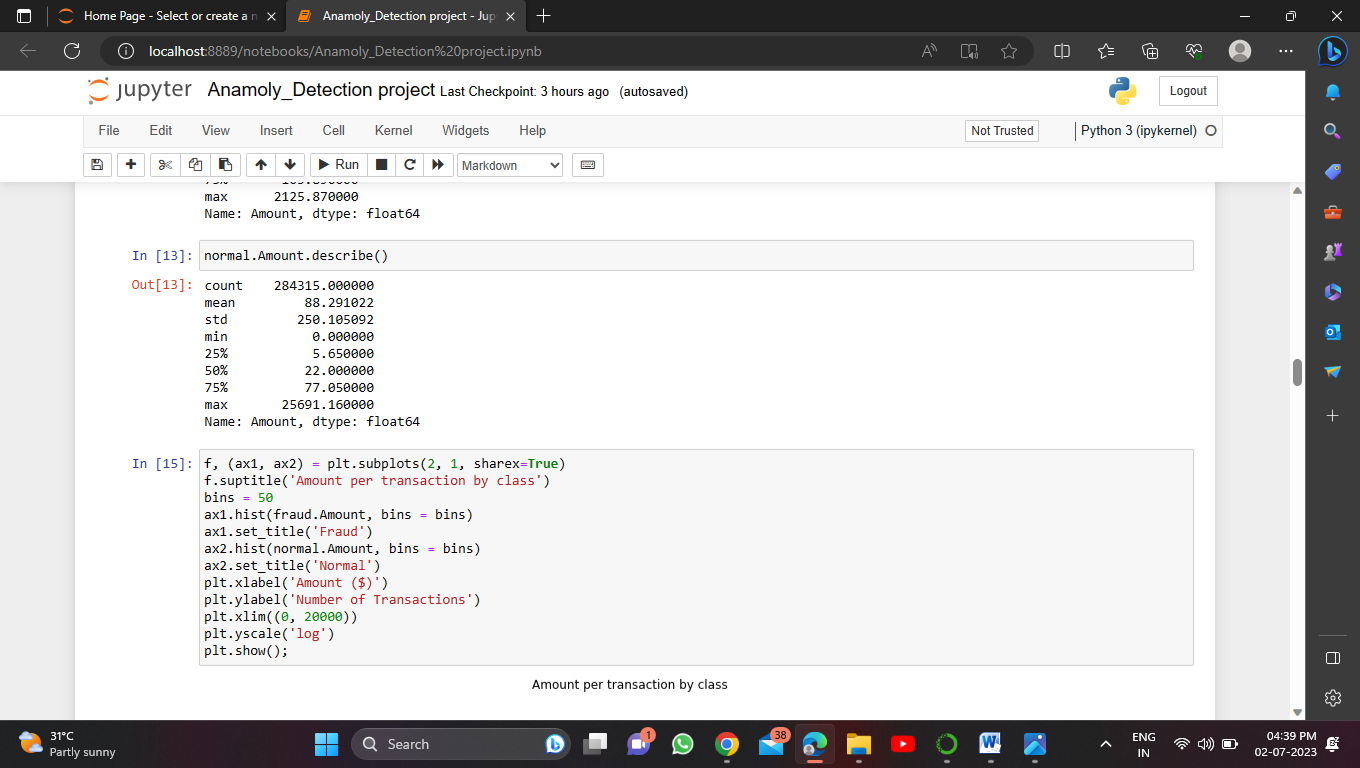


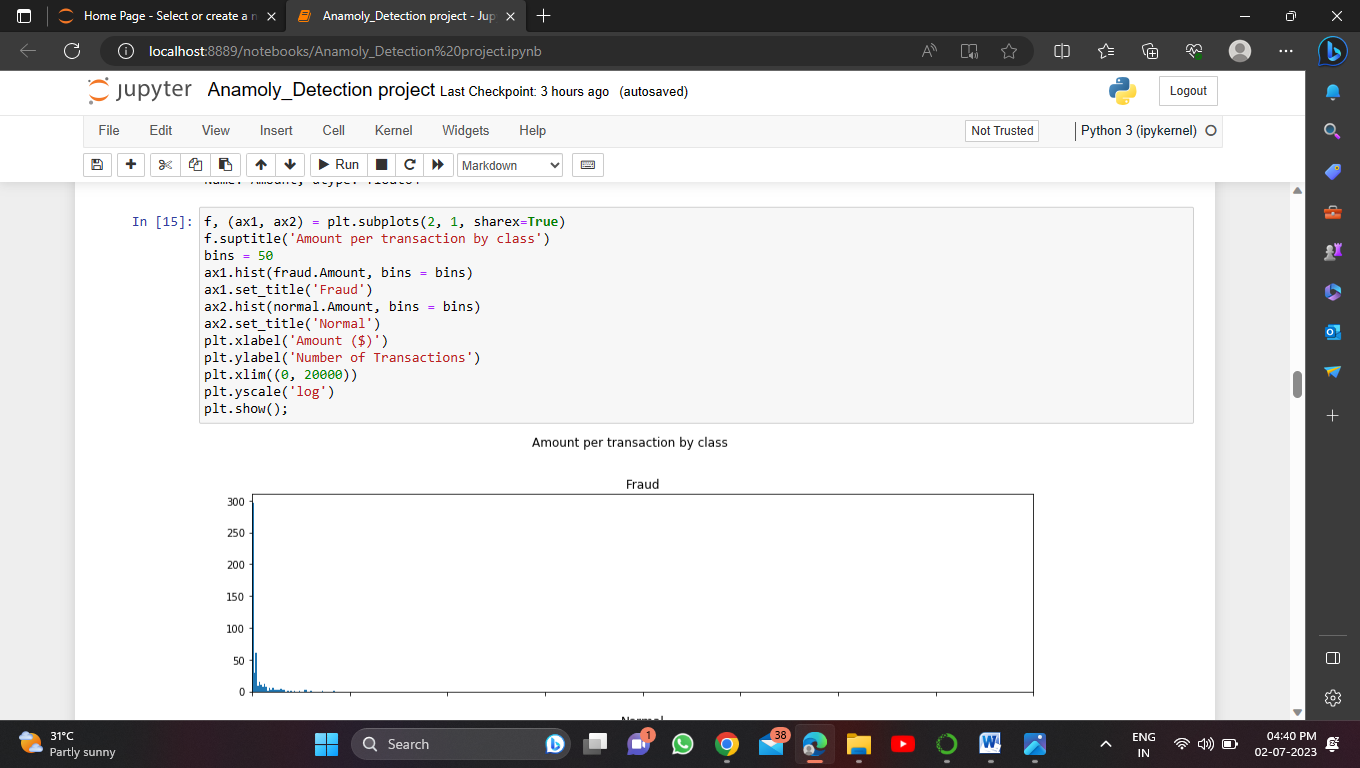


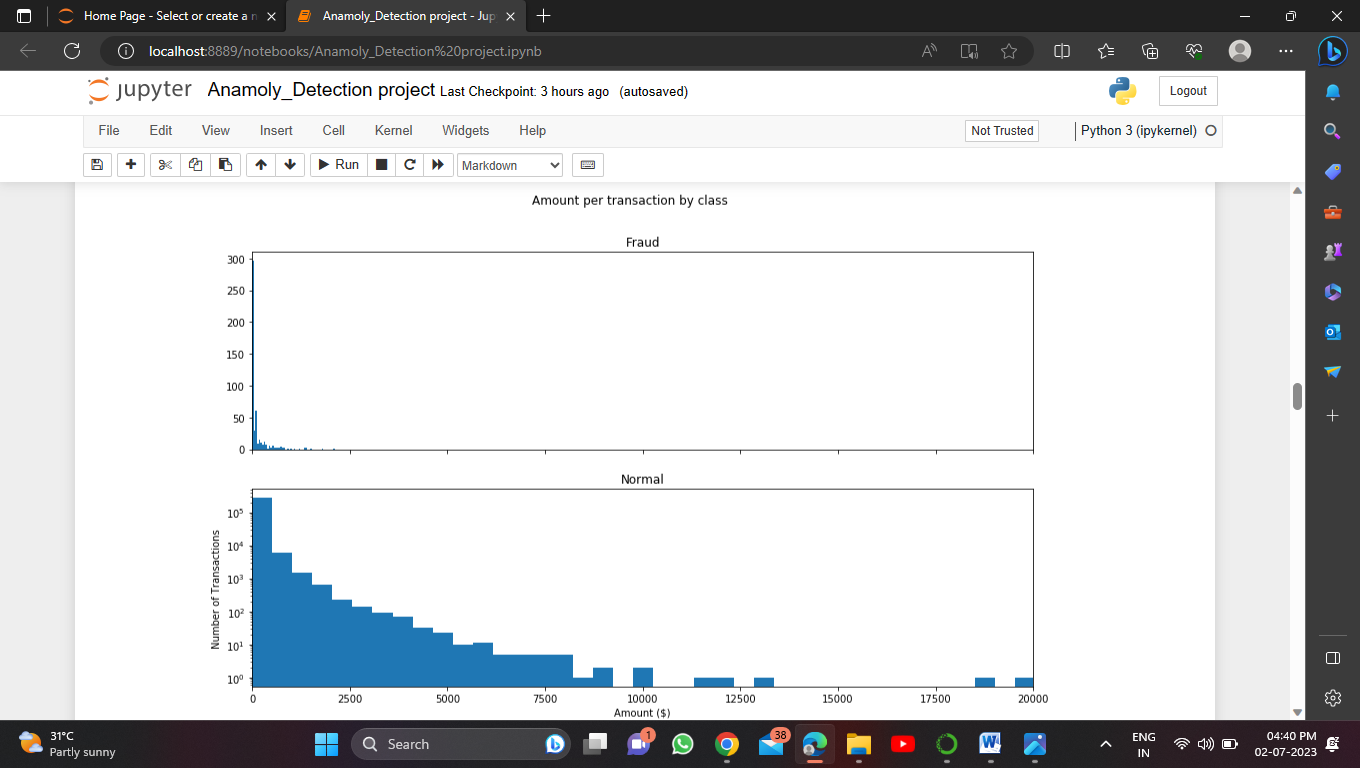


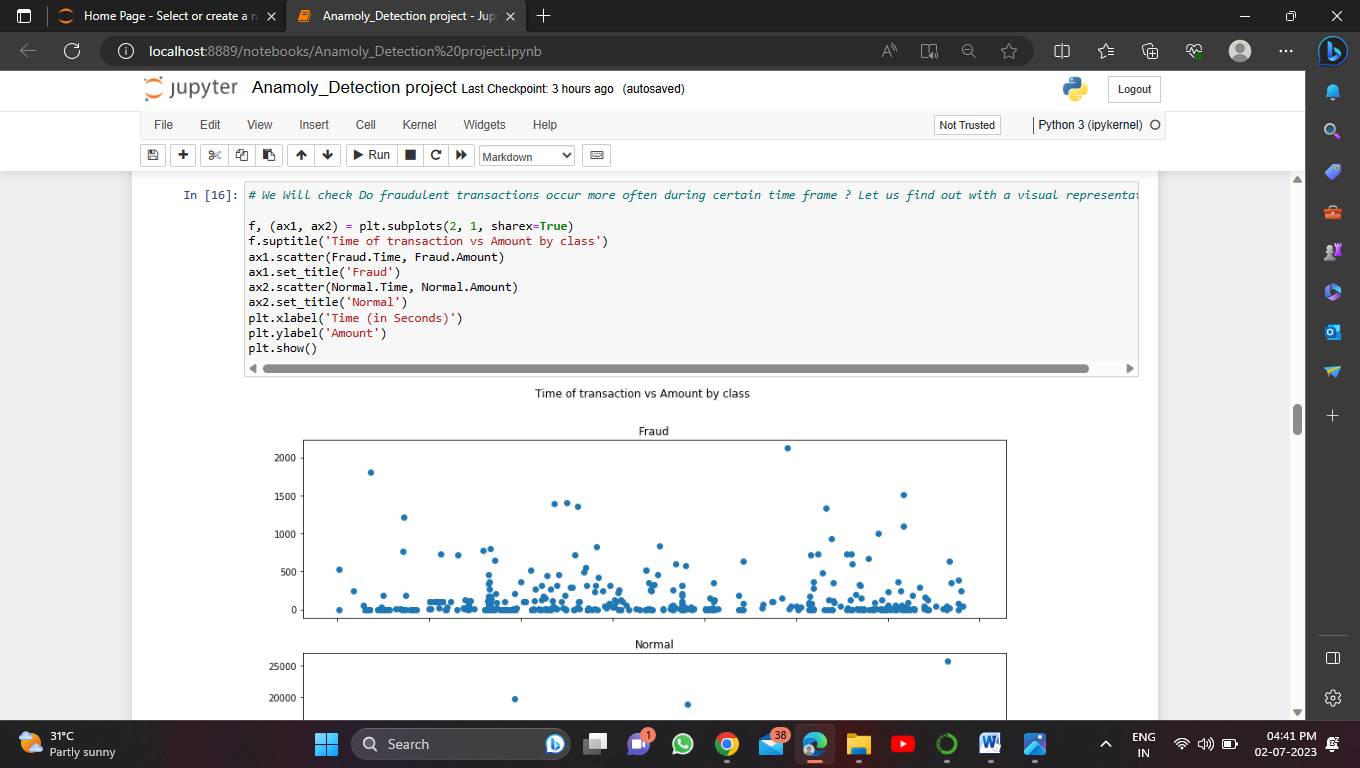


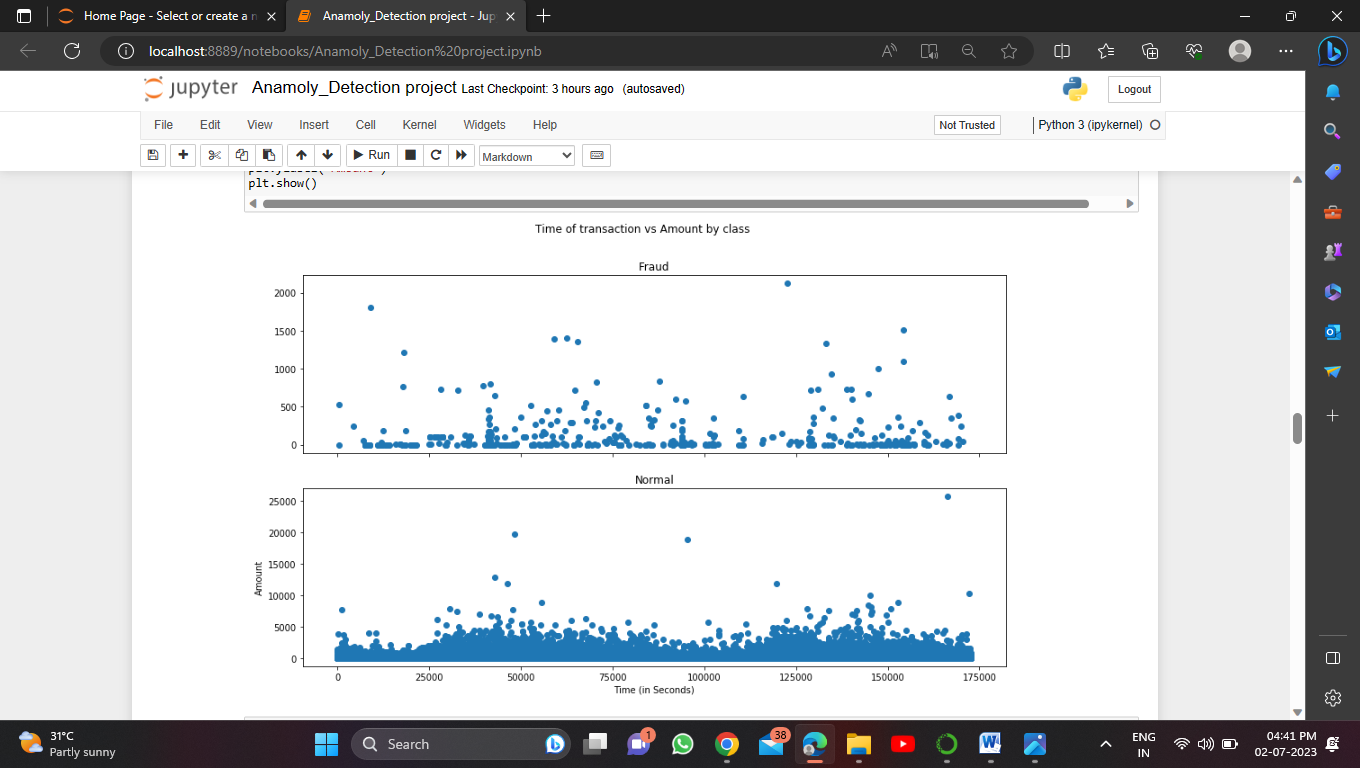


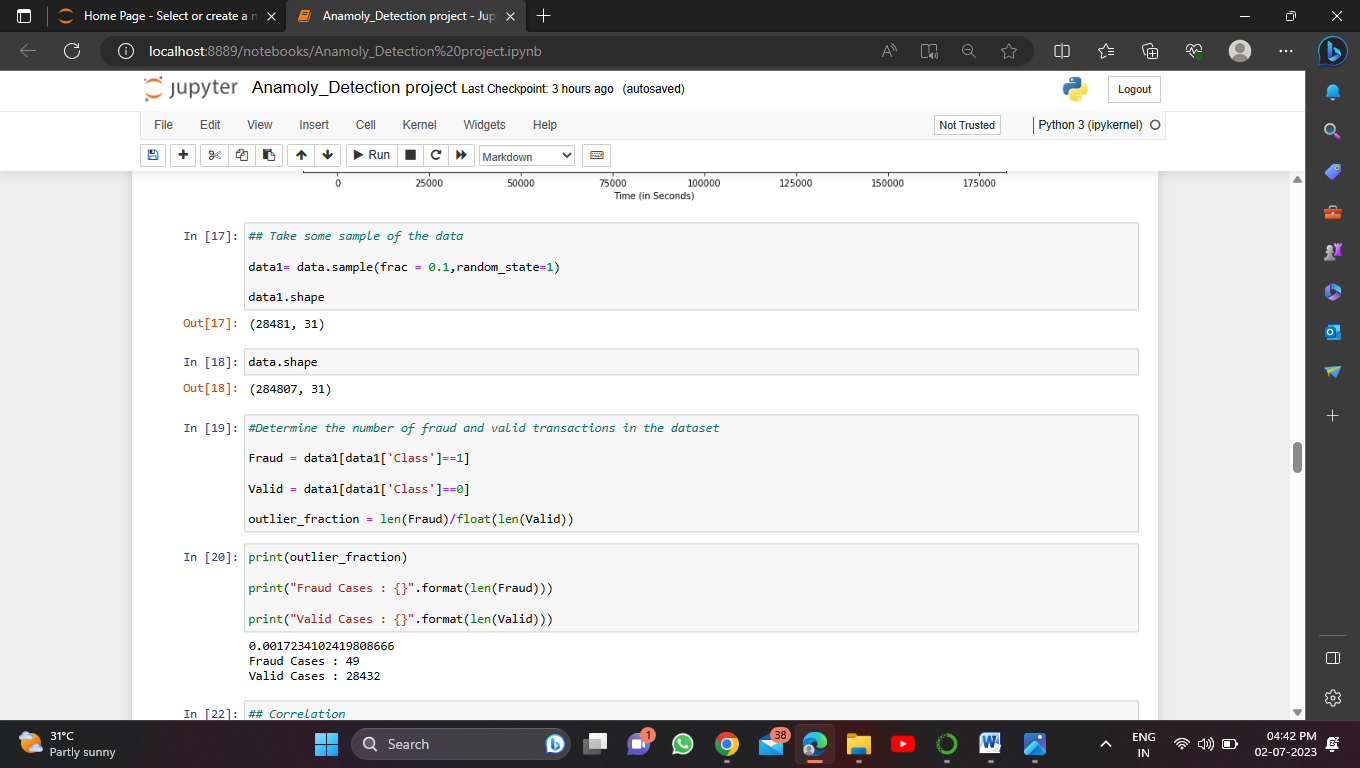


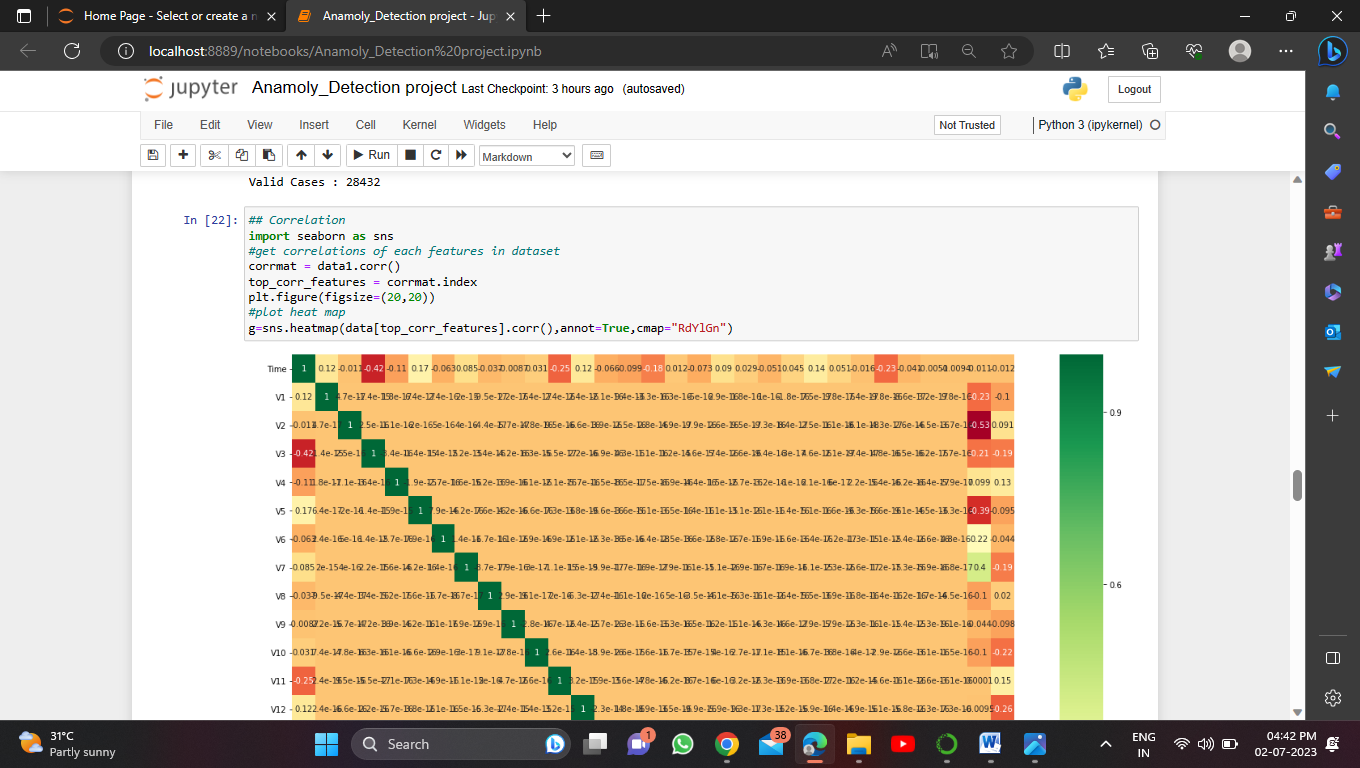


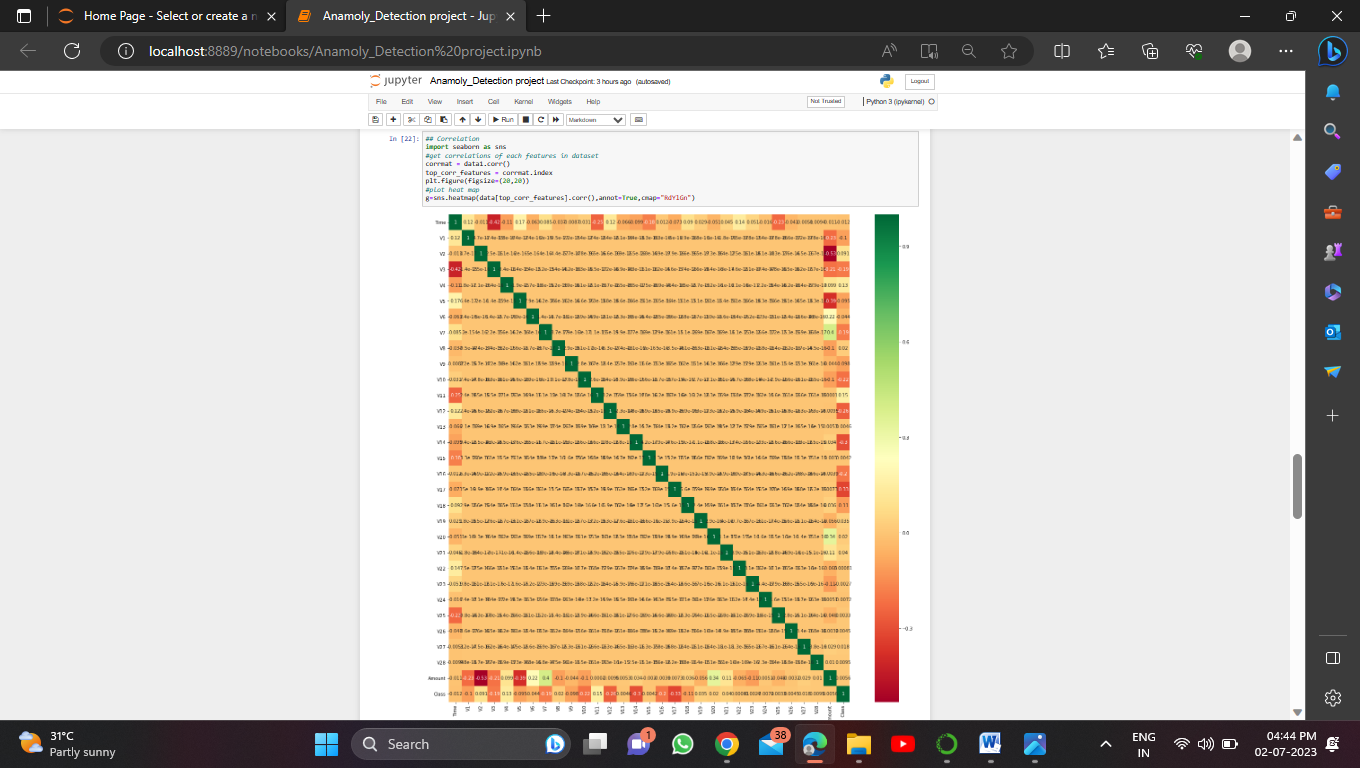


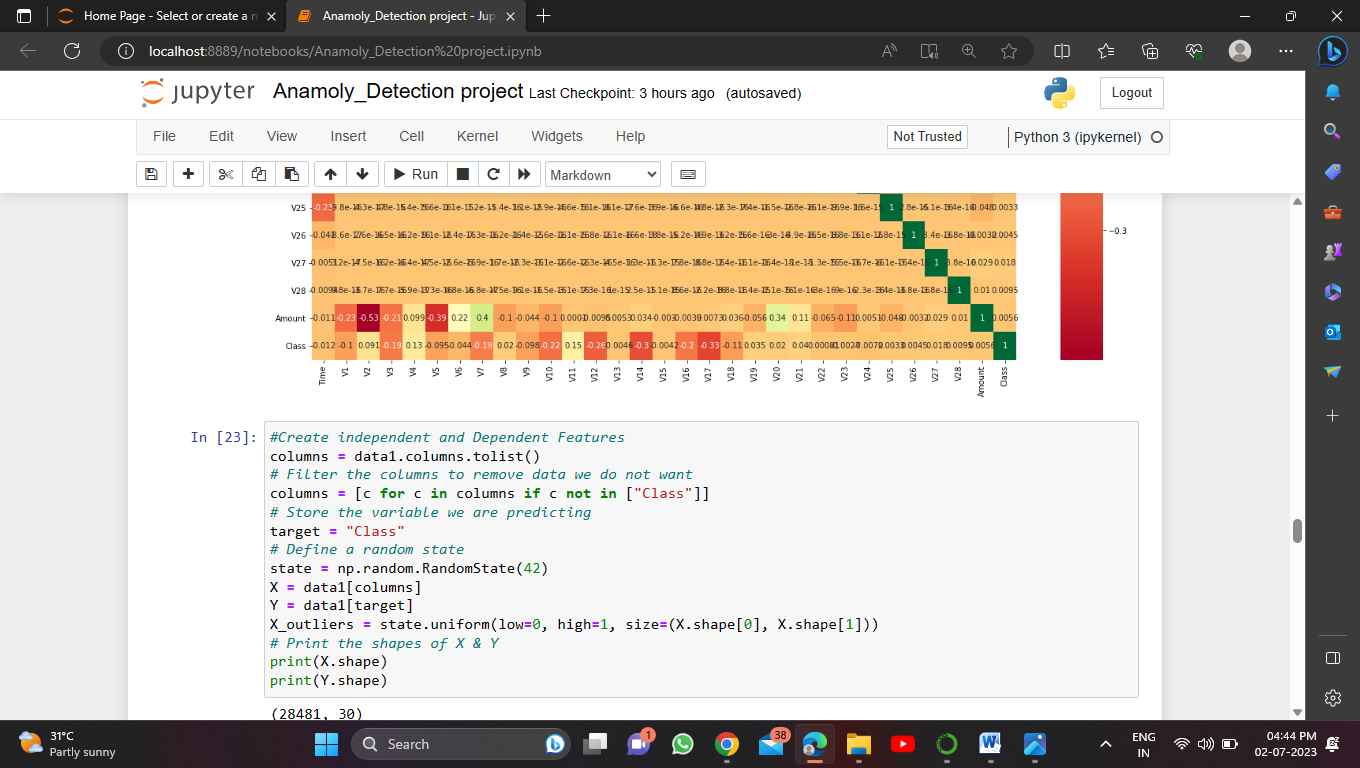


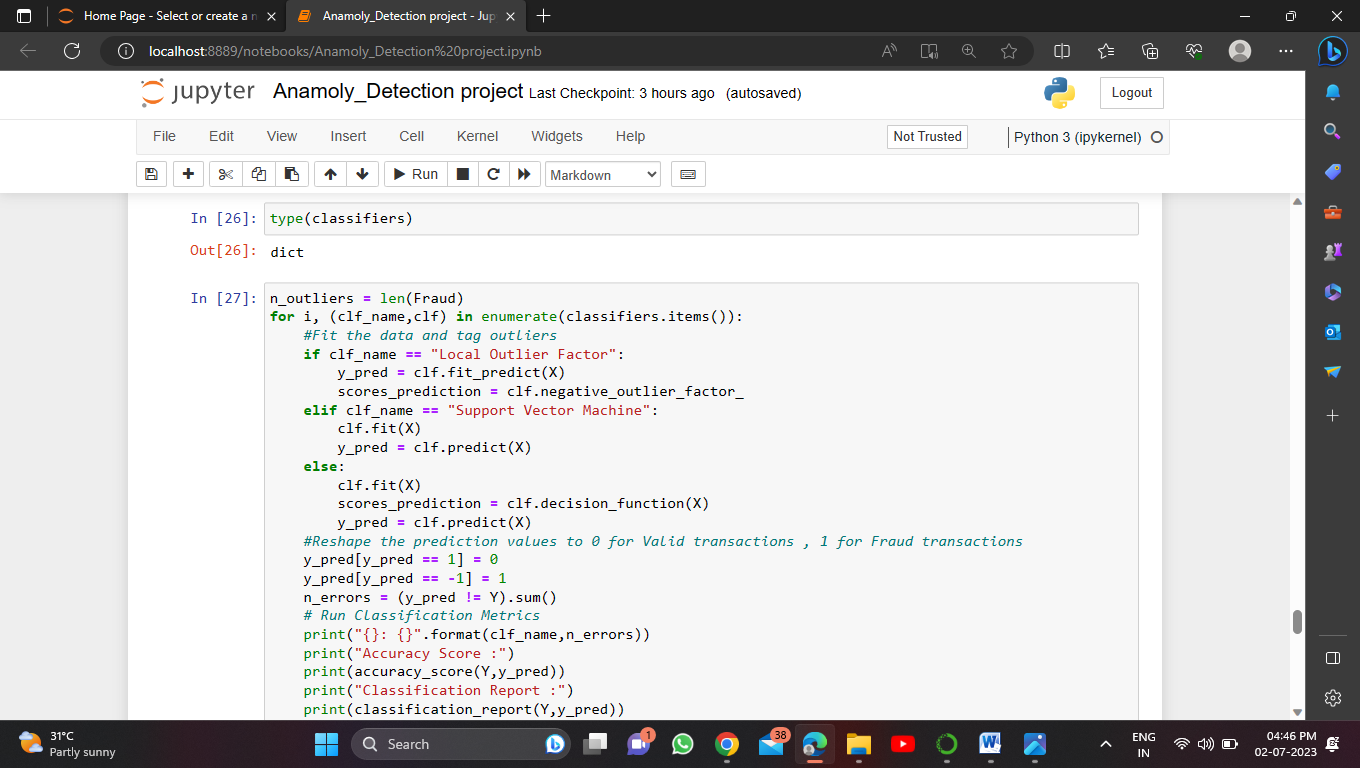


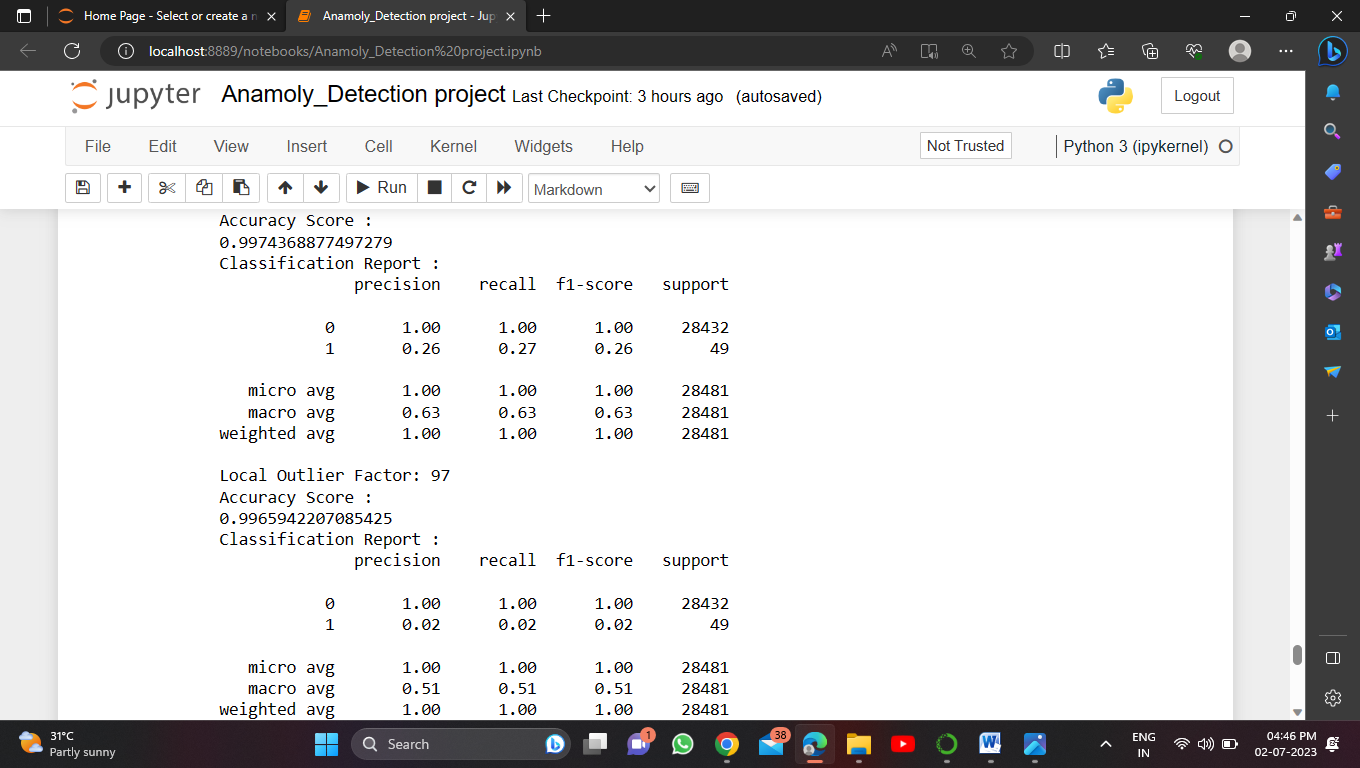


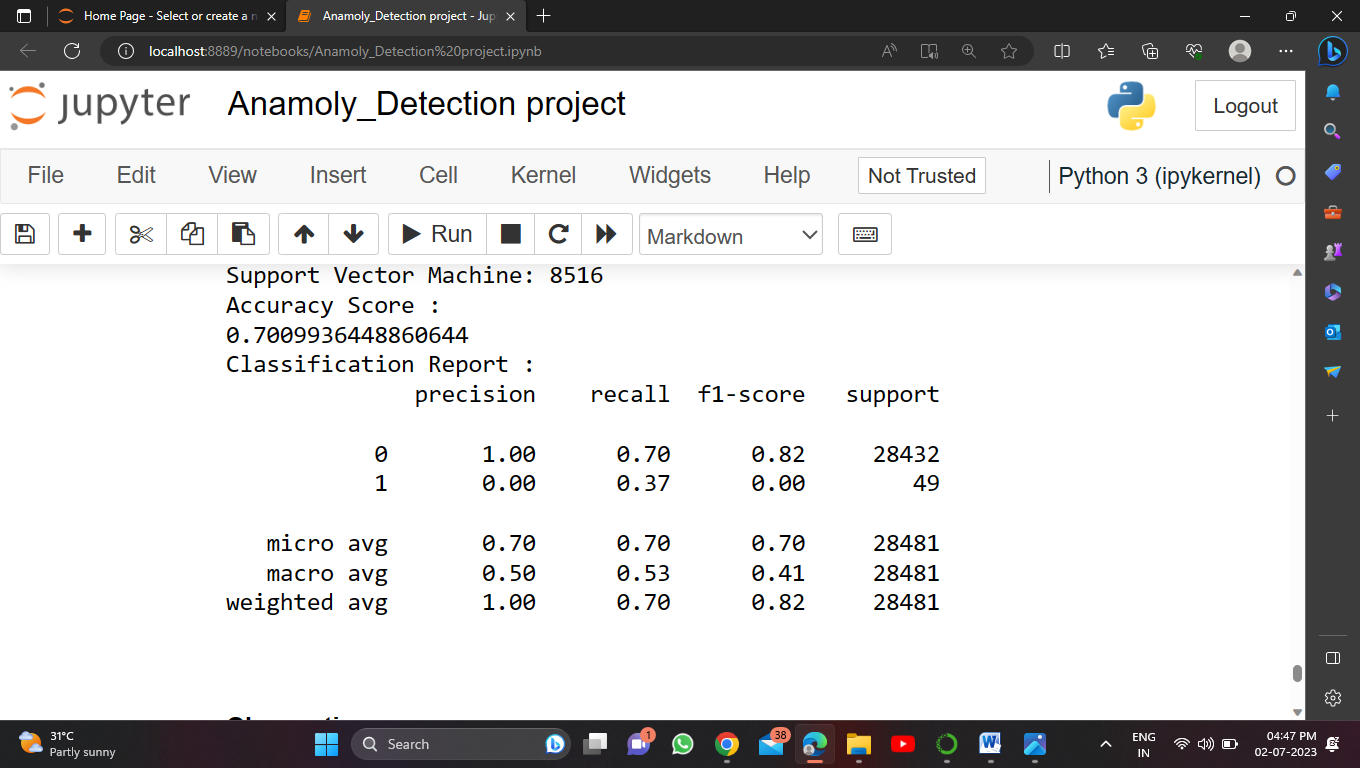












**Conclusion**

* Isolation Forest detected 73 errors versus Local Outlier Factor detecting 97 errors vs. SVM detecting 8516 errors
* Isolation Forest has a 99.74% more accurate than LOF of 99.65% and SVM of 70.09
* When comparing error precision & recall for 3 models , the Isolation Forest performed much better than the LOF as we can see that the detection of fraud cases is around 27 % versus LOF detection rate of just 2 % and SVM of 0%.
* So overall Isolation Forest Method performed much better in determining the fraud cases which is around 30%
* We can also improve on this accuracy by increasing the sample size or use deep learning algorithms however at the cost of computational expense. We can also use complex anomaly detection models to get better accuracy in determining more fraudulent cases.

​

**BIBLIOGRAPHY**

**1. Han J. and Kamber M. (2003): “*Data Mining, Concepts and Techniques*”, Academic Press, 2003.**

**2. Han J., Pei J., and Yin Y. (2000): “*Mining Frequent Patterns without Candidate Generation*”. In proceedings of International Conference on Management of Data (ACM SIGMOD’00), pages 1-12, ACM Press Dallas, TX, United States, May 2000.**

**3. Hand D., Mannila H. and Smyth P. (2001): “*Principle of Data Mining*”. MIT Press, Cambridge, Massachusetts, USA, 2001.**

**4. Hipp J., Guntzer U. and Nakhaeizadeh G. (2000): “*Algorithms for Association Rule Mining: A General Survey and Comparison*”. SIGKDD Explorations, Vol. 2, No. 1, pages 58-64, July 2000.**