CREDIT CARD FRAUD DETECTION

****

|  |  |  |
| --- | --- | --- |
| **Enrollment Number** | **Name** | **Batch** |
| 16103063  16103186  16103144  16103193 | Shivam Kumar Singh  Yasharth Pathak  Abhishek Choudhary  Somil Rastogi | B8  B8  B8  B8 |

Credit card fraud is a wide-ranging term for theft and fraud committed using or involving a payment card, such as a credit card or debit card, as a fraudulent source of funds in a transaction. The purpose may be to obtain goods without paying, or to obtain unauthorized funds from an account.

In proposed system, we present behaviour and Location Analysis (BLA).Which does not require fraud signatures and yet is able to detect frauds by considering a cardholder’s spending habit. Card transaction processing sequence by the stochastic process of a BLA. The details of items purchased in Individual transactions are usually not known to any Fraud Detection System (FDS) running at the bank that issues credit cards to the cardholders. Hence, I feel that BLA is an ideal choice for addressing this problem. Another important advantage of the BLA -based approach is a drastic reduction in the number of False Positives transactions identified as malicious by an FDS although they are actually genuine. An FDS runs at a credit card issuing bank. Each incoming transaction is submitted to the FDS for verification. FDS receives the card details and the value of purchase to verify, whether the transaction is genuine or not. The types of goods that are bought in that transaction are not known to the FDS. It tries to find any anomaly in the transaction based on the spending profile of the cardholder, shipping address, and billing address, etc. If the FDS confirms the transaction to be of fraud, it raises an alarm, and the issuing bank declines the transaction.

The credit card fraud detection features uses user behaviour and location scanning to check for unusual patterns. These patterns include user characteristics such as user spending patterns as well as usual user geographic locations to verify his identity. If any unusual pattern is detected, the system requires re-verification.

The system analyses user credit card data for various characteristics. These characteristics include user country, usual spending procedures. Based upon previous data of that user the system recognizes unusual patterns in the payment procedure.

We have created our project in an anaconda environment utilising the jupyter notebook because through jupyter notebooks we are able to visualise every step, and also it’s a great way to document our code.

We have posted step by step screenshots as to what is happening in our project report.

We have performed the above mentioned task in primarily two basic steps:

* Exploring and understanding the dataset
* Performing the task using two different types of anomaly detection methods

**Exploring and understanding the Dataset**

We begin with downloading the csv file of the dataset **‘creditcard.csv’** hosted on **kaggle.com** and placing the downloaded csv file in our projects folders so that actual dataset can be extracted from the csv file to be used in our project.

We are using the following python libraries in our projects:

* **Numpy**: - A general purpose fundamental array processing package for scientific computation with Python.
* **Pandas**: - Pandas is an open source Python library providing high performance, easy to use data structures and data analysis tools for Python.
* **Matplotlib**: - Matplotlib is a plotting library for python programming language and its numerical mathematics extension Numpy. It provides an Object Oriented API for embedding plots into applications using general purpose GUI toolkits like Tkinter, wxPython, etc.
* **Seaborn**: - It is a Python data visualization library based on matplotlib. It provides high level interface for drawing attractive and informative statistical graphics. Used in this project to do a correlation matrix.
* **Scipy**: - Scipy is a free and open source Python library used for scientific and technical computing. It contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing and other common tasks in science and engineering.

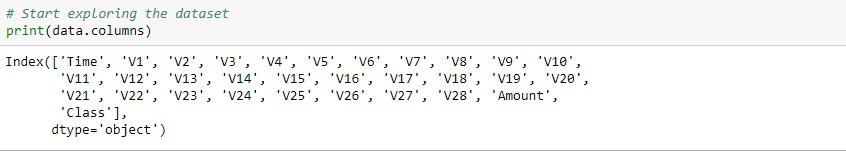
So, we need to **import** them.



Loading the dataset is going to be done by **pandas**.

s2.jpg

After loading the dataset, we start to explore it.

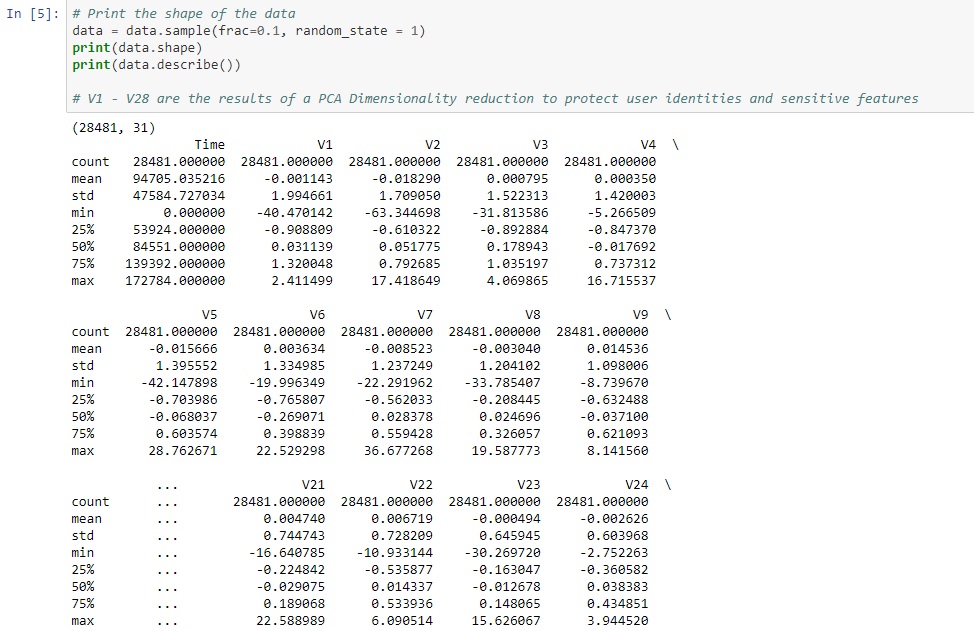


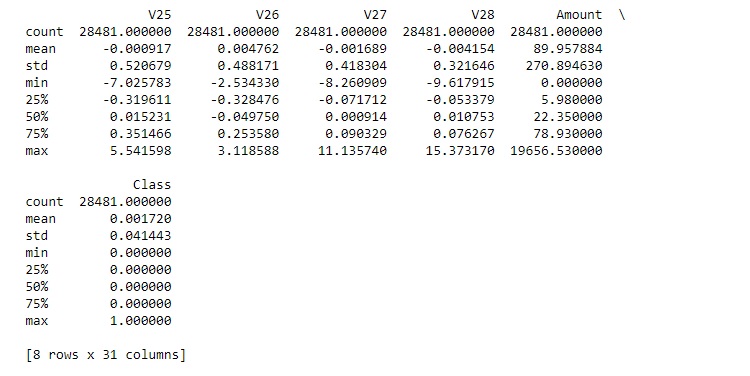
V1 to V28 are actually result of a PCA dimensionality reduction that was used to protect sensitive information in this dataset.

Class here is going to be 0 for a valid transaction and 1 means a fraudulent transaction.

Using data.shape and data.describe() to actually view the data.

We find out there are 284000 entries as rows and 31 columns as attributes (time,amount,class,V1-V28)



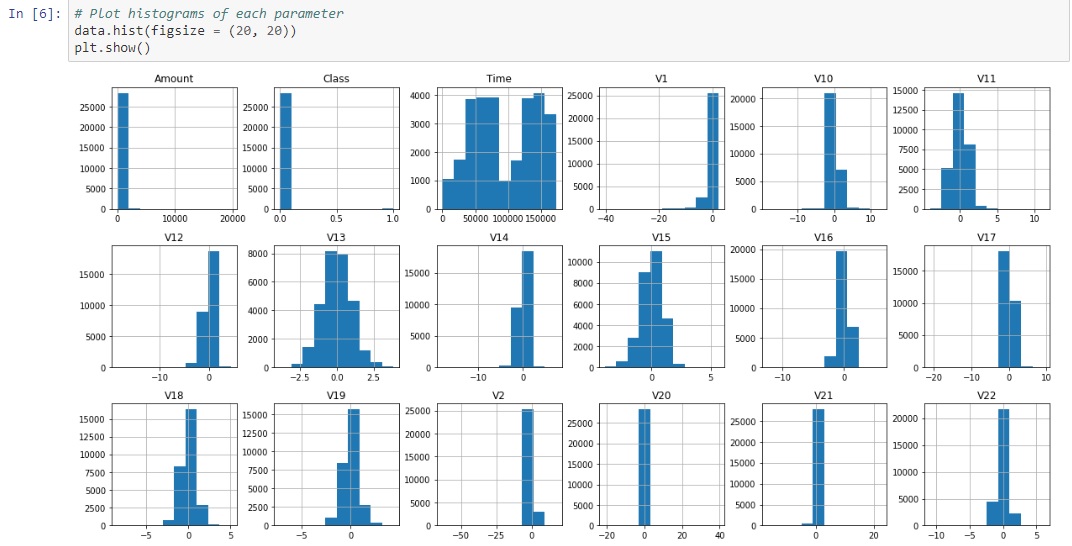
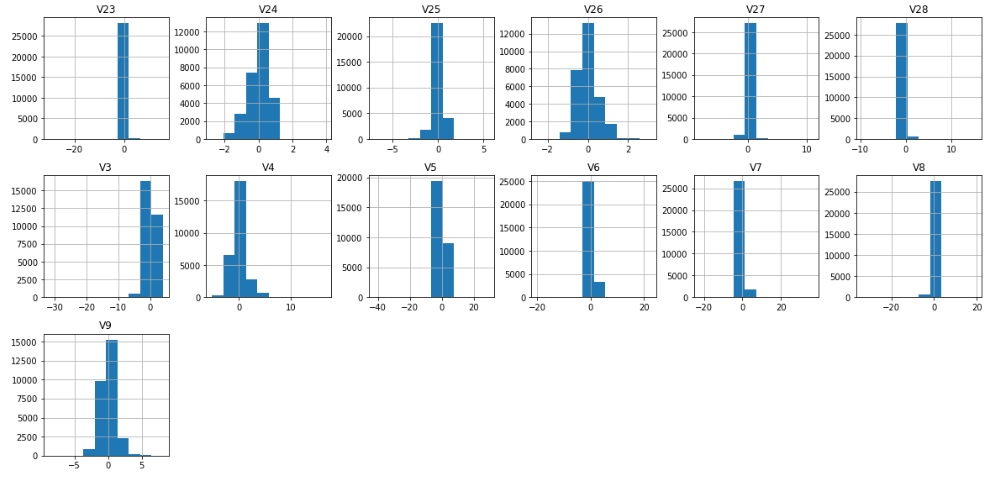


Looking at the class column ,mean is way too much closer to zero which means we have way more valid transactions than fraudulent transactions.

In order to save on time and computational requirement we sample only a fraction of this dataset , hence the value of rows as 28481 which is exactly 10% of the total entries of the dataset.

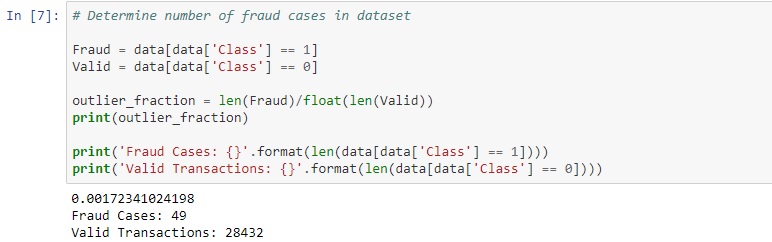
Since it utilises machine learning algorithms, normal approach would be to have as many observations to train our data as possible , but keeping in mind the time required and computational complexity we down sampled the data to 10% and utilised it.

Data exploration visually can be done by means of plotting a **histogram** of each parameter.

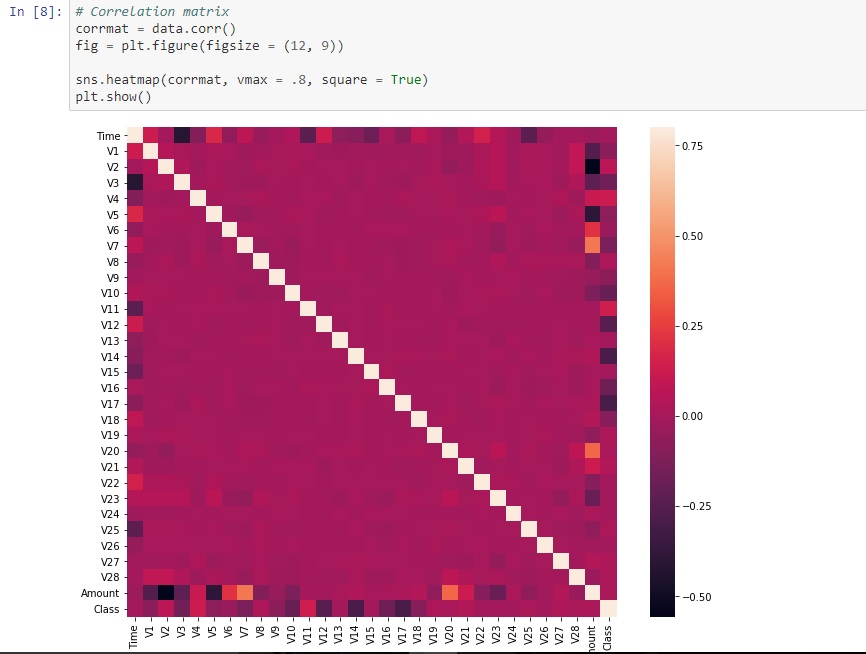
Looking in the class histogram , we see quite close to zero so we have very little fraudulent cases and our dataset mainly consists of valid transactions.

We calculate the actual number of fraudulent and valid cases so we can get an outlier fraction that goes into our future anomaly detection methods.



We get an outlier fraction of 0.0017234 which is actually the result of division of number of fraud cases to the number of valid cases.

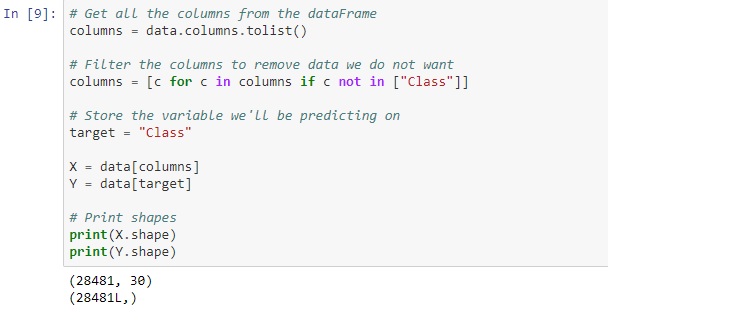
Building a correlation matrix , to see if there is any strong correlation between any of the variables. It also shows us what features are important for overall classification.



Observing the correlation matrix with the heatmap using the seaborn library we earlier imported.

* We see a lot of values really close to zero. So there are not much strong relationships between our parameters.
* However, we do see some variations in the relationships between different parameter and class here.
* Lighter-> positive correlation
* Darker->negative correlation

We require Slight modification of our dataset, which actually involves separating the dataset into X containing all the labels except the class(30 on total) and Y containing only the class label for all the 28000 samples from our dataset.



So now we have completely explored our data and manipulated or formatted it in a way so that our machine learning algorithms can be applied on it.

**Performing the task using two different types of anomaly detection methods**

Now that we have processed our data, we can begin deploying our machine learning algorithms. We will use the following techniques:

* Isolation Forest Algorithm
* Local Outlier factor Algorithm

**Local Outlier Factor (LOF)**

The anomaly score of each sample is called **Local Outlier Factor**. It measures the local deviation of density of a given sample with respect to its neighbours. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighbourhood.

**Isolation Forest Algorithm**

The **Isolation Forest** ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Recursive partitioning creates a decision tree that strives to correctly classify members of the population by splitting it into sub-populations based on several dichotomous independent variables

Since recursive partitioning can be represented by a tree structure, the number of splitting required to isolate a sample is equivalent to the path length from the root node to the terminating node.

This path length, averaged over a forest of such random trees, is a measure of normality and our decision function.

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

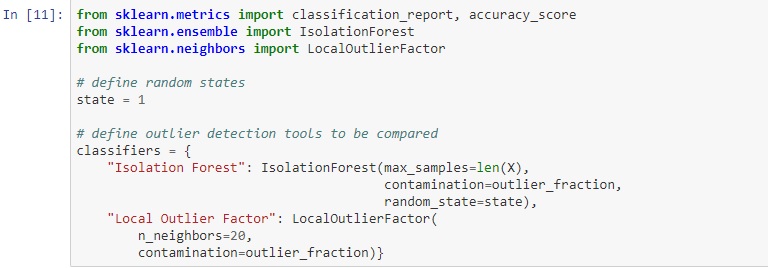
First import the necessary **Sklearn** library which contains these algorithms before actually fitting our data on them.

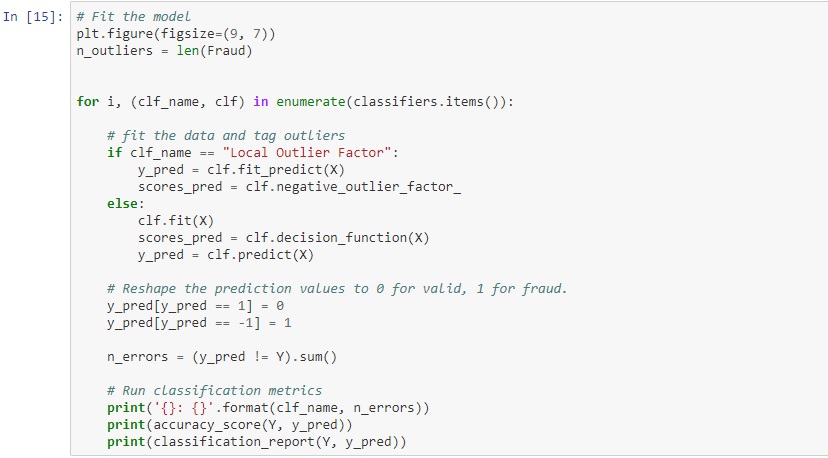
**Scikit-learn** (formerly **scikits.learn**) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language. It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [**NumPy**](https://en.wikipedia.org/wiki/NumPy) and [**SciPy**](https://en.wikipedia.org/wiki/SciPy).

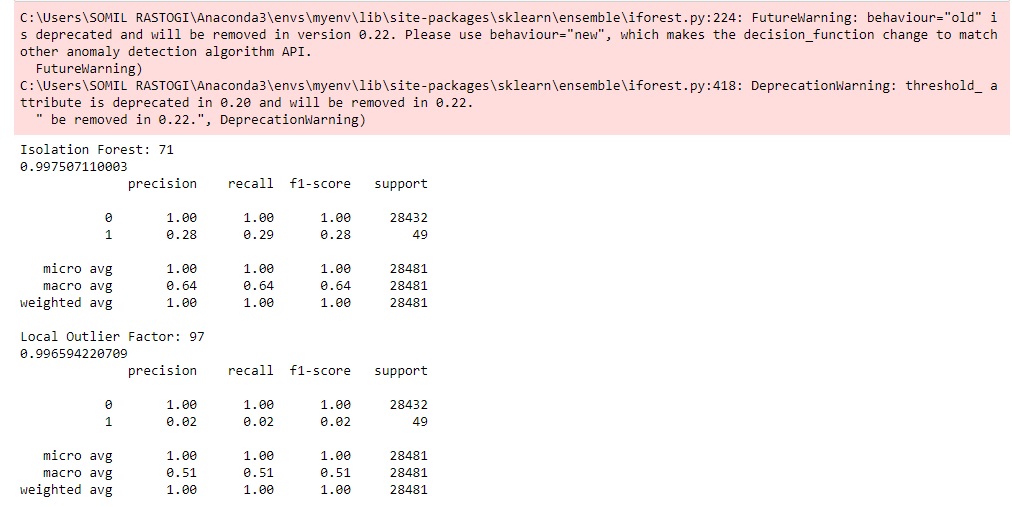
We now perform the actual machine learning part of the project.

Steps involved are:-

* Defining a random state.
* Defining the outlier detection methods by putting them in a dictionary of classifiers.
* Actually fitting the model
* Result the y\_pred is going to give us is -1 for outlier and +1 for inliers, so we need to process it before comparing to our class labels which are 0 for valid and 1 for fraudulent transactions.
* We do a comparison of y\_pred to Y which is our target.
* Then we run the classification metrics to observe our results.







**Precision** here accounts for the cases which are false positives whereas **Recall** accounts for the cases which are false negatives.

Upon comparing the two algorithms we have applied, we come to see that isolation forest did a much better task in comparison to Local Outlier factor. Although both gave a accuracy percentage of close to 99 percent its upon the comparison of precision and recall through which we are able to distinguish between the two algorithms.

Local outlier factor gave a precision of 0.02 in recognising fraudulent cases as evident from the result screenshot ,So that means that we have very few actual fraudulent cases that are getting labelled as fraudulent cases.

However with the case of isolation forest algorithm , So we had 99 percent ninety nine point seventy five percent accuracy but we had a precision of 30 percent. So that's a lot better than 0.02 but still we're only correctly identifying about 30 percent of our actual fraudulent cases.

Upon further digging in and comparing with various algorithms as posted on the data-science website **kaggle** we came to the final conclusion that **Isolation Forest Algorithm** works best in this case.