FINAL PROJECT REPORT: ANTI-MONEY LAUNDERING

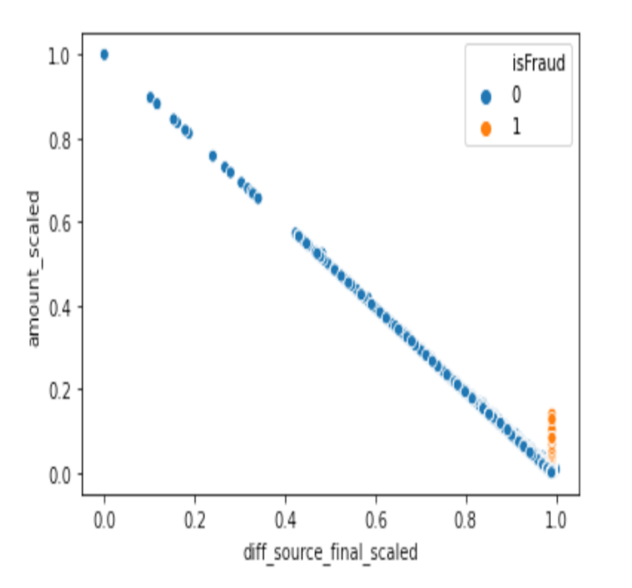
**Abstract:**

Money laundering is the process of masking the origin of income generated by various illegitimate sources like drug trafficking, terrorist funding, illegal arms trade, etc. The total amount money laundered in the United States alone estimates to $800 billion to $2 trillion every year and the current financial systems and law enforcements can catch an underwhelming 1% of this amount.

This is because of the ever-evolving nature of money laundering techniques which are getting smarter by the day, in contrast to the Anti-Money-Laundering techniques adopted by the financial systems today. Modern money laundering techniques like smurfing easily outsmart these systems and move a significant amount of "dirty money" while still flying under the radar. In our project we aim to apply statistical data mining and Machine Learning techniques on a very large network of financial transactions in hopes of finding an algorithm which can pick out fraudulent transactions from this huge swarm of transactions.

**Project Work:**

Firstly, we carried out basic exploratory data analysis (EDA) to get the data understanding. Out of all the transactions types: Cash\_Out, Debit and Transfer implies that the amount was from transferred from source account to destination account balance while for Cash\_In it was vice-versa. The count of fraudulent transactions is proportionately very low compared to non-fraudulent transactions which makes our dataset very skewed. Considering this fact, we tried to apply Anomaly Detection which is the most appropriate and popular machine learning algorithm for skewed data. We applied feature engineering which is carried out in the function named preprocessing in the jupyter code. We tried to deduce different features which could very well segregate the fraudulent and non-fraudulent transactions. This step could have helped divide the dataset as fraud transactions. According to the transaction type, the final account balance of destination accounts and source accounts was calculated. We could see the discrepancy between the final balance calculated and the figures on the dataset. We made two features each to capture this inconsistency in the course and destination accounts. And with the help of them, we were finally able to segregate the dataset using two features: amount transferred and measure of discrepancy of final balance of source account (diff\_source\_final\_scaled) which is shown in below figure.



But on calculating the probabilities, there was not any clear detection of fraud transactions. Hence, we moved on to random forest technique (with 50 trees) which is one of the most accurate learning algorithms and runs efficiently on large dataset just like this dataset.

The result of random forest implementation is given below:

**Confusion matrix**: [[165680 0]

[ 5 498]]

**Evaluation Matrics:**

**precision** **recall** **F1-score** **support**

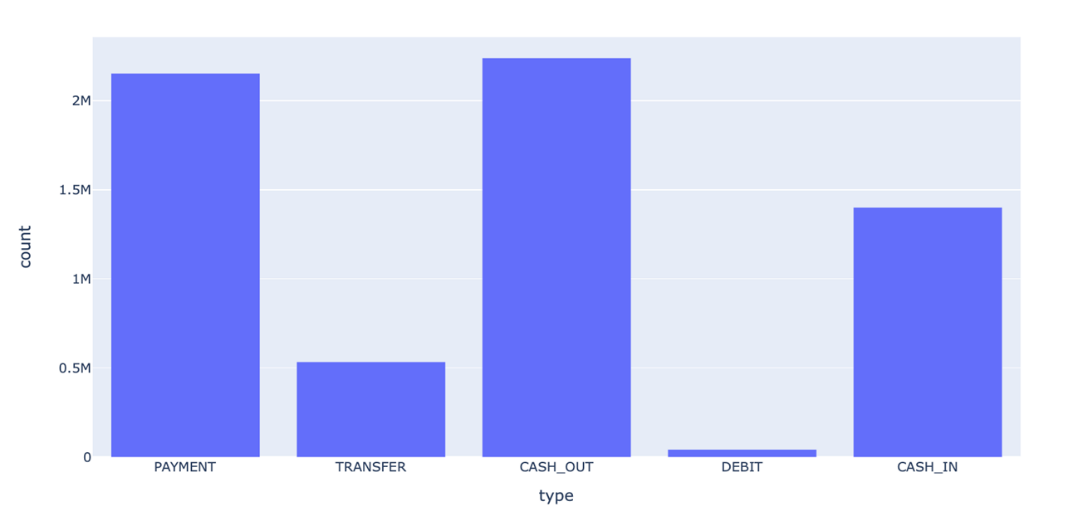
0 1.00 1.00 1.00 165680

1 1.00 0.99 1.00 503

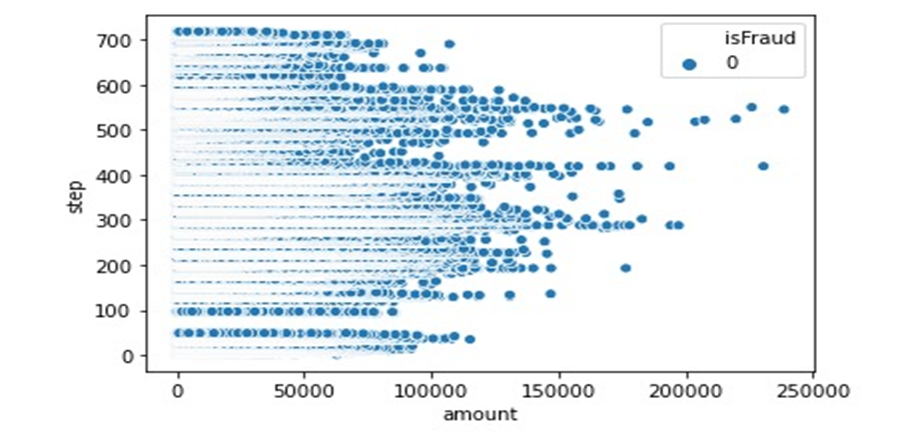
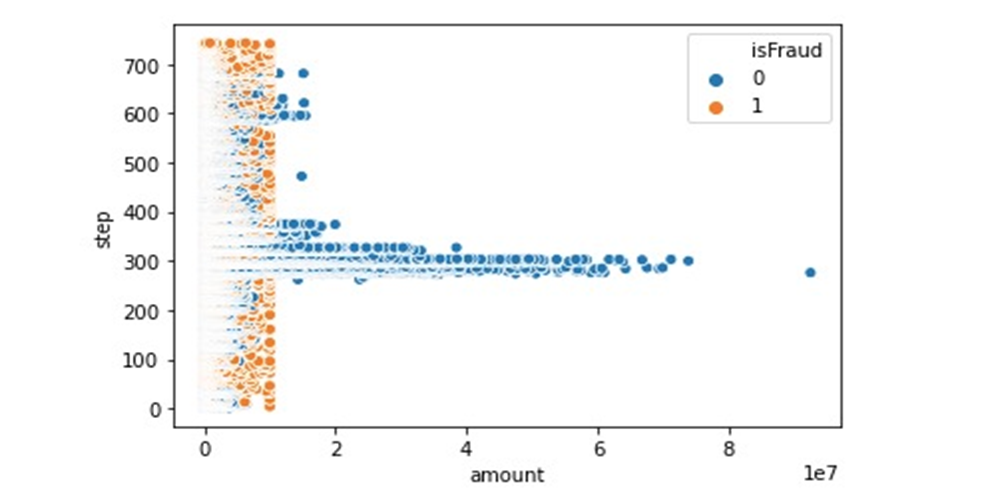
**Conclusion:**

**Analysis: Basis Statistics of Data**

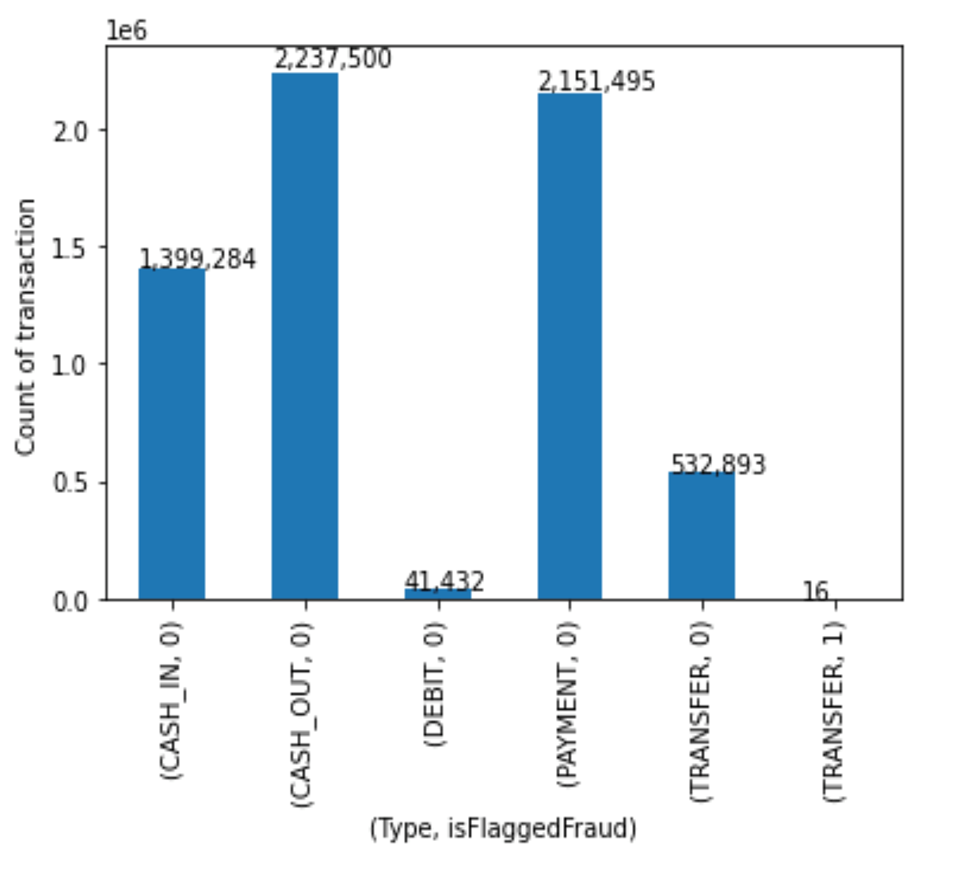
1. Count distribution of type of transactions initiated by customers shows that Cash\_Out and Payment types constitute highest transaction count of more than 2 million.



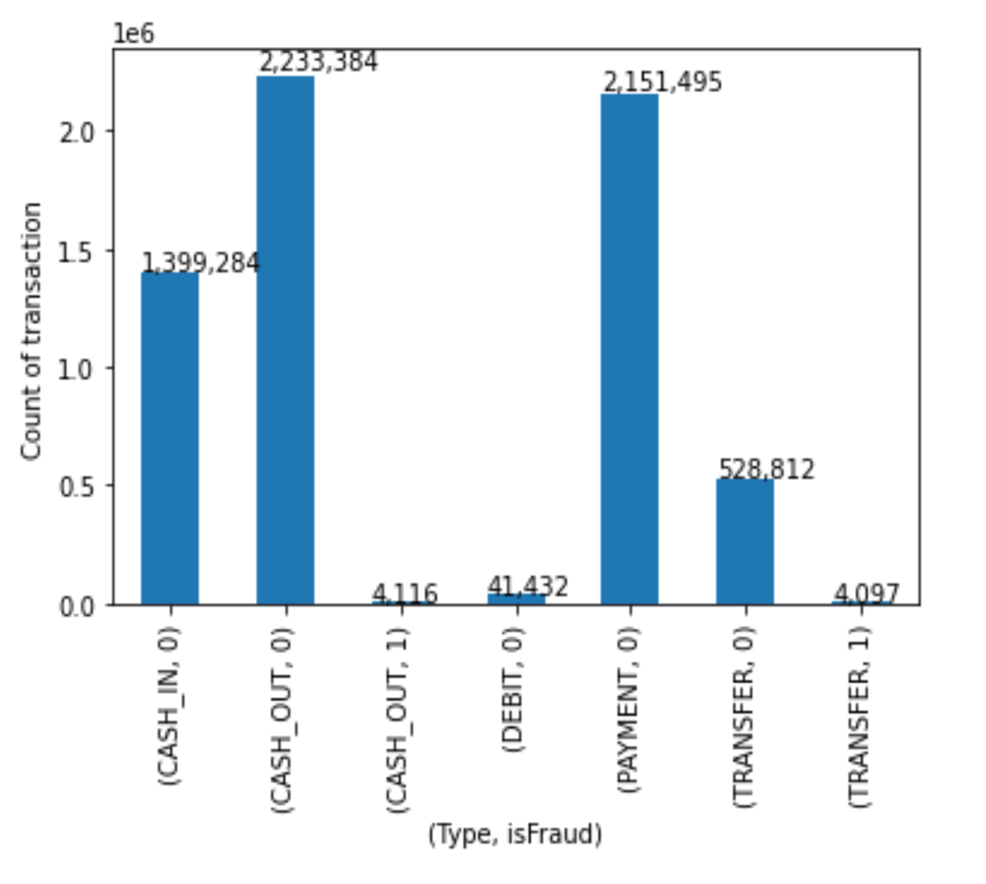
1. All fraud transactions are found in customer transactions only as per the below graphs which shows amounts transferred stepwise for merchant and customers transactions.



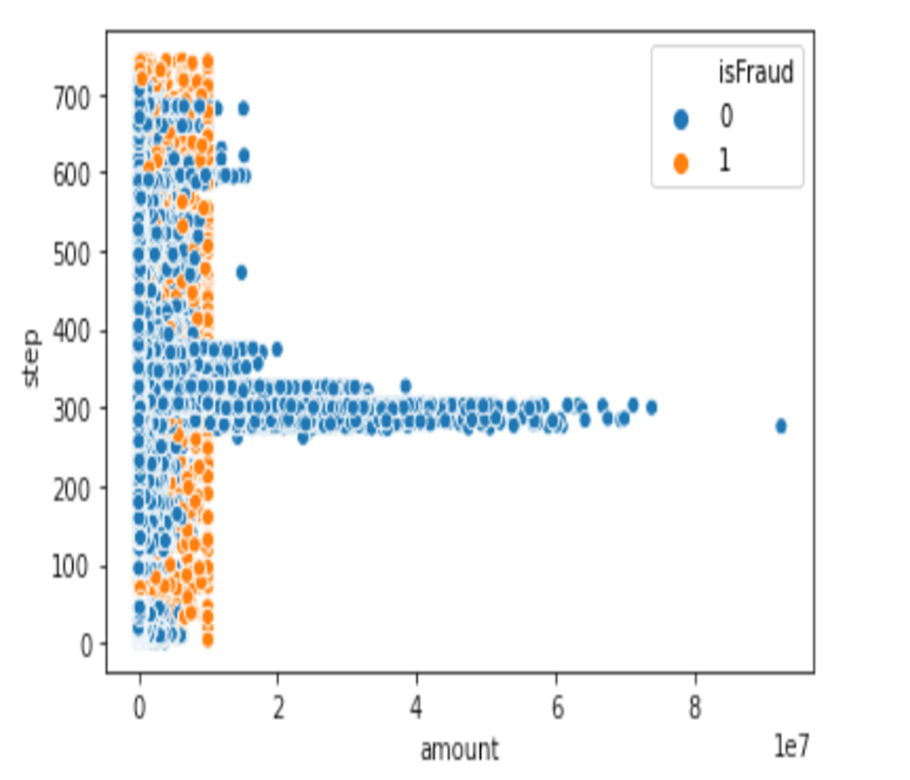
1. Among all the customer transactions, count of transactions having IsFlaggedFraud to be 1 is 16 which is a very low count to extract any pattern to identify fraud transactions.



1. The target variable IsFraud can be found only in two transaction types Cash\_Out and Transfer.



1. In the graph plotted between step (1 hour time) and amount transferred, all fraud transactions can be found having transferred amount to be less than 1 crore rupees.



1. This is the correlation matrix wherein there is no good correlation score between the target variable and other input variables.

