**Fraud Detection Model**

**Content:**

The datasets contains transactions made by customers of a finance company. This dataset presents transactions that occurred in some days, where we have 8213 frauds out of 63,62,619 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.129% of all transactions.

It contains not only numerical input variables but also sum categotical variables as well. The dataset has columns as Step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, isFraud, isFlaggedFraud which indicated to perticular features and all of them are also mentioned below.

**Step**: maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation).

**Type:** CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER

**Amount**: amount of the transaction in local currency.

**nameOrig:** customer who started the transaction

**oldbalanceOrg**: initial balance before the transaction

**newbalanceOrig:** new balance after the transaction

**nameDest:** - customer who is the recipient of the transaction

**oldbalanceDest:** - initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).

**newbalanceDest** - new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).

**isFraud** - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.

**isFlaggedFraud** - The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

**Exploratory Data Analysis:**

Talking about any null or missing values in the dataset, I would say that there were no missing or NaN values in the raw dataset. Also, from the descriptive statistics of our dataset we could clearly see that our data is wide spread, i.e is having high variance. At this point, I could guess that the dataset might be unbalanced and would require undersampling in order to train our machine learning model well.

Moving forward with the Exploratory data analysis, we encountered the unbalanced characteristic of our dataset, we also analysed the presence of fraudulent transactions deeply for different types of transactions in order to get more clarified and proper view of how data is present. After this, I filtered the dataset by dropping the column which were not contributing significantly in providing any fruitful information for our model, such as “**nameOrig**” and “**nameDest**”.

Moving forward, I analysed the pattern of transactions for both fraudulent transactions and genuine/valid transactions. I did this by checking the balance in the account of both the sender and the receiver before and after the transaction on a condition that the amount given is less than or equal to the amount that is in the sender's account.

and, the amount received is less than or equal to the amount that is in the receiver's account. I found that,

Number of observations where the amount given is greater than the amount that is in the sender's account: 4079080

Number of observations where the amount given is greater than the amount that is in the sender's account: 4079080

After getting these results, for double checking of fraudulent transactions or can say getting one more feature to analyse the transaction being valid or fraud, I checked for error balance in the accounts, and found that

Percentage of observations with balance errors in the account giving money: 85.0

Percentage of observations with balance errors in the account receiving money: 74.0

And thus, from such outcomes I was able to conclude that

1. There is erroneous results in the new and old balance accounts for both sender and receiver
2. Some of this erroneous results are due to fraudulent transactions
3. We cannot get rid of this features as well so we will let them be and add a new feature called 'errorbalance’

After adding these feature I went to check the time of a day when fraudulent transaction happen. By visualizing the data against time, I found that **from hour 0 to hour 9, valid/genuine transactions very rarely occur. On the other hand, fraudulent transactions still occur at similar rates to any hour of the day outside of hours 0 to 9.**

And that is why I added the hours of a day feature into our data set as a column.

**Descriptive Statistics of Amount, both Valid and Fraud**

It seems that during fraudulent transactions, the amount moved is capped at 10 million currency units.

Whereas for valid transactions, the amount moved is capped at about 92.4 million currency units.

Only valid transaction involved amounts larger than 10,000,000, however these transactions make up less than 0.01% of the relevant data.

When the amounts moved is less than 10,000,000 there doesn't seem to be a large difference fraudulent and valid transactions.

I will leave the variable amount as it is and will not be creating a feature out of it.

## **Handling Categorical Data**

So, we have a column "type" which consists of categorical features as "TRANSFER" and "CASH\_OUT". Now, as we cannot assign ranking to these features, as neither of them if higher or lower from another one.

Therefore, I will be applying **One-Hot Encoding** for these variables.

One-Hot encoding involves creating indicator variables for each category in a categorical variable. If an observation is part of a particular category (e.g. the transaction type is CASH\_OUT), the indicator variable associated with the category would be 1. If it isn't part of a particular category, then the indicator variable associated with that category would be 0.

## **Scaling and Splitting the Data into Test and Train**

Data scaling is a recommended pre-processing step when working with many machine learning algorithms. Data scaling can be achieved by normalizing or standardizing real-valued input and output variables. I scaled down the data using StandardScalar() function in order to make data ready and fit for training the machine learning model.

But, Why did I scaled my data. This is because Machine learning models learn a mapping from input variables to an output variable. As such, the scale and distribution of the data drawn from the domain may be different for each variable. Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. This can be thought of as subtracting the mean value or centering the data.

After scaling my dataset, I split it into the test and train data sets.

## **Handling Imbalanced Data**

A widely adopted technique for dealing with highly unbalanced datasets is called resampling. It consists of removing samples from the majority class (under-sampling) and/or adding more examples from the minority class (over-sampling). The simplest implementation of **over-sampling** is to duplicate random records from the minority class, which can cause overfishing. In **under-sampling**, the simplest technique involves removing random records from the majority class, which can cause loss of information.

Chart, diagram

Description automatically generated