Fraud Detection

1)Loading dataset:

In this step we load our dataset and try to get familiar with its unique characteristics. In this task we have two datasets one for training and one for testing.

I used a for loop to get some characteristics of all of the columns that are the most important ones : number of unique values , number of null values , percentage of null values and the type of the column.

We can also clearly see that our target column is in the form of binary, this will later help us in the model selection phase to choose the appropriate model for training our data.

2) Sampling:

Since the test dataset is going to be used only for evaluating the model the whole test dataset isn’t needed, because of its size. So we can get a sample from our test dataset.

First we need to know how many fraud instances and non-fraud ones exist in this dataset. The results show that the majority of the dataset rows are filled with information of non-fraud instances.

Here we can take a sample from non-fraud part of the dataset in a way that it’d be 10 times larger than the fraud part of the dataset (ration 1 / 10).

We also define the training and testing sets for model training.

3)Processing:

There are few transformations that we apply on our dataset in this phase:

1. Haversine distance:

The purpose of the **haversine package** is to provide functions for calculating distances between two points on the Earth’s surface using **latitude and longitude coordinates**. Specifically, it uses the **haversine formula**.

In our dataset we have both of these information so we calculate this distance to somehow create a new feature based on it.

1. Age:

By defining and using this class we’re trying to find the age feature based on the date of birth and transaction time stamp, which represents the date of a transaction.

1. Data Features:

The purpose of this class is to extract some information and use them as features from trans\_date\_trans\_time

This features includes day of week, day, month and year. We extract these and add them as new columns to our dataset.

1. Memory Efficient Transformer:

The purpose of this class is to create a memory-efficient transformer for numerical data in a pandas DataFrame.

It achieves this by downcasting the data types of numeric columns (both float and integer) to use less memory while preserving the data’s integrity.

1. Drop columns :

In this step we drop all of the columns we used in the previous steps that we don’t need them anymore.

* We do all of these transformation on the x\_train and also x\_test , because it’s necessary to have the right version of both for model training phase.

4) Scaling:

Scaling numerical features is always as important step while getting data ready for training a model on it.

Basically, scaling makes sure that all numerical features will get a fair treatment in the training process and it makes dataset ready for model training.

Here, I used standard scaler which is a common way to get through the scaling stage.

5)Encoding:

In this step I used one-hot encoder to encode categorical features in the both training set and testing set.

6) Model and ROC curve:

This is the final step, we train all of model listed on our data then apply all of the tests on our model using testing set.

Here is the ROC curve:

