MODEL: **Fraud detection system** -

Problem statement: Used to check whether a credit card transaction is legit (Correct / Legal / Fraud free) or not.

ML Algorithm Used: Logistic Regression Model

**WORK FLOW:**

1. Data collection part:

* Getting the dataset from Kaggle.
* Information about the dataset.
* ABOUT THE DATASET:

The dataset contains transactions made by credit cards.

Class column tells whether a transaction is legal (0) or fraud (1).

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. Only 0.172% is fraud 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

dependent cost-sensitive learning.

Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0

otherwise.

The dataset does not contain any missing value for each column.

1. Data preprocessing:

* The dataset that we use here is an imbalance dataset i.e., here we do not have equal number of fraud and legal transactions details. In the dataset, the number of legal transactions (284,807) is greater as compared to fraud (492). Hence to balance them is a challenge.
* Imbalanced dataset cannot be fed into the model to predict the value because it would give the result on the basis of majority items present. Here in this case, most of the data points in the trained model are for legal transaction hence answer every time would be legal (no fraud transactions could be identified). Hence the dataset needs to be preprocessed.
* Sampling Technique (Simple Random Sampling) is used to get the uniform dataset which can be fed into the model.

1. Data Analysis: Analysis of various features in the dataset.

* Labels are:

Legit/legal transaction - 0

Fraud transaction - 1

1. Train Test Split:

* Splitting the dataset into training data and testing data.
* Find the accuracy score of both training data and testing data is to be calculated.
* Then checking that the model is not overfitted nor underfitted.

1. Logistic Regression Model:

* Logistic Regression model is used because it is a binary classification problem.
* Binary classification refers to those classification tasks that have two class labels.

1. Evaluation:

* With the help of test data, predict the accuracy of the model.
* Checking the accuracy score for overfitting and underfitting.

**WHOLE PROCESS:**

* All the python coding part is done on GOOGLE COLAB.
* **IMPORTING LIBRARIES:**

**To split the data into training data and testing data:**

from sklearn.model\_selection import train\_test\_split

**To check the performance of our model:**

from sklearn.metrics import accuracy\_score

**To import the model that we will be using:**

from sklearn.linear\_model import LogisticRegression

**For the preprocessing of the dataset:**

from sklearn.preprocessing import StandardScaler

**Loading the dataset (in the form of csv file) to the pandas data frame**

card\_data=pd.read\_csv('/content/creditcard.csv')

**To get information about the dataset**

card\_data.info ()

**Checking the number of missing values in each column:**

card\_data.isnull().sum()

Labels are:

1. Legit/legal transaction, 1- Fraud transaction

**To check the distribution of legal and fraud transactions in the dataset:** ‘Class’ column

card\_data[‘Class’].value\_counts()

**Separate the two types of transactions:** (can be done with the help of class values)

authorized=card\_data[card\_data.Class ==0]

fraud=card\_data[card\_data.Class ==1]

**Printing the details of the two groups:**

print(authorized.shape)

print(fraud.shape)

**Statistical measures of the data:**

authorized.Amount.describe()

**Comparing the values for transactions:** Grouping the two classes on the basis of mean

card\_data.groupby(‘Class’).mean()

It will calculate mean value for all the transaction level values. There is a significant difference between the means of both the transactions. The difference is useful to determine whether the transaction is fraudulent or legit.

**Under-Sampling:**

Building a sample dataset containing similar distribution of normal transaction and fraudulent transaction.

Here we are going to take randomly 492 samples from the legit transactions and join them with the 492 fraudulent transactions, so as to balance the dataset.

authorized\_sample= authorized.sample(n=492) #It will take random samples (random samping)

If we got a bad sample then the mean values of the sample would be largely different.

**Concatenating the two data frames:**

new\_dataset = pd.concat([authorized\_sample , fraud], axis=0)

axis =0 signifies that the values should be added below (row wise)

axis =1 signifies that the values should be added vertically (column wise)

**Splitting the data into Features & Targets**

X=new\_dataset.drop(column=’Class’,axis=1) #Targets

Y=new\_dataset[‘Class’] #Corresponding answers

axis=1 means we need to remove a vertical column

**Splitting the dataset into training and testing data:**

X\_train , X\_test , Y\_train , Y\_test = train\_test\_split(X , Y , test\_size=0.2 , stratify=Y,random\_state=2)

test\_size=0.2 the test data will be 20% of the whole dataset.

stratify=Y so that there is even distribution of the both the classes (Y) in the train and test dataset

random\_state=2 for the random splitting of the dataset.

* **Training the model:**

**Loading our model:**

model=LogisticRegression()

**Fitting / Inputting the dataset (training dataset) into the model, to get it trained:**

model.fit(X\_train,Y\_train)

X\_train: the training data values

Y\_train: the corresponding targets

* **MODEL EVALUATION:**

**First prediction the value for the given training inputs:**

X\_train\_prediction=model.predict(X\_train)

**Calculating the accuracy score of the predicted value by the model with the orginal target value provided.**

training\_data\_accuracy=accuracy\_score(X\_train\_prediction, Y\_train)

X\_train\_prediction: It is the value predicted by the model for the given training data values. (Answer by the model)

Y\_train: It is the target value that should come for the given training dataset (Original answer)

Good Accuracy score lies between 75 to 80 % or more.

**First prediction the value for the given testing inputs:**

X\_test\_prediction = model.predict(X\_test)

**Calculating the accuracy score of the predicted value by the model with the original target value provided:**

testing\_data\_accuracy=accuracy\_score(X\_test\_prediction,Y\_test)

X\_test\_prediction: It is the value predicted by the model for the given testing data values. (Answer by the model)

Y\_test: It is the target value that should come for the given testing dataset (Original answer)

The accuracy score for the training and testing dataset should be closer to each other for a good trained model.

**Training data accuracy:** 0.9224904701397713 equivalent to 92.24% (Variable)

**Testing data accuracy:** 0.9137055837563451 equivalent to 91.37% (Variable)

***Project link*:** [https://github.com/Akriti2510/American-Express-Makeathon/blob/main/ML-based%20fraud%20Detection%20.ipynb](%20https:/github.com/Akriti2510/American-Express-Makeathon/blob/main/ML-based%20fraud%20Detection%20.ipynb)