 

## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CREDIT CARD FRAUD DETECTION USING PYTHON

Submitted by

## SRIHARI M 927621BEC211

## SHANJAY R K 927621BEC195

**Elite Training Project Report**

Submission Date: 0**8/08/2023**

**Signature of Staff Incharge Signature of HOD**

**Signature of Elite Training Coordinator**

**Dr.D.Pradeep, AP/CSE**

**Summary**

The Python-based fake credit card detection project is like a digital

guard that watches out for fake or phony credit cards. It checks important things like the special code on the back, when the card expires, the name on the card, and if the address matches. It uses smart technology to notice strange behavior or unusual locations. This guard also uses a special chip and fingerprints from devices to be extra sure. From setting things up, writing code, testing thoroughly, and explaining everything clearly, this project creates a strong tool to stop bad guys from making fake transactions with credit cards.

**Screen Shots**

**Source code**

Importing the Dependencies

[ ]

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

[ ]

# loading the dataset to a Pandas DataFrame

credit\_card\_data = pd.read\_csv('/content/credit\_data.csv')

[ ]

# first 5 rows of the dataset

credit\_card\_data.head()

[ ]

credit\_card\_data.tail()

[ ]

# dataset informations

credit\_card\_data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 284807 entries, 0 to 284806

Data columns (total 31 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Time 284807 non-null float64

1 V1 284807 non-null float64

2 V2 284807 non-null float64

3 V3 284807 non-null float64

4 V4 284807 non-null float64

5 V5 284807 non-null float64

6 V6 284807 non-null float64

7 V7 284807 non-null float64

8 V8 284807 non-null float64

9 V9 284807 non-null float64

10 V10 284807 non-null float64

11 V11 284807 non-null float64

12 V12 284807 non-null float64

13 V13 284807 non-null float64

14 V14 284807 non-null float64

15 V15 284807 non-null float64

16 V16 284807 non-null float64

17 V17 284807 non-null float64

18 V18 284807 non-null float64

19 V19 284807 non-null float64

20 V20 284807 non-null float64

21 V21 284807 non-null float64

22 V22 284807 non-null float64

23 V23 284807 non-null float64

24 V24 284807 non-null float64

25 V25 284807 non-null float64

26 V26 284807 non-null float64

27 V27 284807 non-null float64

28 V28 284807 non-null float64

29 Amount 284807 non-null float64

30 Class 284807 non-null int64

dtypes: float64(30), int64(1)

memory usage: 67.4 MB

[ ]

# checking the number of missing values in each column

credit\_card\_data.isnull().sum()

Time 0

V1 0

V2 0

V3 0

V4 0

V5 0

V6 0

V7 0

V8 0

V9 0

V10 0

V11 0

V12 0

V13 0

V14 0

V15 0

V16 0

V17 0

V18 0

V19 0

V20 0

V21 0

V22 0

V23 0

V24 0

V25 0

V26 0

V27 0

V28 0

Amount 0

Class 0

dtype: int64

[ ]

# distribution of legit transactions & fraudulent transactions

credit\_card\_data['Class'].value\_counts()

0 284315

1 492

Name: Class, dtype: int64

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> fraudulent transaction

[ ]

# separating the data for analysis

legit = credit\_card\_data[credit\_card\_data.Class == 0]

fraud = credit\_card\_data[credit\_card\_data.Class == 1]

[ ]

print(legit.shape)

print(fraud.shape)

(284315, 31)

(492, 31)

[ ]

# statistical measures of the data

legit.Amount.describe()

count 284315.000000

mean 88.291022

std 250.105092

min 0.000000

25% 5.650000

50% 22.000000

75% 77.050000

max 25691.160000

Name: Amount, dtype: float64

[ ]

fraud.Amount.describe()

count 492.000000

mean 122.211321

std 256.683288

min 0.000000

25% 1.000000

50% 9.250000

75% 105.890000

max 2125.870000

Name: Amount, dtype: float64

[ ]

# compare the values for both transactions

credit\_card\_data.groupby('Class').mean()

Under-Sampling

Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions

Number of Fraudulent Transactions --> 492

[ ]

legit\_sample = legit.sample(n=492)

Concatenating two DataFrames

[ ]

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

[ ]

new\_dataset.head()

[ ]

new\_dataset.tail()

[ ]

new\_dataset['Class'].value\_counts()

1 492

0 492

Name: Class, dtype: int64

[ ]

new\_dataset.groupby('Class').mean()

Splitting the data into Features & Targets

[ ]

X = new\_dataset.drop(columns='Class', axis=1)

Y = new\_dataset['Class']

[ ]

print(X)

Time V1 V2 ... V27 V28 Amount

203131 134666.0 -1.220220 -1.729458 ... 0.173995 -0.023852 155.00

95383 65279.0 -1.295124 0.157326 ... 0.317321 0.105345 70.00

99706 67246.0 -1.481168 1.226490 ... -0.546577 0.076538 40.14

153895 100541.0 -0.181013 1.395877 ... -0.229857 -0.329608 137.04

249976 154664.0 0.475977 -0.573662 ... 0.058961 0.012816 19.60

... ... ... ... ... ... ... ...

279863 169142.0 -1.927883 1.125653 ... 0.292680 0.147968 390.00

280143 169347.0 1.378559 1.289381 ... 0.389152 0.186637 0.76

280149 169351.0 -0.676143 1.126366 ... 0.385107 0.194361 77.89

281144 169966.0 -3.113832 0.585864 ... 0.884876 -0.253700 245.00

281674 170348.0 1.991976 0.158476 ... 0.002988 -0.015309 42.53

[984 rows x 30 columns]

[ ]

print(Y)

203131 0

95383 0

99706 0

153895 0

249976 0

..

279863 1

280143 1

280149 1

281144 1

281674 1

Name: Class, Length: 984, dtype: int64

Split the data into Training data & Testing Data

[ ]

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

[ ]

print(X.shape, X\_train.shape, X\_test.shape)

(984, 30) (787, 30) (197, 30)

Model Training

Logistic Regression

[ ]

model = LogisticRegression()

[ ]

# training the Logistic Regression Model with Training Data

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

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, l1\_ratio=None, max\_iter=100,

multi\_class='auto', n\_jobs=None, penalty='l2',

random\_state=None, solver='lbfgs', tol=0.0001, verbose=0,

warm\_start=False)

Model Evaluation

Accuracy Score

[ ]

# accuracy on training data

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

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

[ ]

print('Accuracy on Training data : ', training\_data\_accuracy)

Accuracy on Training data : 0.9415501905972046

[ ]

# accuracy on test data

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

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

[ ]

print('Accuracy score on Test Data : ', test\_data\_accuracy)

Accuracy score on Test Data : 0.9390862944162437