#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 A DATASET

credit\_card\_data.head()

credit\_card\_data.tail()

# DATASET INFORMATIONS

credit\_card\_data.info()

# CHECKING THE NUMBER OF MISSING VALUES IN EACH COLUMN

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

# DISTRIBUTION OF LEGIT TRANSACTIONS & FRAUDULENT TRANSACTIONS

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

# SEPERATING 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)

# STATISTICAL MEASURE OF THE DATA

legit.Amount.describe()

fraud.Amount.describe()

# 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 DATA FRAMES

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

new\_dataset.head()

new\_dataset.tail()

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

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)

print(Y)

#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)

#MODEL TRAINING

LOGISTIC REGRESSION

model = LogisticRegression()

# TRAINING THE LOGISTIC REGRESSION MODEL WITH WITH TRAINING DATA

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

#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 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)