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()

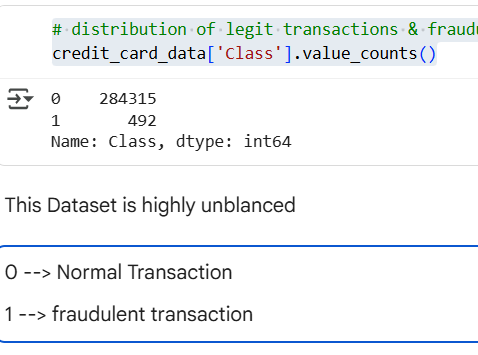
credit\_card\_data.tail()

# 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()



# separating the data for analysis

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

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

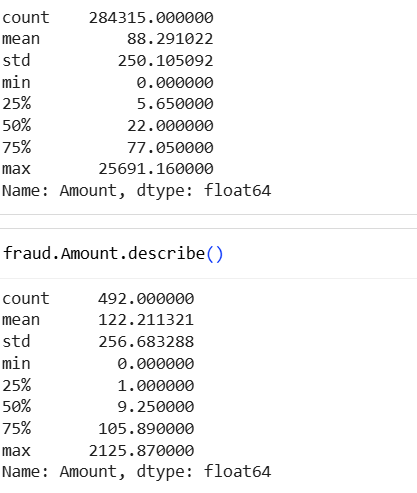
print(legit.shape)

print(fraud.shape)



# statistical measures of the data

legit.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

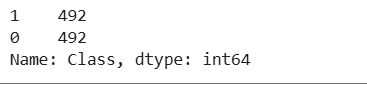
Number of Fraudulent Transactions --> 492

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

new\_dataset.head()

new\_dataset.tail()

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

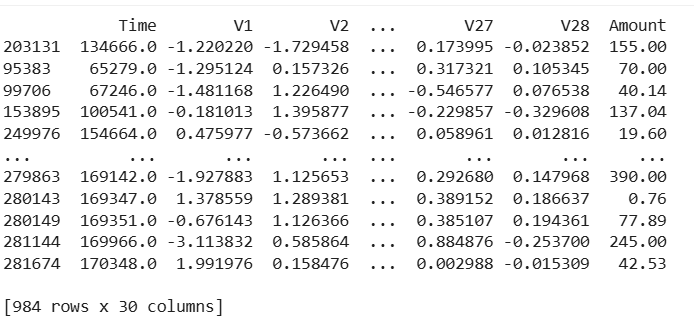


new\_dataset.groupby('Class').mean()

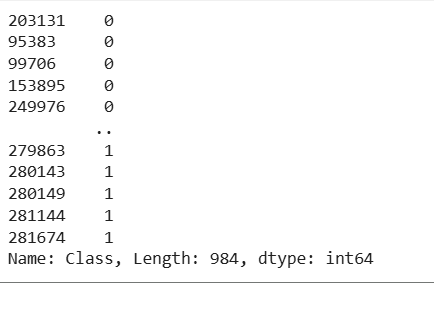
X = new\_dataset.drop(columns='Class', axis=1)

Y = new\_dataset['Class']

print(X)



print(Y)



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 = 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 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