## 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

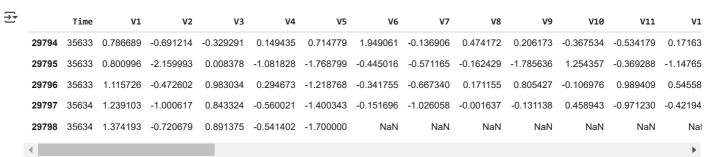
from sklearn.metrics import accuracy\_score

#Loading dataset to Pandas Dataframe
credit\_card\_data = pd.read\_csv('/content/creditcard.csv')

#first 5 rows of the dataset
credit\_card\_data.head()

<del></del>	T	ime	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	
	0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.
	1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.
	2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.
	3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.
	4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.
	4 ■														•

credit\_card\_data.tail()



#number of rows and columns
credit card data.shape

**→** (29799, 31)

24 V24

25 V25

26 V26

#dataset information
credit\_card\_data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 29799 entries, 0 to 29798 Data columns (total 31 columns): # Column Non-Null Count Dtype ---0 Time 29799 non-null int64 29799 non-null float64 2 V2 29799 non-null float64 3 V3 29799 non-null float64 4 ٧4 29799 non-null float64 5 V5 29799 non-null float64 6 V6 29799 non-null float64 7 V7 29799 non-null float64 8 V8 29799 non-null float64 9 V9 29799 non-null float64 10 V10 29799 non-null float64 11 V11 29799 non-null float64 12 V12 29799 non-null float64 13 V13 29799 non-null float64 14 V14 29799 non-null float64 29799 non-null 15 V15 float64 29799 non-null float64 16 V16 29799 non-null float64 17 V17 18 V18 29799 non-null float64 19 V19 29799 non-null float64 20 V20 29799 non-null float64 21 V21 29799 non-null float64 22 V22 29799 non-null float64 23 V23 29799 non-null float64

29799 non-null

29799 non-null

29799 non-null

float64

float64

float64

```
27 V27 29799 non-null float64
28 V28 29799 non-null float64
29 Amount 29799 non-null float64
30 Class 29799 non-null float64
dtypes: float64(30), int64(1)
memory usage: 7.0 MB
```

#check missing values in dataset
credit\_card\_data.isnull().sum()

```
v
  Time
          0
   V1
          0
   V2
          0
   V3
          0
   V4
          0
   V5
          0
   V6
          1
   V7
   V8
          1
   V9
  V10
          1
  V11
          1
  V12
  V13
  V14
  V15
          1
  V16
          1
  V17
  V18
  V19
  V20
  V21
          1
  V22
  V23
  V24
  V25
  V26
  V27
  V28
 Amount 1
  Class
dtype: int64
```

credit\_card\_data\_cleaned = credit\_card\_data.dropna()

print (credit\_card\_data\_cleaned)

```
\overline{\mathbf{T}}
                         V1
                                    V2
                                              V3
                                                         V4 ...
                                                                        V26
                                                                                  V27
                                                                                             V28 Amount Class
             Time
                0 \ -1.359807 \ -0.072781 \ \ 2.536347 \ \ 1.378155 \ \ \dots \ -0.189115 \ \ 0.133558 \ -0.021053
                                                                                                  149.62
                                                                                                             0.0
    1
                0 \quad 1.191857 \quad 0.266151 \quad 0.166480 \quad 0.448154 \quad \dots \quad 0.125895 \quad -0.008983 \quad 0.014724
                                                                                                    2.69
                                                                                                             0.0
                                                             ... -0.139097 -0.055353 -0.059752
    2
                1 -1.358354 -1.340163 1.773209 0.379780
                                                                                                  378.66
                                                                                                             0.0
                                                             ... -0.221929
                1 -0.966272 -0.185226 1.792993 -0.863291
                                                                            0.062723
                                                                                        0.061458
                                                                                                  123.50
                                                              ... 0.502292
    4
                2 -1.158233 0.877737
                                       1.548718
                                                   0.403034
                                                                             0.219422
                                                                                        0.215153
                                                                                                   69.99
                                                                                                             0.0
                                                              . . .
    35736
           38239 -3.120568 -0.121316
                                       1.307875 1.582101
                                                             ... -0.070668 -0.870421 0.624730
                                                                                                    2.28
                                                                                                             0.0
    35737
            38240 1.232390 0.122010
                                        0.157352
                                                   0.261906
                                                             ... 0.158446 -0.018118 -0.003168
                                                                                                    1.79
                                                                                                             0.0
    35738
            38240 1.114040 0.571203
                                        0.427035
                                                   2.442135
                                                              ... -0.025773 -0.035775
                                                                                       0.011219
                                                                                                    24.99
                                                                                                             0.0
                                                             ... -0.314989 0.027467 0.010613
            38241 1.057020 0.007895 0.239256 1.236048
    35739
                                                                                                    53.96
                                                                                                             0.0
           38241 -1.546226  0.693338  1.002815 -1.528992  ...  0.739989  0.043625 -0.140629
    35740
                                                                                                    0.76
```

```
[35741 rows x 31 columns]
```

#distribution of legit transaction and fradulent transaction
credit\_card\_data['Class'].value\_counts()



count

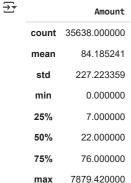
```
Class0.0 356381.0 103
```

This Dataset is highly unbalanced

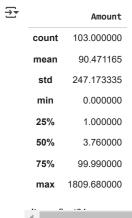
0 = Normal Transaction

1 = Fradulent Transaction

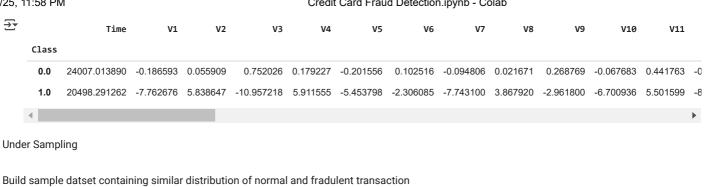
#statistical measures of the data
legit.Amount.describe()



fraud.Amount.describe()



#compare the values for both transactions
credit\_card\_data.groupby('Class').mean()



legit\_sample = legit.sample(n=103)

Concatenation two dataframes

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

new\_dataset.head()

<b>→</b>		Time	V1	V2	V3	V4	V5	V6	V7	V8	<b>V</b> 9	V10	V11	V12
	19656	30442	0.960089	-0.833866	0.849283	0.633604	-1.385595	-0.318221	-0.578834	0.027567	-1.033440	0.946116	1.172593	0.510031
	33533	37268	1.149505	0.396102	0.615531	2.310495	-0.180587	-0.071877	-0.104923	0.173158	-0.685428	0.893060	0.597537	-0.479812
	5015	4596	1.116750	-0.470737	0.228822	-1.806058	-0.133105	0.472825	-0.437491	0.206482	3.326446	-2.211929	1.892923	-0.755930
	7505	10247	1.062773	-0.093531	1.485816	1.439849	-0.880143	0.484425	-0.923960	0.357869	2.100503	-0.406822	2.034377	-1.887537
	7480	10180	-0.467669	1.075407	1.725491	0.062699	0.068057	-0.543345	0.483334	0.011039	0.655667	-0.506955	2.218326	-1.950564
	4													•

new\_dataset.tail()

<del>_</del>		Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
	30442	35926	-3.896583	4.518355	-4.454027	5.547453	-4.121459	-1.163407	-6.805053	2.928356	-4.917130	-6.600461	3.367846	-7.888978
	30473	35942	-4.194074	4.382897	-5.118363	4.455230	-4.812621	-1.224645	-7.281328	3.332250	-3.679659	-7.524368	2.954344	-7.099825
	30496	35953	-4.844372	5.649439	-6.730396	5.252842	-4.409566	-1.740767	-6.311699	3.449167	-5.416284	-7.833556	3.657350	-7.781448
	31002	36170	-5.685013	5.776516	-7.064977	5.902715	-4.715564	-1.755633	-6.958679	3.877795	-5.541529	-7.502112	3.676703	-7.642983
	33276	37167	-7.923891	-5.198360	-3.000024	4.420666	2.272194	-3.394483	-5.283435	0.131619	0.658176	-0.794994	3.266066	-2.719185
	4													<b>&gt;</b>

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

<del>\_</del> count

> Class 0.0 103

1.0 103

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

**→** Time ٧1 V2 V10 V11 ٧3 ۷4 ۷5 ۷6 ۷7 V8 V9 Class 21641.097087 0.0 0.117184 0.202749 0.905756 0.477824 0.36955 -0.163345 0.479711 -0.7 20498.291262 -7.762676 5.838647 -10.957218 5.911555 -5.453798 -2.306085 -7.74310 3.867920 -2.96180 -6.700936 5.501599 -8.40 1.0

Spliting the data in features and targets

```
X = new_dataset.drop(columns='Class',axis=1)
```

Y = new\_dataset['Class']

```
2/19/25, 11:58 PM
                                                                  Credit Card Fraud Detection.ipynb - Colab
    print(X)
    ₹
                              V1
                                        V2
                                                   V3 ...
                                                                 V26
                                                                            V27
                                                                                      V28 Amount
                 Time
         19656 \quad 30442 \quad 0.960089 \quad -0.833866 \quad 0.849283 \quad \dots \quad -0.344320 \quad 0.036178 \quad 0.055045 \quad 158.00
                                                       ... -0.257171 -0.024791
         33533 37268 1.149505 0.396102 0.615531
                                                                                 0.012328
                                                                                            10.51
         5015
                 4596
                       1.116750 -0.470737
                                            0.228822
                                                       ... -0.768814 0.117068
                                                                                 0.001664
                                                                                             2.18
                                                      ... -0.442725 0.050931
         7505
                10247 1.062773 -0.093531 1.485816
                                                                                 0.016670
                                                                                             4.99
         7480
                10180 -0.467669 1.075407
                                            1.725491
                                                       ... 0.015819
                                                                      0.232992
                                                                                 0.104472
                                                                                             2.67
                                                       . . .
         30442
                35926 -3.896583 4.518355 -4.454027
                                                       ... 0.412191
                                                                      0.635789
                                                                                 0.501050
                                                                                             4.56
         30473
                35942 -4.194074 4.382897 -5.118363
                                                       ... -0.131687
                                                                      0.473934
                                                                                 0.473757
                                                                                            14.46
                                                       ... -0.270328
         30496
                35953 -4.844372 5.649439 -6.730396
                                                                      0.210214
                                                                                 0.391855
                                                                                           111.70
                36170 -5.685013 5.776516 -7.064977
                                                       ... -0.049447
         31002
                                                                      0.303445 0.219380
                                                                                           111.70
                                                       ... 0.520508 1.937421 -1.552593
         33276 37167 -7.923891 -5.198360 -3.000024
                                                                                            12.31
         [206 rows x 30 columns]
    print(Y)
        19656
    →
         33533
                  0.0
         5015
                  0.0
         7505
                  0.0
         7480
                  0.0
         30442
                  1.0
         30473
                  1.0
         30496
                  1.0
         31002
                  1.0
         33276
         Name: Class, Length: 206, dtype: float64
    split data in training and test data
    X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.3, stratify=Y, random\_state=3)
    print(X.shape,X_train.shape,X_test.shape)
    → (206, 30) (144, 30) (62, 30)
    Model Training
    Logistic Regression
    model = LogisticRegression()
    #training logistic regression model with training data
    model.fit(X_train,Y_train)
    /usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status-
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
         Increase the number of iterations (\max_{}iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             \underline{\texttt{https://scikit-learn.org/stable/modules/linear\_model.html\#logistic-regression}}
           n_iter_i = _check_optimize_result(
          ▼ LogisticRegression ① ?
         LogisticRegression()
    Model Evaluation
    Accuracy Score
```

```
Generated code may be subject to a license | AnmolGulati6/Rock-vs-Mines-Classification-using-Logistic-Regression
#accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,Y_train)
\label{lem:generated} \textit{Generated code may be subject to a license} \ | \ Al-ML-ZETECH-UNIVERSITY/ROCK-VS-MINE-PREDICTION \\ \textit{print('Accuracy on training data : ',training_data_accuracy)} 
Accuracy on training data : 0.97222222222222
#accuracy on test data
```

```
x_test_prediction = moder.predict(A_test)
test_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

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

→ Accuracy on test data : 0.9838709677419355