Data Engineer INTERN at HACKVEDA LIMITED

AUTHOR : BANDANA PRAKASH

TASK 5 : CREDIT CARD FRAUD DETECTION

PURPOSE : Build a machine learning model to identify fraudulent credit card transactions.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
file=pd.read_csv("creditcard.csv")
```

```
file.head(10)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.1104
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.1012
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.9094
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.1903
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.1374
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.0263
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.1541
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.0575
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.2042
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.1207

10 rows × 31 columns

```
file.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+

8 rows × 31 columns

```
file.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
```

```
file['Class'].value_counts()
```

```
Class
0    284315
1       492
Name: count, dtype: int64
```

```
normal=file[file.Class==0]
```

```
fraud=file[file.Class==1]
```

```
print(normal.shape)
```

```
(284315, 31)
```

```
print(fraud.shape)
```

```
(492, 31)
```

```
normal.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
```

```
file.groupby('Class').mean()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	
Class													
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	-0.000644	-0.00
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.71

2 rows × 30 columns

```
normal_sample=normal.sample(n=492)
```

```
new_file=pd.concat([normal_sample,fraud],axis=0)
```

```
new_file.head(10)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
32993	37045.0	1.273090	-0.744403	1.083617	-0.682701	-1.537089	-0.502750	-1.064374	0.039786	-0.738666	...	0.465367	1.22964
176252	122689.0	-0.619340	0.650909	0.853761	-0.441992	1.189456	0.079074	1.075137	-0.224906	-0.173498	...	-0.337960	-0.88283
71866	54473.0	-1.459553	0.016956	1.063610	-1.484100	-0.244744	-1.080333	-0.089253	0.466594	1.243897	...	0.144194	0.45999
189917	128610.0	1.778519	-0.112447	-1.234223	0.996746	0.961386	1.807237	-0.624287	0.614767	0.549400	...	-0.183639	-0.42127
153148	98024.0	-0.557542	1.064676	0.524862	-1.771705	1.141241	-0.310842	0.624603	0.006453	1.478922	...	-0.514084	-1.28654
91669	63576.0	0.910530	-1.359016	0.862437	-0.590947	-1.567072	0.005582	-0.959290	0.164846	-0.678745	...	0.586836	1.21331
248153	153810.0	2.102483	-1.302057	0.379806	-0.486967	-1.830271	-0.165263	-1.633904	0.141010	1.060734	...	0.134621	0.69733
175828	122505.0	-2.783805	-2.928222	-1.500618	-1.979360	1.645353	0.802380	1.036764	-0.020237	-1.068077	...	0.242157	1.41124
201518	133915.0	2.006453	-1.760294	-0.688837	-1.337221	-1.645638	-0.848000	-0.970753	-0.328486	-1.233229	...	-0.044086	0.14975
57869	48115.0	1.314915	-0.980378	-0.032665	-2.770975	-1.047365	-0.705180	-0.491240	-0.076395	0.571959	...	-0.416346	-0.48188

10 rows × 31 columns

```
new_file['Class'].value_counts()
```

```
Class
0    492
1    492
Name: count, dtype: int64
```

```
new_file.groupby('Class').mean()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	
Class													
0	96327.323171	0.077345	-0.022423	0.081215	-0.129899	0.029804	-0.009543	0.086710	-0.040480	-0.027447	...	-0.071245	0.017
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713

2 rows × 30 columns

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

```
Y=new_file['Class']
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

```
model=LogisticRegression()
```

```
model.fit(X_train,Y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
X_train_prediction=model.predict(X_train)
```

```
training_data_acuracy=accuracy_score(X_train_prediction,Y_train)*100
```

```
print(f"Training Data Accuracy: {training_data_acuracy}%")
```

```
Training Data Accuracy: 93.90088945362135%
```

```
X_test_prediction=model.predict(X_test)
```

```
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)*100
```

```
print(f"Test Data Accuracy: {test_data_accuracy}%")
```

```
Test Data Accuracy: 91.87817258883248%
```