```
import numpy as np
import pandas as pd
```

```
df=pd.read_csv("/content/drive/MyDrive/datasets/creditcard.csv")
df
```

me	V1	V2	V3	V4	V5	V6	V7	V8	
0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.3
0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2
1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3
2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8
...	...	...	...	...	...	...	...	...	
6.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.9
7.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.5
8.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.4
8.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.3
2.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.4

. columns

```
df.isna().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
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

```
df.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006

5 rows × 31 columns

```
df.size
```

```
8829017
```

```
df.shape
```

```
(284807, 31)
```

```
df.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
```

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

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

```
legit = df[df.Class == 0]
fraud = df[df.Class == 1]
```

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

```
(284315, 31)
(492, 31)
```

Below Amount Is In USD

```
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
```

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

Class	Time	V1	V2	V3	V4	V5	V6
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737

2 rows × 30 columns

Under-Sampling

Build a sample dataset containing similar distribution of normal transaction and fraud transaction

```
legit_sample= legit.sample(n=492)
```

```
df1=pd.concat([legit_sample,fraud], axis=0)
```

```
df1
```

ime	V1	V2	V3	V4	V5	V6	V7	V8	
56.0	-0.413186	0.973255	1.131028	-0.172396	0.371011	-0.002820	0.511332	0.259824	-0.5
76.0	1.480710	-1.244288	0.544534	-1.222229	-1.196488	0.595869	-1.320128	0.078122	-1.1
49.0	1.136690	-0.775679	0.713119	-0.654121	-1.260787	-0.697379	-0.541589	-0.124737	-1.1
71.0	1.410318	-0.358219	-0.328514	-0.909914	-0.221053	-0.423138	-0.237233	-0.095909	-1.3
81.0	-0.384654	1.097950	1.270343	0.010712	0.232524	-0.508251	0.650037	0.053149	-0.4
...	...	...	...	...	...	...	...	...	
42.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.0
47.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.1
51.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.6
66.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.6
48.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.5

```
df1['Class'].value_counts()
```

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

```
df1.groupby('Class').mean()
```

Class	Time	V1	V2	V3	V4	V5	V6	
0	96812.729675	0.098551	-0.038829	0.090610	-0.032804	-0.033002	0.057451	-0.0219
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.5687

2 rows × 30 columns

```
x=df1.iloc[:,:-1]
y=df1.iloc[:,-1]
x
```

	Time	V1	V2	V3	V4	V5	V6	V7
121612	76256.0	-0.413186	0.973255	1.131028	-0.172396	0.371011	-0.002820	0.511332
94196	64776.0	1.480710	-1.244288	0.544534	-1.222229	-1.196488	0.595869	-1.320128
61925	50049.0	1.136690	-0.775679	0.713119	-0.654121	-1.260787	-0.697379	-0.541589
40943	40471.0	1.410318	-0.358219	-0.328514	-0.909914	-0.221053	-0.423138	-0.237233
105354	69481.0	-0.384654	1.097950	1.270343	0.010712	0.232524	-0.508251	0.650037
...	...	...	...	...	...	...	...	...
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850

```
y
```

```
121612    0
94196     0
61925     0
40943     0
105354    0
          ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=.3,stratify=y,random_state=2)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
ypred
```

```
array([1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0])
```

```
df2=pd.DataFrame({'Actual_value':ytest,'Predicted_value':ypred})
df2
```

	Actual_value	Predicted_value
43681	1	1
6331	1	1
151011	1	1
219141	0	0
215984	1	1
...	...	...
82418	0	0
31697	0	0
200163	0	0
8617	1	1
231978	1	0

296 rows x 2 columns

```
from sklearn.metrics import accuracy_score,confusion_matrix
AC=accuracy_score(ypred,ytest)
AC
```

```
0.9358108108108109
```

```
# training Data Accuracy
xtrain_pred = model.predict(xtrain)
train_accuracy=accuracy_score(xtrain_pred,ytrain)
```

```
print("Accuracy On Training Data : ",train_accuracy)
```

```
Accuracy On Training Data :  0.9578488372093024
```

```
# accuracy on test data
xtest_pred= model.predict(xtest)
test_accuracy=accuracy_score(xtest_pred,ytest)
```

```
print("Accuracy On Testing Data : ",test_accuracy)
```

```
Accuracy On Testing Data :  0.9358108108108109
```