```
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
```

```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has executed in the current browser session. Please rerun this cell to enable.

```
Saving creditcard.csv to creditcard.csv
```

```
creditcard = pd.read_csv('creditcard.csv')
```

```
creditcard.head()
```

	Time	V1	V2	V3	V4	V5	V6	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.

```
creditcard.tail()
```

	Time	V1	V2	V3	V4	V5	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.

```
creditcard.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
```

```
creditcard.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
```

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

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

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

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

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

```
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
```

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

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

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

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

```
new_dataset.head()
```

	Time	V1	V2	V3	V4	V5	
231957	146993.0	2.109054	-0.111066	-1.366454	0.260127	0.142871	-0.8
164338	116645.0	-1.640866	-0.596601	1.348165	-0.921482	1.810207	1.5

```
new_dataset.tail()
```

	Time	V1	V2	V3	V4	V5	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.01
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.32
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.00
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.94
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.09

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

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

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

	Time	V1	V2	V3	V4	V5	
Class							
0	95245.638211	-0.028249	0.030169	0.020995	0.025481	0.022514	(
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-

```
x = new_dataset.drop(columns='Class',axis=1)
y= new_dataset['Class']
```

```
print(x)
```

```
               Time         V1         V2  ...        V27        V28  Amo
```

```
    231957  146993.0  2.109054 -0.111066  ... -0.075393 -0.067993    2
    164338  116645.0 -1.640866 -0.596601  ...  0.264933  0.127688   60
    226037  144479.0 -0.565083 -0.284368  ...  0.349133  0.198275   22
    256351  157651.0 -2.301312  1.470114  ...  0.213168 -0.199976   51
    72775    54853.0 -1.863631  0.570899  ...  0.501557  0.155116    9
    ...          ...       ...       ...  ...       ...       ...
    279863  169142.0 -1.927883  1.125653  ...  0.292680  0.147968  390
    280143  169347.0  1.378559  1.289381  ...  0.389152  0.186637    0
    280149  169351.0 -0.676143  1.126366  ...  0.385107  0.194361   77
    281144  169966.0 -3.113832  0.585864  ...  0.884876 -0.253700  245
    281674  170348.0  1.991976  0.158476  ...  0.002988 -0.015309   42

    [984 rows x 30 columns]
```

```
print(y)
```

```
    231957    0
    164338    0
    226037    0
    256351    0
    72775     0
             ..
    279863    1
    280143    1
    280149    1
    281144    1
    281674    1
    Name: Class, Length: 984, dtype: int64
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2
```

```
print(x.shape, x_train.shape, x_train.shape)
```

```
    (984, 30) (787, 30) (787, 30)
```

```
model = LogisticRegression()
```

```
model.fit(x_train, y_train)
```

```
    LogisticRegression(C=1.0, class_weight=None, dual=False, fit_inter
                       intercept_scaling=1, l1_ratio=None, max_iter=10
                       multi_class='auto', n_jobs=None, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001,
                       warm_start=False)
```

```
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 Training data:  0.9339263024142312
```

```
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:  0.8883248730964467
```