

```
In [19]: 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
```

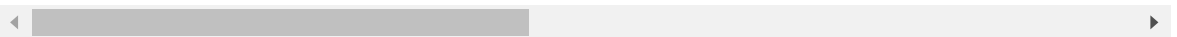
```
In [20]: credit_card_data = pd.read_csv('creditcard.csv')
```

```
In [21]: credit_card_data.head()
```

Out[21]:

	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

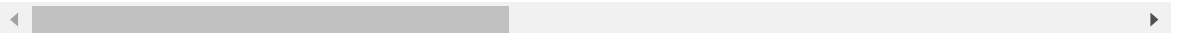


```
In [22]: credit_card_data.tail()
```

Out[22]:

	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



```
In [23]: credit_card_data.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
```

```
In [24]: credit_card_data.isnull().sum()
```

```
Out[24]: 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
```

```
In [25]: credit_card_data['Class'].value_counts()
```

```
Out[25]: 0    284315
          1      492
          Name: Class, dtype: int64
```

```
In [26]: legit = credit_card_data[credit_card_data.Class == 0]
          fraud = credit_card_data[credit_card_data.Class == 1]
```

```
In [27]: print(legit.shape)
          print(fraud.shape)
```

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

```
In [28]: legit.Amount.describe()
```

```
Out[28]: 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
```

```
In [29]: fraud.Amount.describe()
```

```
Out[29]: 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
```

```
In [30]: credit_card_data.groupby('Class').mean()
```

```
Out[30]:
```

	Time	V1	V2	V3	V4	V5	V6	V
<b>Class</b>								
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.00963
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.56873

2 rows × 30 columns

```
In [31]: legit_sample = legit.sample(n=492)
```

```
In [32]: new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

```
In [33]: new_dataset.head()
```

```
Out[33]:
```

	Time	V1	V2	V3	V4	V5	V6	V7
116315	74225.0	1.040422	-0.645040	0.719917	-0.722675	-1.020289	-0.135606	-0.563518
245709	152871.0	2.292102	-1.182032	-1.512987	-1.761744	-0.758521	-0.920015	-0.645555
49028	43883.0	-2.179315	0.707457	1.347898	-0.046303	-0.233723	2.038246	-0.920347
248495	153966.0	-0.959710	0.622997	-0.213053	-0.849642	1.173397	-1.556615	2.443682
136006	81504.0	1.286516	0.099337	0.027363	-0.187848	-0.197086	-0.978010	0.263119

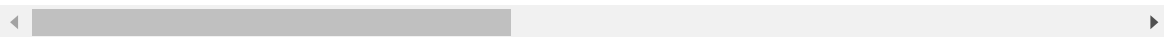
5 rows × 31 columns

```
In [34]: new_dataset.tail()
```

```
Out[34]:
```

	Time	V1	V2	V3	V4	V5	V6	V7
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050

5 rows × 31 columns



```
In [35]: new_dataset['Class'].value_counts()
```

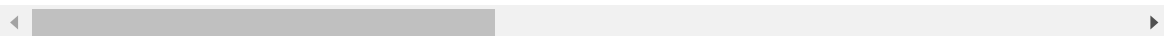
```
Out[35]: 0    492
         1    492
         Name: Class, dtype: int64
```

```
In [36]: new_dataset.groupby('Class').mean()
```

```
Out[36]:
```

	Time	V1	V2	V3	V4	V5	V6	V
Class								
0	94114.278455	0.114365	-0.080838	0.056436	-0.081280	-0.014571	-0.171028	-0.04706
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.56873

2 rows × 30 columns



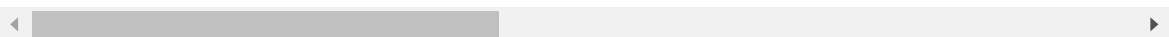
```
In [37]: X = new_dataset.drop(columns='Class', axis=1)
         Y = new_dataset['Class']
```

In [39]: X

Out[39]:

	Time	V1	V2	V3	V4	V5	V6	V7
<b>116315</b>	74225.0	1.040422	-0.645040	0.719917	-0.722675	-1.020289	-0.135606	-0.563518
<b>245709</b>	152871.0	2.292102	-1.182032	-1.512987	-1.761744	-0.758521	-0.920015	-0.645555
<b>49028</b>	43883.0	-2.179315	0.707457	1.347898	-0.046303	-0.233723	2.038246	-0.920347
<b>248495</b>	153966.0	-0.959710	0.622997	-0.213053	-0.849642	1.173397	-1.556615	2.443682
<b>136006</b>	81504.0	1.286516	0.099337	0.027363	-0.187848	-0.197086	-0.978010	0.263119
...	...	...	...	...	...	...	...	...
<b>279863</b>	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850
<b>280143</b>	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170
<b>280149</b>	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739
<b>281144</b>	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002
<b>281674</b>	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050

984 rows × 30 columns



In [40]: Y

Out[40]:

116315	0
245709	0
49028	0
248495	0
136006	0
...	..
279863	1
280143	1
280149	1
281144	1
281674	1

Name: Class, Length: 984, dtype: int64

In [41]: X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, str

In [42]: print(X.shape, X\_train.shape, X\_test.shape)

(984, 30) (787, 30) (197, 30)

In [43]: model = LogisticRegression()

In [44]: model.fit(X\_train, Y\_train)

Out[44]: LogisticRegression()

In [45]: X\_train\_prediction = model.predict(X\_train)  
 training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

```
In [46]: print('Accuracy on Training data : ', training_data_accuracy)
```

Accuracy on Training data : 0.9440914866581956

```
In [47]: X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
In [48]: print('Accuracy score on Test Data : ', test_data_accuracy)
```

Accuracy score on Test Data : 0.9187817258883249

```
In [ ]:
```