In [4]:
```python
#Import necessary libraries
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
```

In [10]:
```python
#loading dataset into Pandas Dataset
data = pd.read_csv(r"C:\Users\LIZZIE\Downloads\creditcard_2023.csv")
```

In [11]:
```python
data.head()
```

Out[11]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	C
0	0	-0.260648	-0.469648	2.496266	-0.083724	0.129681	0.732898	0.519014	-0.130006	0.727159	...	-0.110552	0.217606	-0.134794	0.165959	0.126280	-0.434824	-0.081230	-0.151045	17982.10	
1	1	0.985100	-0.356045	0.558056	-0.429654	0.277140	0.428605	0.406466	-0.133118	0.347452	...	-0.194936	-0.605761	0.079469	-0.577395	0.190090	0.296503	-0.248052	-0.064512	6531.37	
2	2	-0.260272	-0.949385	1.728538	-0.457986	0.074062	1.419481	0.743511	-0.095576	-0.261297	...	-0.005020	0.702906	0.945045	-1.154666	-0.605564	-0.312895	-0.300258	-0.244718	2513.54	
3	3	-0.152152	-0.508959	1.746840	-1.090178	0.249486	1.143312	0.518269	-0.065130	-0.205698	...	-0.146927	-0.038212	-0.214048	-1.893131	1.003963	-0.515950	-0.165316	0.048424	5384.44	
4	4	-0.206820	-0.165280	1.527053	-0.448293	0.106125	0.530549	0.658849	-0.212660	1.049921	...	-0.106984	0.729727	-0.161666	0.312561	-0.414116	1.071126	0.023712	0.419117	14278.97	

5 rows × 31 columns

In [12]:
```python
data.tail()
```

Out[12]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
568625	568625	-0.833437	0.061886	-0.899794	0.904227	-1.002401	0.481454	-0.370393	0.189694	-0.938153	...	0.167503	0.419731	1.288249	-0.900861	0.560661	-0.006018	3.308968	0.081564
568626	568626	-0.670459	-0.202896	-0.068129	-0.267328	-0.133660	0.237148	-0.016935	-0.147733	0.483894	...	0.031874	0.388161	-0.154257	-0.846452	-0.153443	1.961398	-1.528642	1.704306
568627	568627	-0.311997	-0.004095	0.137526	-0.035893	-0.042291	0.121098	-0.070958	-0.019997	-0.122048	...	0.140788	0.536523	-0.211100	-0.448909	0.540073	-0.755836	-0.487540	-0.268741
568628	568628	0.636871	-0.516970	-0.300889	-0.144480	0.131042	-0.294148	0.580568	-0.207723	0.893527	...	-0.060381	-0.195609	-0.175488	-0.554643	-0.099669	-1.434931	-0.159269	-0.076251
568629	568629	-0.795144	0.433236	-0.649140	0.374732	-0.244976	-0.603493	-0.347613	-0.340814	0.253971	...	0.534853	-0.291514	0.157303	0.931030	-0.349423	-1.090974	-1.575113	0.722936

5 rows × 31 columns

In [13]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568630 entries, 0 to 568629
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   id      568630 non-null  int64  
 1   V1      568630 non-null  float64
 2   V2      568630 non-null  float64
 3   V3      568630 non-null  float64
 4   V4      568630 non-null  float64
 5   V5      568630 non-null  float64
 6   V6      568630 non-null  float64
 7   V7      568630 non-null  float64
 8   V8      568630 non-null  float64
 9   V9      568630 non-null  float64
 10  V10     568630 non-null  float64
 11  V11     568630 non-null  float64
 12  V12     568630 non-null  float64
 13  V13     568630 non-null  float64
 14  V14     568630 non-null  float64
 15  V15     568630 non-null  float64
 16  V16     568630 non-null  float64
 17  V17     568630 non-null  float64
 18  V18     568630 non-null  float64
 19  V19     568630 non-null  float64
 20  V20     568630 non-null  float64
 21  V21     568630 non-null  float64
 22  V22     568630 non-null  float64
 23  V23     568630 non-null  float64
 24  V24     568630 non-null  float64
 25  V25     568630 non-null  float64
 26  V26     568630 non-null  float64
 27  V27     568630 non-null  float64
 28  V28     568630 non-null  float64
 29  Amount  568630 non-null  float64
 30  Class   568630 non-null  int64  
dtypes: float64(29), int64(2)
memory usage: 134.5 MB
```

In [14]:
```python
#checking the number of missing values in each column
data.isnull().sum()
```

Out[14]:
```
id        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 [30]:
```python
# iterating the columns
for col in data.columns:
    print(col)
```

```
id
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
```

In [33]:
```python
data['id'].value_counts().describe()
```

Out[33]:
```
count    568630.0
mean          1.0
std           0.0
min           1.0
25%           1.0
50%           1.0
75%           1.0
max           1.0
Name: count, dtype: float64
```

In [35]:
```python
data['id'].value_counts()
```

Out[35]:
```
id
0         1
379089    1
379083    1
379084    1
379085    1
         ..
189537    1
189536    1
189535    1
189534    1
568629    1
Name: count, Length: 568630, dtype: int64
```

In [36]:
```python
data['Class'].value_counts()
```

Out[36]:
```
Class
0    284315
1    284315
Name: count, dtype: int64
```

This dataset is balanced

0--> Normal transactions, 1--> Fraudulent transactions

In [39]:
```python
#separating the data for analysis
legit = data[data.Class==0]
fraud= data[data.Class==1]
```

In [40]:
```python
print(legit.shape)
print(fraud.shape)
```

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

In [41]:
```python
#statistical measures of the data
legit.Amount.describe()
```

Out[41]:
```
count    284315.000000
mean      12026.313506
std        6929.500715
min          50.120000
25%        6034.540000
50%       11996.900000
75%       18040.265000
max       24039.930000
Name: Amount, dtype: float64
```

In [42]:
```python
fraud.Amount.describe()
```

Out[42]:
```
count    284315.000000
mean      12057.601763
std        6909.750891
min          50.010000
25%        6074.640000
50%       12062.450000
75%       18033.780000
max       24039.930000
Name: Amount, dtype: float64
```

In [43]:
```python
#compare the values for both transactions
data.groupby('Class').mean()
```

Out[43]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	
Class																			
0	142442.987714	0.505761	-0.491878	0.682095	-0.735981	0.338639	0.435088	0.491234	-0.144294	0.585522	...	-0.179851	-0.10964	-0.014098	-0.010255	0.130107	-0.061847	-0.071052	-0.21
1	426186.012286	-0.505761	0.491878	-0.682095	0.735981	-0.338639	-0.435088	-0.491234	0.144294	-0.585522	...	0.179851	0.10964	0.014098	0.010255	-0.130107	0.061847	0.071052	0.21

2 rows × 30 columns

Under- Sampling

Build a sample dataset containing similar distribution of normal transactions and fraudulent distribution

Number of legit transactions--> 284315, Number of fraudulent transactions--> 284315

In [46]:
```python
legit_sample= legit.sample(n=800)
fraud_sample= fraud.sample(n=800)
```

concatenating two DataFrames

In [47]:
```python
new_dataset= pd.concat([legit_sample,fraud_sample],axis=0)
```

In [48]:
```python
new_dataset.head()
```

Out[48]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
219384	219384	-0.131479	0.141993	0.953635	0.821038	0.313966	0.773068	0.543296	-0.067266	-0.350559	...	-0.024193	0.637213	-0.182004	0.066360	-0.350867	1.100661	-0.336394	0.309819
25475	25475	1.041276	-0.357327	0.754930	-0.361057	0.087896	-0.121182	0.459637	-0.200888	0.503815	...	-0.206941	-0.682337	0.111932	0.805898	0.285300	0.226614	-0.269472	-0.052861
260489	260489	0.102409	-0.135197	0.163678	-1.176827	1.173452	1.142177	0.666484	-0.036804	0.433093	...	-0.216930	-0.491295	0.074837	-1.703644	-0.920203	0.516805	0.229243	0.179272
64424	64424	1.360050	-0.715281	0.273319	-1.518089	0.029049	-0.036458	0.304912	-0.244800	-0.572337	...	-0.191298	-0.204535	-0.219987	-0.715932	1.308355	-0.064942	-0.236511	-0.099585
146783	146783	-0.282249	0.105555	0.092166	-1.010210	0.490826	0.430134	0.729320	-0.275519	1.690744	...	-0.277331	-0.212349	0.385182	0.480674	-2.352328	1.425932	0.235341	0.800793

5 rows × 31 columns

In [49]:
```python
new_dataset.tail()
```

Out[49]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
412896	412896	-0.817935	0.490379	-0.796826	1.317520	0.345683	-0.757911	-0.166235	-0.287446	-0.431873	...	0.087022	0.401271	-0.086284	-1.044008	0.765322	0.649851	-2.434707	1.147278
321307	321307	1.546416	-0.118042	-0.056220	0.821752	0.562230	0.258668	0.408186	-0.128399	0.384191	...	-0.164900	-0.189393	-0.004138	-0.725525	0.047640	0.349719	-0.188060	0.000712
533104	533104	-0.247515	0.096539	0.013155	-0.140256	0.130185	-0.381091	-0.029821	0.004379	-0.340204	...	0.105203	-0.063353	-0.321200	-0.379078	0.519708	-1.170409	0.485805	0.607464
420213	420213	-0.614155	-1.259334	-0.561114	0.614677	-1.065015	1.734313	2.986113	-0.481076	0.453404	...	-0.236834	-0.206658	-1.889192	0.687181	-0.729296	-0.170756	1.476179	-1.356248
294712	294712	-0.731295	1.572177	-1.595154	2.063410	-0.732428	-1.614182	-1.368899	0.961123	-1.731828	...	0.620444	0.214858	0.423992	-1.922600	-0.968179	1.328986	2.358059	1.709709

5 rows × 31 columns

In [50]:
```python
new_dataset['Class'].value_counts()
```

Out[50]:
```
Class
0    800
1    800
Name: count, dtype: int64
```

In [51]:
```python
new_dataset.groupby('Class').mean()
```

Out[51]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	
Class																			
0	144482.72250	0.488293	-0.517348	0.710511	-0.744012	0.340790	0.425064	0.545705	-0.151963	0.591005	...	-0.226200	-0.104454	-0.010676	-0.049085	0.128647	-0.098589	-0.048508	-0.21
1	422918.52625	-0.529468	0.511008	-0.706815	0.755236	-0.379165	-0.428741	-0.512588	0.130669	-0.618796	...	0.210003	0.064983	0.072461	0.092950	-0.163914	0.062800	0.095186	0.16

2 rows × 30 columns

Splitting the data into features and targets

In [53]:
```python
X= new_dataset.drop(columns='Class',axis=1)
Y= new_dataset['Class']
```

In [54]:
```python
print(X)
```

```
            id        V1        V2        V3        V4        V5        V6  \
219384  219384 -0.131479  0.141993  0.953635  0.821038  0.313966  0.773068   
25475    25475  1.041276 -0.357327  0.754930 -0.361057  0.087896 -0.121182   
260489  260489  0.102409 -0.135197  0.163678 -1.176827  1.173452  1.142177   
64424    64424  1.360050 -0.715281  0.273319 -1.518089  0.029049 -0.036458   
146783  146783 -0.282249  0.105555  0.092166 -1.010210  0.490826  0.430134   
...        ...       ...       ...       ...       ...       ...       ...   
412896  412896 -0.817935  0.490379 -0.796826  1.317520  0.345683 -0.757911   
321307  321307  1.546416 -0.118042 -0.056220  0.821752  0.562230  0.258668   
533104  533104 -0.247515  0.096539  0.013155 -0.140256  0.130185 -0.381091   
420213  420213 -0.614155 -1.259334 -0.561114  0.614677 -1.065015  1.734313   
294712  294712 -0.731295  1.572177 -1.595154  2.063410 -0.732428 -1.614182   

              V7        V8        V9  ...       V20       V21       V22  \
219384  0.543296 -0.067266 -0.350559  ... -0.078191 -0.024193  0.637213   
25475   0.459637 -0.200888  0.503815  ... -0.243899 -0.206941 -0.682337   
260489  0.666484 -0.036804  0.433093  ... -0.075986 -0.216930 -0.491295   
64424   0.304912 -0.244800 -0.572337  ... -0.431520 -0.191298 -0.204535   
146783  0.729320 -0.275519  1.690744  ...  1.041652 -0.277331 -0.212349   
...          ...       ...       ...  ...       ...       ...       ...   
412896 -0.166235 -0.287446 -0.431873  ... -0.425166  0.087022  0.401271   
321307  0.408186 -0.128399  0.384191  ... -0.458814 -0.164900 -0.189393   
533104 -0.029821  0.004379 -0.340204  ...  0.093243  0.105203 -0.063353   
420213  2.986113 -0.481076  0.453404  ... -1.040854 -0.236834 -0.206658   
294712 -1.368899  0.961123 -1.731828  ...  1.251361  0.620444  0.214858   

             V23       V24       V25       V26       V27       V28    Amount  
219384 -0.182004  0.066360 -0.350867  1.100661 -0.336394  0.309819  21461.61  
25475   0.111932  0.805898  0.285300  0.226614 -0.269472 -0.052861  18933.90  
260489  0.074837 -1.703644 -0.920203  0.516805  0.229243  0.179272   3258.76  
64424  -0.219987 -0.715932  1.308355 -0.064942 -0.236511 -0.099585  12237.93  
146783  0.385182  0.480674 -2.352328  1.425932  0.235341  0.800793  21600.29  
...          ...       ...       ...       ...       ...       ...       ...  
412896 -0.086284 -1.044008  0.765322  0.649851 -2.434707  1.147278  13572.59  
321307 -0.004138 -0.725525  0.047640  0.349719 -0.188060  0.000712  12084.23  
533104 -0.321200 -0.379078  0.519708 -1.170409  0.485805  0.607464   9324.72  
420213 -1.889192  0.687181 -0.729296 -0.170756  1.476179 -1.356248  18629.54  
294712  0.423992 -1.922600 -0.968179  1.328986  2.358059  1.709709  19547.81  

[1600 rows x 30 columns]
```

In [55]:
```python
print(Y)
```

```
219384    0
25475     0
260489    0
64424     0
146783    0
         ..
412896    1
321307    1
533104    1
420213    1
294712    1
Name: Class, Length: 1600, dtype: int64
```

split the data into training data and testing data

In [56]:
```python
X_train,X_test,Y_train,Y_test= train_test_split(X,Y, test_size=0.2, stratify=Y, random_state=2)
```

In [57]:
```python
print(X.shape, X_train.shape, X_test.shape)
```

```
(1600, 30) (1280, 30) (320, 30)
```

Model Training

Logistic Regression

In [60]:
```python
model= LogisticRegression()
```

In [61]:
```python
#training the model with the training data
model.fit(X_train, Y_train)
```

Out[61]:
```
▾ LogisticRegression
LogisticRegression()
```

Model Evaluation

Accuracy Score

In [64]:
```python
#accuracy on training data
X_train_prediction= model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction, Y_train)
```

In [65]:
```python
print('accuracy on training data:',training_data_accuracy)
```

```
accuracy on training data: 0.796875
```

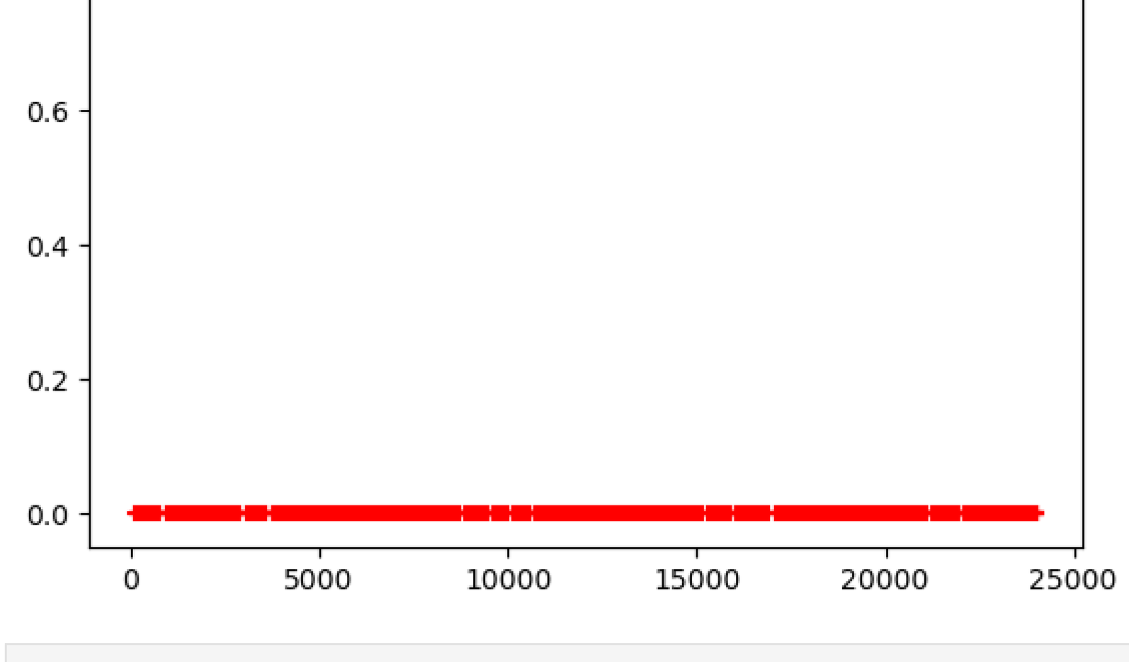
In [67]:
```python
#accuracy on the test data
X_test_prediction= model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction, Y_test)
```

In [68]:
```python
print('accuracy on test data:',test_data_accuracy)
```

```
accuracy on test data: 0.78125
```

In [76]:
```python
#plotting a scatter plot on the data
plt.scatter(new_dataset.Amount,new_dataset.Class,marker='+',color='red')
```

Out[76]:
```
<matplotlib.collections.PathCollection at 0x22265f83b90>
```



In [ ]: