In [32]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [34]:
```
credit_data = pd.read_csv(r"C:\Users\jasja\Downloads\archive (4)\creditcard
credit_data.head()
```

Out[34]:

	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 [35]:
```
credit_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 [36]:
```
credit_data.isnull().sum()
```

Out[36]:
```
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 [37]:
```
credit_data["Class"].value_counts()
```

Out[37]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [38]:
```
#separating the data for analysis
legit =credit_data[credit_data.Class==0]
fraud = credit_data[credit_data.Class==1]
legit.shape
```

Out[38]:
```
(284315, 31)
```

In [39]:
```
legit['Amount'].describe()
```

Out[39]:
```
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 [40]:
```
fraud['Amount'].describe()
```

Out[40]:
```
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 [41]:
```
credit_data.groupby('Class').mean()
```

Out[41]:

Class	Time	V1	V2	V3	V4	V5	V6	V
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.0096
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.5687

2 rows × 30 columns

In [42]:
```
legit_sample= legit.sample(492)
```

In [43]:
```
new_data = pd.concat([legit_sample,fraud], axis =0)
new_data.head()
```

Out[43]:

	Time	V1	V2	V3	V4	V5	V6	V7
196998	131794.0	-1.429201	1.808884	-0.210138	0.768587	0.317984	-0.301567	0.593217
68254	52899.0	-1.529130	-0.794546	1.538048	-2.498540	-1.201862	0.199774	-0.609701
118459	75089.0	1.137041	-0.023618	0.584986	0.502923	-0.712457	-0.817085	-0.100671
153529	99324.0	0.637890	-0.036912	-1.739914	-0.999703	3.457883	3.169310	0.131887
62298	50218.0	1.133963	-0.127570	0.592740	0.895993	-0.685885	-0.460257	-0.181356

5 rows × 31 columns

In [44]:
```
x = new_data.drop(columns='Class',axis=1)
y = new_data['Class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,str
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(787, 30) (197, 30) (787,) (197,)
```

In [45]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
C:\Users\jasja\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[45]:

▾ LogisticRegression

LogisticRegression()

In [46]:
```
x_train_prediction = model.predict(x_train)
training_accuracy = accuracy_score(x_train_prediction, y_train)
print('Accuracy on Training data: ', training_accuracy)
```

```
Accuracy on Training data:  0.9504447268106735
```

In [47]:
```
x_test_predict = model.predict(x_test)
testing_accuracy = accuracy_score(x_test_predict,y_test)
print('Accuracy on Testing data: ', testing_accuracy)
```

```
Accuracy on Testing data:  0.9441624365482234
```

In [ ]: