In [11]:
```
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 [12]:
```
credited_card_df = pd.read_csv(r'C:\Users\Admin\OneDrive\Desktop\creditcard.csv')
credited_card_df.head()
```

Out[12]:

	x	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

5 rows × 31 columns

In [13]:
```
credited_card_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284809 entries, 0 to 284808
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   x       284805 non-null  float64
 1   V1      284809 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  float64
dtypes: float64(31)
memory usage: 67.4 MB
```

In [14]:
```
credited_card_df.shape
```

Out[14]:
```
(284809, 31)
```

In [15]:
```
credited_card_df['Class'].value_counts()
```

Out[15]:
```
0.0    284315
1.0       492
Name: Class, dtype: int64
```

In [16]:
```
credited_card_df.describe()
```

Out[16]:

	x	V1	V2	V3	V4	V5	
count	284805.000000	284809.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8
mean	94813.765629	0.759751	-8.252296e-13	-9.637438e-13	8.316234e-13	1.592013e-13	4
std	47488.287552	288.675554	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.3
min	0.000000	-56.407510	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.8
25%	54201.000000	-0.920372	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7
50%	84691.000000	0.018109	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2
75%	139321.000000	1.315648	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3
max	172792.000000	120883.000000	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.3

8 rows × 31 columns

In [17]:
```
legit= credited_card_df[credited_card_df.Class==0]
fraud= credited_card_df[credited_card_df['Class']==1]
```

In [18]:
```
legit
```

Out[18]:

	x	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0986
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0851
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2476
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3774
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2705
...	...	...	...	...	...	...	...	...	
284804	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.3053
284805	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.2948
284806	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.7084
284807	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.6791
284808	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.4146

284315 rows × 31 columns

In [19]:
```
fraud
```

Out[19]:

	x	V1	V2	V3	V4	V5	V6	V7	V8
541	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391657
623	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067794
4920	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399147
6108	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248778
6329	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496358
...	...	...	...	...	...	...	...	...	...
279865	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211
280145	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525
280151	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158
281146	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733
281676	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384

492 rows × 31 columns

In [20]:
```
fraud.shape
```

Out[20]:
```
(492, 31)
```

In [21]:
```
legit.shape
```

Out[21]:
```
(284315, 31)
```

In [22]:
```
legit_sample = legit.sample(n=492)
credited_card_df = pd.concat([legit_sample,fraud],axis=0)
```

In [23]:
```
credited_card_df['Class'].value_counts()
```

Out[23]:
```
0.0    492
1.0    492
Name: Class, dtype: int64
```

In [24]:
```
credited_card_df.groupby('Class').mean()
```

Out[24]:

Class	x	V1	V2	V3	V4	V5	V6	V7	
0.0	95917.508130	-0.157794	0.103825	0.075746	-0.066376	0.002951	0.041324	-0.069352	-0.0624
1.0	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.5706

2 rows × 30 columns

In [25]:
```
x = credited_card_df.drop('Class',axis=1)
y = credited_card_df['Class']
```

In [26]:
```
x.shape
```

Out[26]:
```
(984, 30)
```

In [27]:
```
y.shape
```

Out[27]:
```
(984,)
```

In [30]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y,random_sta
```

In [34]:
```
model = LogisticRegression()
model.fit(x_train,y_train)
ypred = model.predict(x_test)
```

In [35]:
```
accuracy_score(ypred,y_test)
```

Out[35]:
```
0.9187817258883249
```

In [ ]: