LogisticRegression:(01-06-23)

In [80]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [81]:

```
df = pd.read_csv(r"C:\Users\thara\Downloads\archive.zip")
df
```

26	0	0	-1.00000	-1.00000	0.00000	0.00000	-1.00000	1.00000	1.00000	-0.37500	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000
27	1	0	1.00000	0.08380	1.00000	0.17387	1.00000	-0.13308	0.98172	0.64520	1.00000	0.47904	1.00000	0.59113	1.00000	0.70758	1.00000	0.82777
28	0	0	-1.00000	-1.00000	1.00000	1.00000	1.00000	-1.00000	-1.00000	1.00000	1.00000	-1.00000	-1.00000	-1.00000	0.00000	0.00000	1.00000	1.00000
29	1	0	1.00000	-0.14236	1.00000	-0.16256	1.00000	-0.23656	1.00000	-0.07514	1.00000	-0.25010	1.00000	-0.26161	1.00000	-0.21975	1.00000	-0.38606
30	1	0	1.00000	-1.00000	1.00000	1.00000	1.00000	-1.00000	1.00000	-1.00000	1.00000	-1.00000	1.00000	-0.01840	1.00000	-1.00000	1.00000	1.00000
31	1	0	0.88208	-0.14639	0.93408	-0.11057	0.92100	-0.16450	0.88307	-0.17036	0.88462	-0.31809	0.85269	-0.31463	0.82116	-0.35924	0.80681	-0.33632
32	1	0	0.71253	-0.02595	0.41287	-0.23067	0.98019	-0.09473	0.99709	-0.10236	1.00000	-0.10951	0.58965	1.00000	0.83726	-1.00000	0.82270	-0.17863
33	1	0	1.00000	-0.15899	0.72314	0.27686	0.83443	-0.58388	1.00000	-0.28207	1.00000	-0.49863	0.79962	-0.12527	0.76837	0.14638	1.00000	0.39337
34	1	0	0.66161	-1.00000	1.00000	1.00000	1.00000	-0.67321	0.80893	-0.40446	1.00000	-1.00000	1.00000	-0.89375	1.00000	0.73393	0.17589	0.70982
35	1	0	1.00000	0.00433	1.00000	-0.01209	1.00000	-0.02960	1.00000	-0.07014	0.97839	-0.06256	1.00000	-0.06544	0.97261	-0.07917	0.92561	-0.13665
36	0	0	1.00000	1.00000	1.00000	-1.00000	0.00000	0.00000	0.00000	0.00000	-1.00000	-1.00000	0.00000	0.00000	0.00000	0.00000	-1.00000	1.00000
37	1	0	0.91241	0.04347	0.94191	0.02280	0.94705	0.05345	0.93582	0.01321	0.91911	0.06348	0.92766	0.12067	0.92048	0.06211	0.88899	0.12722
38	1	0	1.00000	0.02461	0.99672	0.04861	0.97545	0.07143	0.61745	1.00000	0.91036	0.11147	0.88462	0.53640	0.82077	0.14137	0.76929	0.15189

In [82]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [83]:

```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 350 Rows and 35 columns
```

In [84]:

```
df.head()
```

Out[84]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.17755	0.59755	-0.44945	0.60536	-0.38223	0.84356	-0
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-0.69707	-0.51685	-0.97515	0.05499	-0
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	0.00827	0.54591	0.00299	0.83775	-0
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-1.00000	0.14516	0.54094	-0
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-0.00712	0.34395	-0.27457	0.52940	-0
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.00000	0.00000	-0.04572	-0.15540	-0.00343	-0

In [85]:

```
features_matrix = df.iloc[:,0:34]
```

In [86]:

```
target_vector = df.iloc[:,-1]
```

In [87]:

```
print('The Features Matrix Has %d Rows And %d columns(s)'%(features_matrix.shape))
print('The Target Matrix Has %d Rows And %d Columns(s)'%(np.array(target_vector).reshape(-1, 1).shape))
```

```
The Features Matrix Has 350 Rows And 34 columns(s)
The Target Matrix Has 350 Rows And 1 Columns(s)
```

In [88]:

```
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [89]:

```
algorithm = LogisticRegression(penalty=None,dual=False, tol=1e-4,C=1.0, fit_intercept=True,intercept_scaling=1,
                               class_weight=None,random_state=None,solver='lbfgs',max_iter=10000,
                               multi_class='auto',verbose=0, warm_start=False, n_jobs=None,l1_ratio=None)
```

In [90]:

```
Logistic_Regression_Model = algorithm.fit(features_matrix_standardized,target_vector)
```

In [91]:

```
192, 0.56971, -0.29674, 0.36946, -0.47357, 0.56811, -0.51171, 0.4107800000000003, -0.4616800000000003, 0.21266, -0.3409, 0.42267, -0.54487
```

In [92]:

```
predictions = Logistic_Regression_Model.predict(observation)
print('The Model predicted The observation To Belong To Class %s'%(predictions))
```

```
The Model predicted The observation To Belong To Class ['g']
```

In [93]:

```
print('The Algorithm Was Trained To predict The One Of The Classes: %s'%(algorithm.classes_))
```

```
The Algorithm Was Trained To predict The One Of The Classes: ['b' 'g']
```

In [94]:

```
print("""The Model Says The Probability Of The observation We Passed belonging To The Class ['b'] is %s"""
      %(algorithm.predict_proba(observation)[0][0]))
print()
print("""The Model Says The Probability Of The observation We Passed belonging To The Class ['g'] is %s"""
      %(algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class ['b'] is 3.2643660690556686e-05

The Model Says The Probability Of The observation We Passed belonging To The Class ['g'] is 0.9999673563393094
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: