In [1]:

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

In [2]:

```
df=pd.read_csv(r"C:\Users\Arshad Shaik\Downloads\archive (1).zip")
df
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.5117
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.2656
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.4022
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.9069
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.6515
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.0153
...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.0420
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.0136
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.0319
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.0209
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.1511

350 rows × 35 columns

In [3]:

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

In [4]:

```
print('This DataFrame ha %d Rows and %d Columns'%(df.shape))
```

```
This DataFrame ha 350 Rows and 35 Columns
```

In [5]:

```
df.head(8)
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.
5	1	0	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79766	-0.
6	0	0	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00000	-1.
7	1	0	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92570	-0.

In [6]:

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

In [7]:

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

In [8]:

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

```
The Features Matrix Has 350 Rows And 34 Columns
The Features Matrix Has 350 Rows And 1 Columns
```

In [9]:

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

In [10]:

```
algorithm = LogisticRegression(penalty=None,dual=False, tol=1e-4,C=1.0, fit_intercept=True,i
 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 [11]:

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

In [12]:

```
999999,-0.17755,0.59755,-0.44945,0.60536,-0.38223,0.8435600000000001,-0.38542,0.58212,-0.3219
```

In [13]:

```
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 [14]:

```
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 [15]:

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

```
The Model Says The Probability Of The observation We Passed belonging To The C
lass ['b'] is 2.5112558470263835e-05

The Model Says The Probability Of The observation We Passed belonging To The C
lass ['g'] is 0.9999748874415297
```

In [ ]: