In [4]:

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

In [5]:

```
df=pd.read_csv(r"C:\Users\poorn\Downloads\archive (7).zip")
df
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03
...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92

In [6]:

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

In [7]:

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

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

In [9]:

```
df.head()
```

Out[9]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.5087
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.7308
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.0000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.5279
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.0378

In [10]:

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

In [11]:

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

In [12]:

```
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
```

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

In [13]:

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

In [14]:

```
algorithm=LogisticRegression(penalty=None,dual=False,tol=1e-4,C=1.0,fit_intercept=True,i
                             random_state=None,solver='lbfgs',max_iter=1000,multi_class=
                             n_jobs=None,l1_ratio=None)
```

In [16]:

```

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

In [17]:

```
observation=[[1,0,0.99539,-0.5889,0.85242999999999999,0.02306,0.83397999999999,-0.37708,1
             0.59755,-0.44945,0.60536,-0.38223,0.84356000000000001,-0.38542,0.58212,-0.3
             0.56811,-0.51171,0.41078000000000003,-0.46168000000000003,0.21256,-0.3409,0
```

In [19]:

```
predictions=Logistic_Regression_Model.predict(observation)
print('The Model predicted the observation to belog to class %s'%(predictions))
```

```
The Model predicted the observation to belog to class ['g']
```

In [20]:

```
print('The algorithm was trained to predict one of the two classes:%s'%(algorithm.classe
```

```
The algorithm was trained to predict one of the two classes:['b' 'g']
```

In [21]:

```
print("""The model says the probability of the obserbvation we passedbelonging to class[
 %(algorithm.predict_proba(observation)[0][0]))
print()
print("""The model says the probability of the observation we passed belonging to class[
 %(algorithm.predict_proba(observation)[observation[0][1]]))
```

```
The model says the probability of the obserbvation we passedbelonging to c
lass['b']is 0.0

The model says the probability of the observation we passed belonging to c
lass['g']is [0. 1.]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: