In [21]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [22]:
```
df=pd.read_csv(r"C:\Users\DELL\OneDrive\Desktop\baaru\ionosphere_data.csv")
df
```

Out[22]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	c
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	
5	True	False	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	
6	True	False	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	
7	False	False	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	
8	True	False	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	
9	True	False	-0.01864	-0.08459	0.00000	0.00000	0.00000	0.00000	
10	True	False	1.00000	0.06655	1.00000	-0.18388	1.00000	-0.27320	

In [24]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [25]:
```
print('The DataFrame has %d Rows and %d columns'%(df.shape))
```

```
The DataFrame has 351 Rows and 35 columns
```

In [26]:
```
df.head()
```

Out[26]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	colum
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.000
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.000
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.889
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.000
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.771

In [27]:
```
features_matrix=df.iloc[:,0:34]
```

In [28]:
```
target_vector=df.iloc[:,-1]
```

In [29]:
```
print('The Feature Matrix Has %d Rows and %d Column(s)'%(features_matrix.shape)
print('The Target Matrix Has %d Rows and %d Column(s)'%(np.array(target_vector)
```

```
The Feature Matrix Has 351 Rows and 34 Column(s)
The Target Matrix Has 351 Rows and 1 Column(s)
```

In [35]:
```
print('The Feature Matrix Has %d Rows and %d Column(s)'%(features_matrix.shape)
print('The Target Matrix Has %d Rows and %d Column(s)'%(np.array(target_vector)
```

```
The Feature Matrix Has 351 Rows and 34 Column(s)
The Target Matrix Has 351 Rows and 1 Column(s)
```

In [36]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [37]:
```
lgorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercep
```

In [38]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_interce
```

In [39]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vec
```

In [40]:
```
observation=[[1,0,0.99539,-0.085889,0.8524299999999999,0.02306,0.83397999999995
              0.59755,-0.44945,0.60536,-0.38223,0.8435600000000001,-0.38542,0.5
              0.56811,-0.51171,0.4107800000000003,-0.4616800000000003,0.21260,-
```

In [41]:
```
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 [42]:
```
print('The algorithm was Trained to predict one of the Two Classes %s'%(algorit
```

```
The algorithm was Trained to predict one of the Two Classes ['b' 'g']
```

In [43]:
```
print(""" The Model says The probabilty of the observation we passed Belonging
print()
print(""" The Model says The probabilty of the observation we passed Belonging
```

```
 The Model says The probabilty of the observation we passed Belonging to clas
s['b'] Is 0.007773084032494881

 The Model says The probabilty of the observation we passed Belonging to clas
s['g'] Is 0.9922269159675051
```

In [ ]: