In [10]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
df=pd.read_csv(r"C:\Users\Svijayalakshmi\Downloads\ionosphere.csv")
df
```

Out[10]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	...	co
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	...	-
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-(
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-(
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	(
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-(
...	...	...	...	...	...	...	...	...	...	...	...	
346	True	False	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-(
347	True	False	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	(
348	True	False	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	(
349	True	False	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-(
350	True	False	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-

351 rows × 35 columns

In [11]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
print('This DataFrame has %d rows and %d columns'%(df.shape))
```

```
This DataFrame has 351 rows and 35 columns
```

In [12]:

```
df.head()
```

Out[12]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	column_
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.8524
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.5087
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.7308
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.0000
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.5279

In [13]:

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

In [14]:

```
target_vector=df.iloc[:,-1]
print('The Feature 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 Feature Matrix has 351 Rows and 34 columns(s)
The Target Matrix has 351 Rows and 1 columns(s)
```

In [15]:

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

In [20]:

```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,cla
```

In [22]:

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

In [24]:

```
observation=[[1,0,0.99539,-0.05889,0.854299999999999,0.02306,0.8339799999999999,-0.37708,1.0,0.0376,0.85242999
```

In [26]:

```
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 [27]:

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

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

In [28]:

```
print(" " "The Model says the probability of the observation we passed belonging to class['b'] Is %s" " "%(alg
print()
```

```
 The Model says the probability of the observation we passed belonging to class['b'] Is 0.007759
545690606995
```

In [29]:

```
print(" " "The Model says the probability of the observation we passed belonging to class['g'] Is %s" " "%(alg
```

```
 The Model says the probability of the observation we passed belonging to class['g'] Is 0.992240
454309393
```

In [ ]: