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\raja\Downloads\archive\ionosphere.csv")
df
```

Out[2]:

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

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 has %d Rows and %d Columns'%(df.shape))
```

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

In [5]:

```
df.head()
```

Out[5]:

	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 [7]:

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

In [8]:

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

In [9]:

```
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 [25]:

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

In [26]:

```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,fit_intercept=True,interce
                            l1_ratio=None)
```

In [27]:

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

In [28]:

```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.8339799999999999,-0.3770
```

In [29]:

```
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 [30]:

```
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 [31]:

```
print("""The model says the probability of the observation we passed belonging to class[
print()
print("""the model says the probability of the observation we passed belonging to class[
```

```
The model says the probability of the observation we passed belonging to c
lass['b']is 0.016301793950620813

the model says the probability of the observation we passed belonging to c
lass['g']is 0.9836982060493792
```