In [16]:

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

In [17]:

```
df=pd.read_csv(r"C:\Users\monim\Downloads\ionosphere.csv")
df
```

Out[17]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.1775
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.6774
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.0534
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	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	0.52798	-0.2027
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.0630
5	1	0	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79766	-0.4792
6	0	0	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00000	-1.0000
7	1	0	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92570	-0.4356
8	1	0	-0.01864	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	-0.45663	-0.3817
9	1	0	1.00000	0.06655	1.00000	-0.18388	1.00000	-0.27320	1.00000	-0.43107	1.00000	-0.4134

In [28]:

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

In [29]:

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

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

In [30]:

```
df.head()
```

Out[30]:

	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 [31]:

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

In [32]:

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

In [37]:

```
print('The Features matrix Has %d rows And %d column(s)'%(features_matrix.shape))
print('The target vector Has %d rows And %d column(s)'%(np.array(target_vector).reshape(
```

```
The Features matrix Has 350 rows And 34 column(s)
The target vector Has 350 rows And 1 column(s)
```

In [47]:

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

In [51]:

```
algorithm=LogisticRegression(max_iter=1000)
```

In [52]:

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

In [54]:

```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.8339799999999999,-0.37708,
              -0.44945,0.60536,-0.38223,0.8435600000000001,-0.38542,0.58212,-0.32192,0.5
```

In [56]:

```
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 [57]:

```
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 [63]:

```
print(""" The model says the probability of the observation we passed belonging 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)[0][1]))
```

```
 The model says the probability of the observation we passed belonging to
class['b'] is 0.008043929702268082

 The model says the probability of the observation we passed belonging to
class ['g'] is 0.9919560702977319
```

In [ ]:

```

```