In [33]:

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

In [34]:

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

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **53** | 1 | 0 | 0.95659 | 0.08143 | 0.97487 | -0.05667 | 0.97165 | -0.08484 | 0.96097 | -0.06561 | 0.94717 | 0.0127 |
| **54** | 1 | 0 | 0.08333 | -0.20685 | -1.00000 | 1.00000 | -1.00000 | 1.00000 | 0.71875 | 0.47173 | -0.82143 | -0.6272 |
| **55** | 1 | 0 | 1.00000 | -0.02259 | 1.00000 | -0.04494 | 1.00000 | -0.06682 | 1.00000 | -0.08799 | 1.00000 | 0.5617 |
| **56** | 0 | 0 | -1.00000 | 1.00000 | 1.00000 | -1.00000 | -1.00000 | 1.00000 | 0.00000 | 0.00000 | 1.00000 | 1.0000 |
| **57** | 1 | 0 | 1.00000 | 0.05812 | 0.94525 | 0.07418 | 0.99952 | 0.13231 | 1.00000 | -0.01911 | 0.94846 | 0.0703 |
| **58** | 1 | 0 | 0.17188 | -1.00000 | -1.00000 | 1.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | -1.00000 | 1.0000 |
| **59** | 1 | 0 | 1.00000 | 0.09771 | 1.00000 | 0.12197 | 1.00000 | 0.22574 | 0.98602 | 0.09237 | 0.94930 | 0.1921 |
| **60** | 1 | 0 | 0.01667 | -0.35625 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.0000 |
| **61** | 1 | 0 | 1.00000 | 0.16801 | 0.99352 | 0.16334 | 0.94616 | 0.33347 | 0.91759 | 0.22610 | 0.91408 | 0.3710 |
| **62** | 1 | 0 | 0.63816 | 1.00000 | 0.20833 | -1.00000 | 1.00000 | 1.00000 | 0.87719 | 0.30921 | -0.66886 | 1.0000 |
| **63** | 1 | 0 | 1.00000 | -0.41457 | 1.00000 | 0.76131 | 0.87060 | 0.18593 | 1.00000 | -0.09925 | 0.93844 | 0.4799 |
| **64** | 1 | 0 | 0.84783 | 0.10598 | 1.00000 | 0.39130 | 1.00000 | -1.00000 | 0.66938 | 0.08424 | 1.00000 | 0.2703 |

In [35]:

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

In [36]:

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

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

In [37]:

```
df.head()
```

Out[37]:

| | 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 [38]:

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

In [39]:

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

In [40]:

```
print('The Features Matrix Has %d Rows And %d Column(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 Column(s)
The Target Matrix Has 350 Rows and 1 columns(s)
```

In [41]:

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

In [44]:

```
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 [54]:

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

In [55]:

```
observation=[[1,0,0.99539,-0.5889,0.8524299999999999,0.02306,0.83397999999999,-0.37708,1
             0.59755,-0.44945,0.60536,-0.38223,0.8435600000000001,-0.38542,0.58212,-0.3
             0.56811,-0.51171,0.41078000000000003,-0.46168000000000003,0.21256,-0.3409,0
```

In [48]:

```
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 [49]:

```
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 [52]:

```
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 [ ]:

```

```