In [1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [7]:
```
df=pd.read_csv(r"file:///D:\Users\DELL\Downloads\ionosphere_data.csv")
df
```

Out[7]:

	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	0.85
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52
5	True	False	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03
6	True	False	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79
7	False	False	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00
8	True	False	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92
9	True	False	-0.01864	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	-0.45

In [8]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [9]:
```
print('This DataFrame has %d rows and %d columns'%(df.shape))
```

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

In [10]:
```
df.head()
```

Out[10]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	column_k	c
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.85243	
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	

In [11]:
```
features_matrix=df.iloc[:,0:34]
```

In [17]:
```
target_vector=df.iloc[:,-1]
```

In [18]:
```
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(-1,1).shape))
```

```
The Features matrix Has 351 rows And 34 column(s)
The target vector Has 351 rows And 1 column(s)
```

In [22]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [23]:
```
algorithm=LogisticRegression(max_iter=1000)
```

In [24]:
```
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,-0.37708,1.0,0.0376
              -0.17755,0.59755,-0.44945,0.60536,-0.38223,0.8435600000000001,-0.38542,0.58212,-0.3219
              0.36946,-0.47357,0.56811,-0.51171,0.41078000000000003,-0.46168000000000003,0.21266,-0.
              0.18641,-0.453]]
```

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.classes_))
```

```
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['b']is %s"
      %(algorithm.predict_proba(observation)[0][0]))
```

```
The model says the probability of the observation we passed belonging to class['b']is 0.00777393160
0142836
```

In [32]:
```
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 0.99222606839
98572
```

In [ ]:
```

```