In [12]:

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

In [13]:

```
df=pd.read_csv(r"C:\Users\shaik\Desktop\202U1A3344\ionosphere_data.csv")
df
```

Out[13]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	co
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0
5	True	False	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0
6	True	False	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0
7	False	False	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0
8	True	False	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0

In [14]:

```
pd.set_option('display.max_row',100000000)
pd.set_option('display.max_column',10000000)
pd.set_option('display.width',95)
```

In [15]:

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

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

In [16]:

```
df.head()
```

Out[16]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	c
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	

In [17]:

```
features_matrix=df.iloc[:,0:34]
target_vector=df.iloc[:,-1]
```

In [18]:

```
print('The features matrix has %d rows and %d columns'
      %(features_matrix.shape))
```

```
The features matrix has 351 rows and 34 columns
```

In [19]:

```
print('The target matrix has %d rows and %d columns'
      %(np.array(target_vector).reshape(-1,1).shape))
```

```
The target matrix has 351 rows and 1 columns
```

In [20]:

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

In [25]:

```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,
                             C=1.0,fit_intercept=True,intercept_scaling=1,
                             class_weight=None,
                             solver='lbfgs',max_iter=100,multi_class='auto',verbose=0,
                             warm_start=False,n_jobs=None)
```

In [26]:

```
Logistic_Regression_model=algorithm.fit(feature_matrix_standardized,target_vector)
```

In [27]:

```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.8339799999999999,
              -0.37708,1.0,0.0376,0.8524299999999999,
              -0.17755,0.59755,-0.44945,0.60536,-0.38223,0.8435600000000001,
              -0.38542,0.58212,-0.32192,0.56971,-0.29674,
              0.36946,-0.47357,0.56811,-0.51171,0.41078000000000003,
              -0.46168000000000003,0.21266,-0.3409,0.42267,-0.54487,
              0.18641,-0.453]]
```

In [36]:

```
predictions=Logistic_Regression_model.predict(observation)
print("The model predicted the observation to belong to class %s"
      %(predictions))
print("The Algorithm was trained to predict one of two classes:%s"
      %(algorithm.classes_))
print("The model says the probability of the observation we passed belonging to class['b
      %(algorithm.predict_proba(observation)[0][0]))
```

```
The model predicted the observation to belong to class ['g']
The Algorithm was trained to predict one of two classes:['b' 'g']
The model says the probability of the observation we passed belonging to c
lass['b']is 0.0077739316001402825
```

In [37]:

```
print("The model says the probability of the observation we passed belonging to class['g
      %(algorithm.predict_proba(observation)[0][1]))
```

```
The model says the probability of the observation we passed belonging to c
lass['g']is 0.9922260683998597
```

In [ ]:

```

```