In [26]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [27]:
```
df=pd.read_csv(r"C:\Users\my pc\Downloads\ionosphere.csv")
df
```

:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.17755	0.59755	-
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.00000	
5	1	0	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79766	-0.47929	0.78225	-
6	0	0	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00000	-1.00000	0.00000	
7	1	0	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92570	-0.43569	0.94510	-
8	1	0	-0.01864	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	-0.45663	-0.38172	0.00000	
9	1	0	1.00000	0.06655	1.00000	-0.18388	1.00000	-0.27320	1.00000	-0.43107	1.00000	-0.41349	0.96232	-
10	1	0	1.00000	-0.54210	1.00000	-1.00000	1.00000	-1.00000	1.00000	0.36217	1.00000	-0.41119	1.00000	

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	0.85243.1	-0.17755	0.
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.

In [31]:
```
features_matrix=df.iloc[:,0:34]
target_matrix=df.iloc[:,-1]
```

In [32]:
```
print("The feature matrix has %d rows and %d columns"%(features_matrix.shape))
```

```
The feature matrix has 350 rows and 34 columns
```

In [33]:
```
print("The target matrix has %d rows and %d columns"%(np.array(target_matrix).reshape(
```

```
The target matrix has 350 rows and 1 columns
```

In [36]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [37]:
```
algorithm=LogisticRegression(max_iter=100)
```

In [39]:
```
Logistic_Regression_model=algorithm.fit(features_matrix_standardized,target_matrix)
```

In [40]:
```
df
```

9539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.17755	0.59755	-0.44945	0.60536
0000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-0.69707	-0.51685
0000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	0.00827	0.54591
0000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-1.00000
0000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-0.00712	0.34395
2337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.00000	0.00000	-0.04572
7588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79766	-0.47929	0.78225	-0.50764	0.74628
0000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00000	-1.00000	0.00000	0.00000	0.00000
6355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92570	-0.43569	0.94510	-0.40668	0.90392
1864	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	-0.45663	-0.38172	0.00000	0.00000	-0.33656
0000	0.06655	1.00000	-0.18388	1.00000	-0.27320	1.00000	-0.43107	1.00000	-0.41349	0.96232	-0.51874	0.90711
0000	-0.54210	1.00000	-1.00000	1.00000	-1.00000	1.00000	0.36217	1.00000	-0.41119	1.00000	1.00000	1.00000

In [42]:
```
observation=[[1,0,0.99539,-0.05889,0.85243,0.02306,0.83398,-0.37708,1.1,0.03760,0.8524
0.60536,-0.38223,0.84356,-0.38542,0.58212,-0.32192,0.56971,-0.29674,0.36946,-0.47357,0
             -0.46168,0.21266,-0.34090,0.42267,-0.54487,0.18641,-0.45300]]
```

In [43]:
```
predictions=Logistic_Regression_model.predict(observation)
```

In [44]:
```
print("The model prdicted the observation belong to class %s"%(predictions))
```

```
The model prdicted the observation belong to class ['g']
```

In [46]:
```
print("The model says the probability of the observation we passed belonging to class[
      %(algorithm.predict_proba(observation)[0][0]))
```

```
The model says the probability of the observation we passed belonging to class['b'] i
s 0.007527265557674467
```

In [47]:
```
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['g'] i
s 0.9924727344423255
```