In [5]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [6]:
```
df=pd.read_csv(r"C:\Users\Welcome\Downloads\ionosphere.csv")
df
```

Out[6]:

	atr1	atr2	atr3	atr4	atr5	atr6	atr7	atr8	atr9	atr10	...	atr26	atr27	atr28	atr29	atr30	atr31	a
0	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	...	-0.51171	0.41078	-0.46168	0.21266	-0.34090	0.42267	-0.54
1	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-0.19040	-0.11593	-0.16626	-0.06
2	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22145	0.43100	-0.17365	0.60436	-0.24
3	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00000	1.00000	-0.20099	0.25682	1.00
4	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53206	0.02431	-0.62197	-0.05707	-0.59
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
346	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.00123	1.00000	0.12815	0.86660	-0.10
347	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.04925	0.93159	0.08168	0.94066	-0.00
348	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.02542	0.92120	0.02242	0.92459	0.00
349	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.07760	0.82983	-0.17238	0.96022	-0.03
350	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.04822	0.78207	-0.00703	0.75747	-0.06

351 rows × 35 columns

In [7]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [8]:
```
print('This DataFrame ha %d Rows and %d Columns'%(df.shape))
```

```
This DataFrame ha 351 Rows and 35 Columns
```

In [9]:
```
df.head()
```

Out[9]:

	atr1	atr2	atr3	atr4	atr5	atr6	atr7	atr8	atr9	atr10	atr11	atr12	atr13	atr14	atr15	atr16	atr17	
0	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.85243	-0.17755	0.59755	-0.44945	0.60536	-0.38223	0.84356	-0.
1	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-0.69707	-0.51685	-0.97515	0.05499	-0.
2	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	0.00827	0.54591	0.00299	0.83775	-0.
3	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-1.00000	0.14516	0.54094	-0.
4	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-0.00712	0.34395	-0.27457	0.52940	-0.

In [10]:
```
features_matrix=df.iloc[:,0:34]
```

In [11]:
```
target_vector=df.iloc[:,-1]
```

In [12]:
```
print('The Features Matrix Has %d Rows And %d Columns'%(features_matrix.shape))
print('The Features Matrix Has %d Rows And %d Columns'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The Features Matrix Has 351 Rows And 34 Columns
The Features Matrix Has 351 Rows And 1 Columns
```

In [16]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [17]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,class_weight=None,ra
```

In [18]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [19]:
```
.29674,0.36946,-0.47357,0.56811,-0.51171,0.4107800000000003,-0.4616800000000003,0.21266,-0.3409,0.42267,-0.54487,0.18641,-0.453]
```

In [20]:
```
predictions=Logistic_Regression_Model.predict(Observation)
print('The Model Predicted The Observations To Belong To Class %s'%(predictions))
```

```
The Model Predicted The Observations To Belong To Class ['g']
```

In [21]:
```
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 [22]:
```
l Says The Probability Of The Observation we Passed Belonging To class['b']Is %s"""%(algorithm.predict_proba(Observation)[0][0]
l Says The Probability Of The Observation we Passed Belonging To class['g']Is %s"""%(algorithm.predict_proba(Observation)[0][1]
```

```
The Model Says The Probability Of The Observation we Passed Belonging To class['b']Is 0.007759545690611991
The Model Says The Probability Of The Observation we Passed Belonging To class['g']Is 0.992240454309388
```

In [ ]: