In [2]:
```
#1 IONOSPHERE
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
df=pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Data Sets\ionosphere_data.csv")
df
```

Out[4]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	...	column_z	column_aa	column_ab	column
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	...	-0.51171	0.41078	-0.46168	0.2
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-0.1
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22145	0.4
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00000	1.0
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53206	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
346	True	False	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.00123	1.0
347	True	False	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.04925	0.9
348	True	False	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.02542	0.9
349	True	False	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.07760	0.8
350	True	False	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.04822	0.7

351 rows × 35 columns

In [5]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
print('This dataframe has %d rows and %d columns'%(df.shape))
```

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

In [6]:
```
df.head(10)
```

Out[6]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	column_k	column_l	column_m	column_n	col
0	True	False	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
1	True	False	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
2	True	False	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
3	True	False	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
4	True	False	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
5	True	False	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.00000	0.00000	-0
6	True	False	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79766	-0.47929	0.78225	-0.50764	0
7	False	False	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00000	-1.00000	0.00000	0.00000	0
8	True	False	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92570	-0.43569	0.94510	-0.40668	0
9	True	False	-0.01864	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	-0.45663	-0.38172	0.00000	0.00000	-0

In [7]:
```
features_matrix=df.iloc[:,0:34]
```

In [8]:
```
target_vector=df.iloc[:,-1]
```

In [9]:
```
print('The features matrix has %d rows and %d columns'%(features_matrix.shape))
```

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

In [10]:
```
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 [11]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [12]:
```
ight=None,random_state=None,solver='lbfgs',max_iter=100,multi_class='auto',verbose=0,warm_start=False,n_jobs=None,l1_ratio=None)
```

In [13]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [14]:
```
)674,0.36946,-0.47357,0.56811,-0.51171,0.41078000000000003,-0.46168000000000003,0.21266,-0.3409,0.42267,-0.54487,0.18641,-0.453]]
```

In [15]:
```
predictions=Logistic_Regression_Model.predict(observation)
```

In [16]:
```
print('The model predicted the observation to belong to class %s'%(predictions))
```

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

In [17]:
```
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 [18]:
```
says the probability of the observation we passed belonging to class['0'] is %s"""%(algorithm.predict_proba(observation)[0][0]))
```

```
The model says the probability of the observation we passed belonging to class['0'] is 0.007773931600142836
```

In [19]:
```
says the probability of the observation we passed belonging to class['1'] is %s"""%(algorithm.predict_proba(observation)[0][1]))
```

```
The model says the probability of the observation we passed belonging to class['1'] is 0.9922260683998572
```