

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

```
In [3]: df=pd.read_csv(r"C:\Users\Jayadeep\Downloads\ionosphere.csv")
df
```

Out[3]:

	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_ac	column_ad	column_ae
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	0.21266	-0.34090	0.42267
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	-0.19040	-0.11593	-0.16626
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	0.43100	-0.17365	0.60436
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	1.00000	-0.20099	0.25682
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	0.02431	-0.62197	-0.05707
...
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	1.00000	0.12815	0.86660
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	0.93159	0.08168	0.94066
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	0.92120	0.02242	0.92459
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	0.82983	-0.17238	0.96022
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	0.78207	-0.00703	0.75747

351 rows × 35 columns



```
In [4]: pd.set_option('display.max_rows',1000000000)
pd.set_option('display.max_columns',1000000000)
pd.set_option('display.width',95)
```

```
In [5]: print('This DataFrame has %d rows and %d columns'%(df.shape))
```

This DataFrame has 351 rows and 35 columns

In [6]: df.head()

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	column_o	column_p	column_q
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	0.60536	-0.38223	0.84356
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	-0.51685	-0.97515	0.05499
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	0.54591	0.00299	0.83775
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	-1.00000	0.14516	0.54094
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	0.34395	-0.27457	0.52940

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

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

In [9]: print('The Feature Matrix has %d Rows and %d columns(s)%(features_matrix.shape))
print('The Target Matrix has %d Rows and %d columns(s)%(np.array(target_vector).reshape(-1,1).shape))

The Feature Matrix has 351 Rows and 34 columns(s)
The Target Matrix has 351 Rows and 1 columns(s)

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

In [24]: cept_scaling=1,class_weight=None,random_state=None,solver='lbfgs',max_iter=100,multi_class='auto',verbose=0,warm_start=False,n_jobs=None,l1_ratio=None)

In [25]: Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)

In [29]: observation=[[1,0,0.99539,-0.05889,0.8542999999999999,0.02306,0.8339799999999999,-0.37708,1.0,0.0376,0.8524299999999999,-0.17755,0.59755,-0.44945,0.60536

In [30]: 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 [32]: 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 [34]: print(" " "The Model says the probability of the observation we passed belonging to class['b'] Is %s" " "%(algorithm.predict_proba(observation)[0][0]))  
print()
```

The Model says the probability of the observation we passed belonging to class['b'] Is 0.005322793491169264

```
In [35]: print(" " "The Model 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['g'] Is 0.9946772065088307

conclusion: The data set we have taken is suitable for this model.The Model says the probability of the observation we passed belonging to class['g'] Is 0.9946772065088307