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

In [2]:
```
df = pd.read_csv(r"C:\Users\Dell\Downloads\ionosphere.csv")
df
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01
...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15

350 rows × 35 columns

In [3]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [4]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 350 Rows and 35 columns
```

In [5]:
```
df.head()
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	·
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	·
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	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	·
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	·

In [6]:
```
features_matrix = df.iloc[:,0:34]
```

In [7]:
```
target_vector = df.iloc[:,-1]
```

In [8]:
```
print('The Features Matrix Has %d Rows And %d columns(s)'%(features_matrix.sha
print('The Target Matrix Has %d Rows And %d Columns(s)'%(np.array(target_vecto
```

```
The Features Matrix Has 350 Rows And 34 columns(s)
The Target Matrix Has 350 Rows And 1 Columns(s)
```

In [9]:
```
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [10]:
```
algorithm = LogisticRegression(penalty=None,dual=False, tol=1e-4,C=1.0, fit_in
                               class_weight=None,random_state=None,solver='lbf
                               multi_class='auto',verbose=0, warm_start=False,
```

In [11]:
```
Logistic_Regression_Model = algorithm.fit(features_matrix_standardized,target_
```

In [18]:
```
observation = [[1, 0, 0.99539, -0.05889, 0.8524299999999999, 0.02306, 0.833979
```

In [19]:
```
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 [20]:
```
print('The Algorithm Was Trained To predict The One Of The Classes: %s'%(algor
```

```
The Algorithm Was Trained To predict The One Of The Classes: ['b' 'g']
```

In [23]:
```
print("""The Model Says The Probability Of The observation We Passed belongir
      %(algorithm.predict_proba(observation)[0][0]))
print()
print("""The Model Says The Probability Of The observation We Passed belonging
      %(algorithm.predict_proba(observation)[0][1]))
```

```
"The Model Says The Probability Of The observation We Passed belonging To Th
e Class ['b'] is 3.9740609243499314e-05

The Model Says The Probability Of The observation We Passed belonging To The
Class ['g'] is 0.9999602593907565
```

In [ ]: