In [4]:
```
pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\teppa\appdata\local\progra
ms\python\python310\lib\site-packages (0.0.post5)
Note: you may need to restart the kernel to use updated packages.
```

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

In [2]:
```
df=pd.read_csv(r"C:\Users\teppa\Desktop\p\ionosphere_data.csv")
df
```

Out[2]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	col
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0
...	...	...	...	...	...	...	...	...	
346	True	False	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0
347	True	False	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0
348	True	False	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0
349	True	False	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0
350	True	False	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0

351 rows × 35 columns

In [3]:
```
pd.set_option('display.max_rows',1000000000)
pd.set_option('display.max_columns',10000000)
pd.set_option('display.width',95)
```

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

```
This DataFrame has351 Rows and 35 columns
```

In [5]:
```
df.head(4)
```

Out[5]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	colun
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.0(
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.0(
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.8{
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.0(

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 column(s)'%(features_matrix.shap
print('The Target matrix Has %d Rows and %d column(s)'%(np.array (target_vecto
```

```
The features matrix has 351 Rows and 34 column(s)
The Target matrix Has 351 Rows and 1 column(s)
```

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

In [10]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_interc
```

In [11]:
```
logistic_Regression_model=algorithm.fit(features_matrix_standardized,target_ve
```

In [12]:
```
observations=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.8339799999999
              -0.38542,0.58212,-0.32192,0.56971,-0.29674,0.36946,-0.47357,0.56
```

In [13]:
```
prediction=logistic_Regression_model.predict(observations)
print('The model predicted the observation to Belong To class %s' %(predictior
```

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

In [14]:
```
print('The algorithm was Trained one of the Two classes:%s' %(algorithm.classe
```

```
The algorithm was Trained one of the Two classes:['b' 'g']
```

In [15]:
```
print("""The model  says the probability of the observation we passed Belongin
print()
print("""The model says the probability of the observation we passed Belonging
```

```
The model  says the probability of the observation we passed Belonging to cla
ss['b'] Is 0.007773931600142836

The model says the probability of the observation we passed Belonging to clas
s['g'] Is 0.9922260683998572
```

In [ ]: