In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
SONARdata=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\sonar.all-data.csv",header=None)
```

In [3]:
```
SONARdata.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57	58	59	6
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027	0.0065	0.0159	0.0072	0.0167	0.0180	0.0084	0.0090	0.0032	
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084	0.0089	0.0048	0.0094	0.0191	0.0140	0.0049	0.0052	0.0044	
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232	0.0166	0.0095	0.0180	0.0244	0.0316	0.0164	0.0095	0.0078	
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121	0.0036	0.0150	0.0085	0.0073	0.0050	0.0044	0.0040	0.0117	
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031	0.0054	0.0105	0.0110	0.0015	0.0072	0.0048	0.0107	0.0094	

5 rows × 61 columns

In [4]:
```
#number of rows and columns
SONARdata.shape
```

Out[4]:
```
(208, 61)
```

In [5]:
```
#describe mathamatical data
SONARdata.describe()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	...	50	51
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	...	208.000000	208.000000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.178003	0.208259	...	0.016069	0.013420
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.118387	0.134416	...	0.012008	0.009634
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.007500	0.011300	...	0.000000	0.000800
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.097025	0.111275	...	0.008425	0.007275
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.152250	0.182400	...	0.013900	0.011400
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.233425	0.268700	...	0.020825	0.016725
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.682800	0.710600	...	0.100400	0.070900

8 rows × 60 columns

In [6]:
```
SONARdata[60].value_counts()
```

Out[6]:
```
M    111
R     97
Name: 60, dtype: int64
```

In [7]:
```
SONARdata.groupby(60).mean()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54
60																
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492	0.251022	...	0.019352	0.016014	0.011643	0.012185	0.009923
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392	0.159325	...	0.012311	0.010453	0.009640	0.009518	0.008567

2 rows × 60 columns

In [8]:
```
#separting data and labels
X=SONARdata.drop(columns=60,axis=1)
y=SONARdata[60]
```

In [9]:
```
print(X)
print(y)
```

```
        0       1       2       3       4       5       6       7       8   \
0    0.0200  0.0371  0.0428  0.0207  0.0954  0.0986  0.1539  0.1601  0.3109
1    0.0453  0.0523  0.0843  0.0689  0.1183  0.2583  0.2156  0.3481  0.3337
2    0.0262  0.0582  0.1099  0.1083  0.0974  0.2280  0.2431  0.3771  0.5598
3    0.0100  0.0171  0.0623  0.0205  0.0205  0.0368  0.1098  0.1276  0.0598
4    0.0762  0.0666  0.0481  0.0394  0.0590  0.0649  0.1209  0.2467  0.3564
..      ...     ...     ...     ...     ...     ...     ...     ...     ...
203  0.0187  0.0346  0.0168  0.0177  0.0393  0.1630  0.2028  0.1694  0.2328
204  0.0323  0.0101  0.0298  0.0564  0.0760  0.0958  0.0990  0.1018  0.1030
205  0.0522  0.0437  0.0180  0.0292  0.0351  0.1171  0.1257  0.1178  0.1258
206  0.0303  0.0353  0.0490  0.0608  0.0167  0.1354  0.1465  0.1123  0.1945
207  0.0260  0.0363  0.0136  0.0272  0.0214  0.0338  0.0655  0.1400  0.1843

        9   ...      50      51      52      53      54      55      56  \
0    0.2111  ...  0.0232  0.0027  0.0065  0.0159  0.0072  0.0167  0.0180
1    0.2872  ...  0.0125  0.0084  0.0089  0.0048  0.0094  0.0191  0.0140
2    0.6194  ...  0.0033  0.0232  0.0166  0.0095  0.0180  0.0244  0.0316
3    0.1264  ...  0.0241  0.0121  0.0036  0.0150  0.0085  0.0073  0.0050
4    0.4459  ...  0.0156  0.0031  0.0054  0.0105  0.0110  0.0015  0.0072
..      ...  ...     ...     ...     ...     ...     ...     ...     ...
203  0.2684  ...  0.0203  0.0116  0.0098  0.0199  0.0033  0.0101  0.0065
204  0.2154  ...  0.0051  0.0061  0.0093  0.0135  0.0063  0.0063  0.0034
205  0.2529  ...  0.0155  0.0160  0.0029  0.0051  0.0062  0.0089  0.0140
206  0.2354  ...  0.0042  0.0086  0.0046  0.0126  0.0036  0.0035  0.0034
207  0.2354  ...  0.0181  0.0146  0.0129  0.0047  0.0039  0.0061  0.0040

        57      58      59
0    0.0084  0.0090  0.0032
1    0.0049  0.0052  0.0044
2    0.0164  0.0095  0.0078
3    0.0044  0.0040  0.0117
4    0.0048  0.0107  0.0094
..      ...     ...     ...
203  0.0115  0.0193  0.0157
204  0.0032  0.0062  0.0067
205  0.0138  0.0077  0.0031
206  0.0079  0.0036  0.0048
207  0.0036  0.0061  0.0115

[208 rows x 60 columns]
0      R
1      R
2      R
3      R
4      R
      ..
203    M
204    M
205    M
206    M
207    M
Name: 60, Length: 208, dtype: object
```

In [10]:
```
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2,stratify=y)
```

In [11]:
```
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (166, 60)
shape of X_test=  (42, 60)
shape of y_train=  (166,)
shape of y_test=  (42,)
```

In [12]:
```
#traing LogisticRegression model
model=LogisticRegression()
```

In [13]:
```
#training the LogisticRegression model with training data
model.fit(X_train,y_train)
```

Out[13]:
```
LogisticRegression()
```

In [14]:
```
#model evaluation
#accurcy of training data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,y_train)
```

In [15]:
```
print('Accuracy on training data:',training_data_accuracy)
```

```
Accuracy on training data: 0.8072289156626506
```

In [16]:
```
#accurcy of testing data
X_test_prediction=model.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction,y_test)
```

In [17]:
```
print('Accuracy on test data:',testing_data_accuracy)
```

```
Accuracy on test data: 0.8333333333333334
```

In [21]:
```
#Making a Predictive system
#changing the input _data to a numpy array
input_data1=(0.0115,0.0150,0.0136,0.0076,0.0211,0.1058,0.1023,0.0440,0.0931,0.0734,0.0740,0.0622,0.1055,0.1183,0.1721,0.2584,
input_data_numpy_array1=np.asarray(input_data1)
```

In [22]:
```
#reshape the np array as we are predicting for one instance
input_data_reshaped1=input_data_numpy_array1.reshape(1,-1)
```

In [23]:
```
input_data_reshaped1
```

Out[23]:
```
array([[0.0115, 0.015 , 0.0136, 0.0076, 0.0211, 0.1058, 0.1023, 0.044 ,
        0.0931, 0.0734, 0.074 , 0.0622, 0.1055, 0.1183, 0.1721, 0.2584,
        0.3232, 0.3817, 0.4243, 0.4217, 0.4449, 0.4075, 0.3306, 0.4012,
        0.4466, 0.5218, 0.7552, 0.9503, 1.    , 0.9084, 0.8283, 0.7571,
        0.7262, 0.6152, 0.568 , 0.5757, 0.5324, 0.3672, 0.1669, 0.0866,
        0.0646, 0.1891, 0.2683, 0.2887, 0.2341, 0.1668, 0.1015, 0.1195,
        0.0704, 0.0167, 0.0107, 0.0091, 0.0016, 0.0084, 0.0064, 0.0026,
        0.0029, 0.0037, 0.007 , 0.0041]])
```

In [24]:
```
prediction=model.predict(input_data_reshaped1)
print(prediction)
```

```
['R']
```

In [25]:
```
if(prediction[0]=='R'):
    print('The object is Rock')
else:
    print('The object is Mine')
```

```
The object is Rock
```

In [26]:
```
input_data2=(0.0264,0.0071,0.0342,0.0793,0.1043,0.0783,0.1417,0.1176,0.0453,0.0945,0.1132,0.0840,0.0717,0.1968,0.2633,0.4191,
input_data_numpy_array2=np.asarray(input_data2)
```

In [27]:
```
input_data_numpy_array2
```

Out[27]:
```
array([0.0264, 0.0071, 0.0342, 0.0793, 0.1043, 0.0783, 0.1417, 0.1176,
       0.0453, 0.0945, 0.1132, 0.084 , 0.0717, 0.1968, 0.2633, 0.4191,
       0.505 , 0.6711, 0.7922, 0.8381, 0.8759, 0.9422, 1.    , 0.9931,
       0.9575, 0.8647, 0.7215, 0.5801, 0.4964, 0.4886, 0.4079, 0.2443,
       0.1768, 0.2472, 0.3518, 0.3762, 0.2909, 0.2311, 0.3168, 0.3554,
       0.3741, 0.4443, 0.3261, 0.1963, 0.0864, 0.1688, 0.1991, 0.1217,
       0.0628, 0.0323, 0.0253, 0.0214, 0.0262, 0.0177, 0.0037, 0.0068,
       0.0121, 0.0077, 0.0078, 0.0066])
```

In [28]:
```
input_data_numpy_array2= input_data_numpy_array2.reshape(1,-1)
```

In [30]:
```
prediction=model.predict(input_data_numpy_array2)
print(prediction)
```

```
['M']
```

In [32]:
```
if(prediction[0]=='R'):
    print('The object is Rock')
else:
    print('The object is Mine')
```

```
The object is Mine
```

In [ ]:
```

```

In [ ]:
```

```