In [29]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [13]:
```
sonar_data =pd.read_excel('Sonar Data.xlsx',header=None)
```

In [14]:
```
sonar_data
```

Out[14]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57	58	59	60
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	R
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	R
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	R
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	R
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	R
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
203	0.0187	0.0346	0.0168	0.0177	0.0393	0.1630	0.2028	0.1694	0.2328	0.2684	...	0.0116	0.0098	0.0199	0.0033	0.0101	0.0065	0.0115	0.0193	0.0157	M
204	0.0323	0.0101	0.0298	0.0564	0.0760	0.0958	0.0990	0.1018	0.1030	0.2154	...	0.0061	0.0093	0.0135	0.0063	0.0063	0.0034	0.0032	0.0062	0.0067	M
205	0.0522	0.0437	0.0180	0.0292	0.0351	0.1171	0.1257	0.1178	0.1258	0.2529	...	0.0160	0.0029	0.0051	0.0062	0.0089	0.0140	0.0138	0.0077	0.0031	M
206	0.0303	0.0353	0.0490	0.0608	0.0167	0.1354	0.1465	0.1123	0.1945	0.2354	...	0.0086	0.0046	0.0126	0.0036	0.0035	0.0034	0.0079	0.0036	0.0048	M
207	0.0260	0.0363	0.0136	0.0272	0.0214	0.0338	0.0655	0.1400	0.1843	0.2354	...	0.0146	0.0129	0.0047	0.0039	0.0061	0.0040	0.0036	0.0061	0.0115	M

208 rows × 61 columns

In [99]:
```
sonar_data[60].value_counts()
```

Out[99]:
```
60
M    111
R     97
Name: count, dtype: int64
```

In [97]:
```
sonar_data.groupby(60).mean()
```

Out[97]:

60	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54	55	56	57
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	0.008914	0.007825	0.009060
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	0.007430	0.007814	0.006677

2 rows × 60 columns

In [15]:
```
X=sonar_data.drop(columns=60,axis=1)
```

In [16]:
```
Y=sonar_data[60]
```

In [74]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

In [75]:
```
print(X_test,Y_test)
```

```
        0       1       2       3       4       5       6       7       8   \
73   0.0139  0.0222  0.0889  0.0108  0.0215  0.0136  0.0659  0.0954  0.0786
179  0.0394  0.0420  0.0446  0.0551  0.0597  0.1416  0.0956  0.0802  0.1618
172  0.0180  0.0444  0.0476  0.0698  0.1615  0.0887  0.0596  0.1071  0.3175
192  0.0056  0.0267  0.0221  0.0561  0.0936  0.1146  0.0706  0.0996  0.1673
75   0.0202  0.0104  0.0325  0.0239  0.0807  0.1529  0.1154  0.0608  0.1317
49   0.0119  0.0582  0.0623  0.0600  0.1397  0.1883  0.1422  0.1447  0.0487
104  0.0307  0.0523  0.0653  0.0521  0.0611  0.0577  0.0665  0.0664  0.1460
137  0.0430  0.0902  0.0833  0.0813  0.0165  0.0277  0.0569  0.2057  0.3887
58   0.0225  0.0019  0.0075  0.0097  0.0445  0.0906  0.0889  0.0655  0.1624
40   0.0068  0.0232  0.0513  0.0444  0.0249  0.0637  0.0422  0.1130  0.1911
119  0.0261  0.0266  0.0223  0.0749  0.1364  0.1513  0.1316  0.1654  0.1864
140  0.0412  0.1135  0.0518  0.0232  0.0646  0.1124  0.1787  0.2407  0.2682
36   0.0094  0.0166  0.0398  0.0359  0.0681  0.0706  0.1020  0.0893  0.0381
50   0.0353  0.0713  0.0326  0.0272  0.0370  0.0792  0.1083  0.0687  0.0298
131  0.1150  0.1163  0.0866  0.0358  0.0232  0.1267  0.2417  0.2661  0.4346
63   0.0067  0.0096  0.0024  0.0058  0.0197  0.0618  0.0432  0.0951  0.0836
146  0.1021  0.0830  0.0577  0.0627  0.0635  0.1328  0.0988  0.1787  0.1199
7    0.0519  0.0548  0.0842  0.0319  0.1158  0.0922  0.1027  0.0613  0.1465
92   0.0260  0.0192  0.0254  0.0061  0.0352  0.0701  0.1263  0.1080  0.1523
169  0.0130  0.0120  0.0436  0.0624  0.0428  0.0349  0.0384  0.0446  0.1318
112  0.0454  0.0472  0.0697  0.1021  0.1397  0.1493  0.1487  0.0771  0.1171

        9   ...      50      51      52      53      54      55      56  \
73   0.1015  ...  0.0024  0.0062  0.0072  0.0113  0.0012  0.0022  0.0025
179  0.2558  ...  0.0118  0.0146  0.0040  0.0114  0.0032  0.0062  0.0101
172  0.2918  ...  0.0122  0.0122  0.0114  0.0098  0.0027  0.0025  0.0026
192  0.1859  ...  0.0185  0.0072  0.0055  0.0074  0.0068  0.0084  0.0037
75   0.1370  ...  0.0188  0.0127  0.0081  0.0067  0.0043  0.0065  0.0049
49   0.0864  ...  0.0069  0.0025  0.0103  0.0074  0.0123  0.0069  0.0076
104  0.2792  ...  0.0063  0.0321  0.0189  0.0137  0.0277  0.0152  0.0052
137  0.7106  ...  0.0208  0.0176  0.0197  0.0210  0.0141  0.0049  0.0027
58   0.1452  ...  0.0051  0.0034  0.0129  0.0100  0.0044  0.0057  0.0030
40   0.2475  ...  0.0199  0.0173  0.0163  0.0055  0.0045  0.0068  0.0041
119  0.2013  ...  0.0135  0.0222  0.0175  0.0127  0.0022  0.0124  0.0054
140  0.2058  ...  0.0798  0.0376  0.0143  0.0272  0.0127  0.0166  0.0095
36   0.1328  ...  0.0134  0.0141  0.0191  0.0145  0.0065  0.0129  0.0217
50   0.0880  ...  0.0098  0.0163  0.0242  0.0043  0.0202  0.0108  0.0037
131  0.5378  ...  0.0228  0.0099  0.0065  0.0085  0.0166  0.0110  0.0190
63   0.1180  ...  0.0029  0.0048  0.0023  0.0020  0.0040  0.0019  0.0034
146  0.1369  ...  0.1004  0.0709  0.0317  0.0309  0.0252  0.0087  0.0177
7    0.2838  ...  0.0052  0.0081  0.0120  0.0045  0.0121  0.0097  0.0085
92   0.1630  ...  0.0132  0.0118  0.0120  0.0051  0.0070  0.0015  0.0035
169  0.1375  ...  0.0024  0.0084  0.0100  0.0018  0.0035  0.0058  0.0011
112  0.1675  ...  0.0137  0.0120  0.0042  0.0238  0.0129  0.0084  0.0218

         57      58      59
73   0.0059  0.0039  0.0048
179  0.0068  0.0053  0.0087
172  0.0050  0.0073  0.0022
192  0.0024  0.0034  0.0007
75   0.0054  0.0073  0.0054
49   0.0073  0.0030  0.0138
104  0.0121  0.0124  0.0055
137  0.0162  0.0059  0.0021
58   0.0035  0.0021  0.0027
40   0.0052  0.0194  0.0105
119  0.0021  0.0028  0.0023
140  0.0225  0.0098  0.0085
36   0.0087  0.0077  0.0122
50   0.0096  0.0093  0.0053
131  0.0141  0.0068  0.0086
63   0.0034  0.0051  0.0031
146  0.0214  0.0227  0.0106
7    0.0047  0.0048  0.0053
92   0.0008  0.0044  0.0077
169  0.0009  0.0033  0.0026
112  0.0321  0.0154  0.0053

[21 rows x 60 columns] 73     R
179    M
172    M
192    M
75     R
49     R
104    M
137    M
58     R
40     R
119    M
140    M
36     R
50     R
131    M
63     R
146    M
7      R
92     R
169    M
112    M
Name: 60, dtype: object
```

In [76]:
```
model=LogisticRegression()
```

In [77]:
```
X.shape,X_train.shape,X_test.shape
```

Out[77]:
```
((208, 60), (187, 60), (21, 60))
```

In [78]:
```
model.fit(X_train,Y_train)
```

Out[78]:
```
LogisticRegression()
```

Model Evaluation

In [79]:
```
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

In [80]:
```
print(training_data_accuracy * 100)
```

```
81.28342245989305
```

In [81]:
```
X_test_prediction=model.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

In [82]:
```
print(testing_data_accuracy * 100)
```

```
90.47619047619048
```

In [100…
```
## Making a predictive system
```

In [94]:
```
input_data=(0.0217,0.0340,0.0392,0.0236,0.1081,0.1164,0.1398,0.1009,0.1147,0.1777,0.4079,0.4113,0.3973,0.5078,0.6509,0.8073,0.9819,1.0000,0.9407,0.8
input_data_as_numpy_array =np.array(input_data)
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
prediction=model.predict(input_data_reshaped)
print(prediction)

if (prediction=="R"):
    print("The object is a Rock")
else :
    print("The object is a Mine")
```

```
['M']
The object is a Mine
```

In [ ]:

In [ ]: