


Importing the dependencies

```
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
```

Data Collection and Data processing

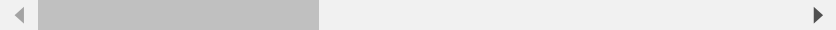
```
# loading the dataset into pandas Dataframe
sonar_data=pd.read_csv('/content/sonar_data.csv',header=None)
```

```
sonar_data.head()
```



	0	1	2	3	4	5	6	
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2

5 rows x 61 columns



```
# number of rows and columns
sonar_data.shape
```

```
(208, 61)
```

```
# deriving some statistical measures of the data
sonar_data.describe()
```



```

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]
```

```
print(Y)
```

```

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
```

Training and testing data

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=1)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(208, 60) (187, 60) (21, 60)
```

Model Training --> Logistic Regression

```
model=LogisticRegression()
```

```
#training the LR model with training data
```

```
model.fit(X_train,Y_train)
```

```

▼ LogisticRegression
LogisticRegression()

```

Model Evaluation

```
#accuracy on training data
```

```
X_train_prediction = model.predict(X_train)
```

```
training_data_accuracy = accuracy_score(X_train_prediction,Y_train)
```

```
print('Accuracy on training data :',training_data_accuracy)
```

```
Accuracy on training data : 0.8342245989304813
```

```
# accuracy on testing data
```

```
X_test_prediction = model.predict(X_test)
```

```
testing_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

```
print("Accuracy on testing data: ",testing_data_accuracy)
```

```
Accuracy on testing data: 0.7619047619047619
```

Making a predictive System

```
input_data=(0.0124,0.0433,0.0604,0.0449,0.0597,0.0355,0.0531,0.0343,0.1052,0.2120,0.1640,0.1901,0.0210,0.0121,0.0203,0.1036,0.1675,0.0418,0.0723,0.0828,0.0494,0.0686,0.1125,0.1741)
```

```
# changing input_data into array
```

```
input_data_as_array = np.asarray(input_data)
```

```
# reshaping the np array as we are predicting for only one instance
```

```
input_data_resaped = input_data_as_array.reshape(1,-1)
```

```
prediction=model.predict(input_data_resaped)
```

```
print(prediction)
```

```
if (prediction[0]=='R'):
```

```
    print("Input data belongs to rock")
```

```
else:
```

```
    print("Input data belongs to mine")
```

```
    ['R']
```

```
Input data belongs to rock
```

```
input_data1=[(0.0124,0.0433,0.0604,0.0449,0.0597,0.0355,0.0531,0.0343,0.1052,0.2120,0.1640,0.1901,0.0210,0.0121,0.0203,0.1036,0.1675,0.0418,0.0723,0.0828,0.0494,0.0686,0.1125,0.1741)]
```

```
input_data1_as_array = np.asarray(input_data1)
```

```
input_data1_resaped= input_data1_as_array.reshape(2,-1)
```

```
prediction1= model.predict(input_data1_resaped)
```

```
print(prediction1)
```

```
['R' 'M']
```

