Rock vs Mine Prediction

In [ ]:
```
#importing the depencies
```

In [2]:
```
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 [ ]:
```
#data collection and data processing
```

In [8]:
```
sonar_data=pd.read_csv('copy of sonar data.csv',header=None)
```

In [9]:
```
sonar_data.head()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	
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.
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.
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.
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.
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

5 rows × 61 columns

In [10]:
```
sonar_data.shape
```

Out[10]:
```
(208, 61)
```

In [11]:
```
sonar_data.describe()#describe the statistical measures of the data
```

Out[11]:

	0	1	2	3	4	5	6	7	
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.00
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.17
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.11
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.00
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.09
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.15
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.23
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.68

8 rows × 60 columns

In [13]:
```
sonar_data[60].value_counts()
```

Out[13]:
```
M    111
R     97
Name: 60, dtype: int64
```

In [ ]:
```
#M-----Mine
#R-----Rock
```

In [14]:
```
sonar_data.groupby(60).mean()
```

Out[14]:

60	0	1	2	3	4	5	6	7	8	9	...
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492	0.251022	... 0.0
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392	0.159325	... 0.0

2 rows × 60 columns

seperating data and label

x=sonar_data.drop(columns=60,axis=1) y=sonar_data[60]

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

print(x) print(y)

In [18]:
```
#training and test data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,stratify=y,random_state=
```

In [19]:
```
print(x.shape,x_test.shape,x_train.shape)
```

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

In [ ]:
```
#model tarining ->>Logistic Regression
```

In [24]:
```
model=LogisticRegression()
```

In [25]:
```
model.fit(x_train,y_train)
```

Out[25]:
```
LogisticRegression
LogisticRegression()
```

In [ ]:
```
#model evaluation
#accuracy on training data
```

In [27]:
```
x_train_prediction=model.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
```

In [29]:
```
print('accuracy on training data:',training_data_accuracy)
```

```
accuracy on training data: 0.8342245989304813
```

In [30]:
```
x_test_prediction=model.predict(x_test)
training_data_accuracy=accuracy_score(x_test_prediction,y_test)
```

In [32]:
```
print('accuracy on test data:',training_data_accuracy)
```

```
accuracy on test data: 0.7619047619047619
```

Making a predictive system

In [36]:
```
input_data=(0.0516,0.0944,0.0622,0.0415,0.0995,0.2431,0.1777,0.2018,0.2611,0.1294,0.2646,(
#changing teh input data into numpy array
input_data_as_numpy_array=np.asarray(input_data)
#reshaped the np arry as we are predicting the on instance
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 Rock')
else:
    print('The object is Mine')
```

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