In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
#read file
df=pd.read_csv('Datasets/Sonar Rock/Copy of sonar data.csv', header= None)
df.head(5)
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0

5 rows × 61 columns

In [4]:
```
df.shape
```

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

In [10]:
```
df[60].value_counts()
```

Out[10]:
```
60
M    111
R     97
Name: count, dtype: int64
```

M= mine R= rock

In [11]:
```
df.groupby(60).mean()
```

Out[11]:

	0	1	2	3	4	5	6	7	
60									
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.21
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.13

2 rows × 60 columns

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (accuracy_score, classification_report, confusion_ma
roc_curve, roc_auc_score, precision_recall_curve)
```

In [8]:
```
df.describe()
```

Out[8]:

	0	1	2	3	4	5	
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.0000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.1217
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.0617
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.0033
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.0809
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.1069
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.1540
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.3729

8 rows × 60 columns

feature selection

In [14]:

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

splitting data

In [17]:

```
x_train, x_test, y_train, y_test= train_test_split(x,y, test_size=0.1, stratify=
```

In [18]:

```
print(x.shape, x_train.shape, x_test.shape)
```

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

Model training

In [19]:

```
model=LogisticRegression()
```

training model

In [25]:

```
model.fit(x_train, y_train)
```

Out[25]:

```
▾ LogisticRegression
LogisticRegression()
```

model evaluation

In [36]:

```
#accuracy of training data
x_train_prediction=model.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction, y_train)
print('Accuracy of training data:',training_data_accuracy*100,'%')
```

```
Accuracy of training data: 83.42245989304813 %
```

```
In [37]: #accuracy of training data
         x_test_prediction=model.predict(x_test)
         test_data_accuracy=accuracy_score(x_test_prediction, y_test)
         print('Accuracy of test data:',test_data_accuracy*100,'%')
```

```
Accuracy of test data: 76.19047619047619 %
```

Prediction model

```
In [40]: new_data= (0.0200,0.0371,0.0428,0.0207,0.0954,0.0986,0.1539,0.1601,0.3109,0.2111

         #converting input data to numpy array
         new_data_array=np.asarray(new_data)

         #reshaping array for prediction
         new_reshaped_data=new_data_array.reshape(1,-1)
```

```
In [41]: prediction=model.predict(new_reshaped_data)
```

```
In [42]: prediction
```

```
Out[42]: array(['R'], dtype=object)
```

```
In [46]: if (prediction[0]=='R'):
             print("Object is Rock")
         else:
             print("Object is Mine")
```

```
Object is Rock
```