```
import pandas as pd
import numpy as np
import matplotlib.pyplot
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

df = pd.read_csv('sonar_data.csv', header=None)

df.head()
```

```
       0       1       2       3       4   ...      56      57      58
59  60
0  0.0200  0.0371  0.0428  0.0207  0.0954  ...  0.0180  0.0084  0.0090
0.0032   R
1  0.0453  0.0523  0.0843  0.0689  0.1183  ...  0.0140  0.0049  0.0052
0.0044   R
2  0.0262  0.0582  0.1099  0.1083  0.0974  ...  0.0316  0.0164  0.0095
0.0078   R
3  0.0100  0.0171  0.0623  0.0205  0.0205  ...  0.0050  0.0044  0.0040
0.0117   R
4  0.0762  0.0666  0.0481  0.0394  0.0590  ...  0.0072  0.0048  0.0107
0.0094   R

[5 rows x 61 columns]
```

```
#num of rows and columns
df.shape
```

```
(208, 61)
```

```
df.describe()
```

```
                0           1           2   ...          57          58
59
count  208.000000  208.000000  208.000000  ...  208.000000  208.000000
208.000000
mean     0.029164    0.038437    0.043832  ...    0.007949    0.007941
0.006507
std      0.022991    0.032960    0.038428  ...    0.006470    0.006181
0.005031
min      0.001500    0.000600    0.001500  ...    0.000300    0.000100
0.000600
25%      0.013350    0.016450    0.018950  ...    0.003600    0.003675
0.003100
50%      0.022800    0.030800    0.034300  ...    0.005800    0.006400
0.005300
75%      0.035550    0.047950    0.057950  ...    0.010350    0.010325
0.008525
max      0.137100    0.233900    0.305900  ...    0.044000    0.036400
0.043900
```

```
[8 rows x 60 columns]
```

```
df[60].value_counts()
```

```
60
M    111
R     97
Name: count, dtype: int64
```

```
df.groupby(60).mean()
```

```
          0         1         2   ...        57        58        59
60                                ...
M   0.034989  0.045544  0.050720  ...  0.009060  0.008695  0.006930
R   0.022498  0.030303  0.035951  ...  0.006677  0.007078  0.006024

[2 rows x 60 columns]
```

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

Xtrain, Xtest, Ytrain, Ytest = train_test_split(x, y, test_size=0.1,
random_state=42, stratify=y)

print(x.shape, Xtrain.shape,Xtest.shape)

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

Model Training

```
model = LogisticRegression()

#Train
model.fit(Xtrain,Ytrain)

LogisticRegression()
```

Model Evaluation

```
#Accuracy score for train data
YtrainPred = model.predict(Xtrain)
accuracyTrain = accuracy_score(Ytrain, YtrainPred)
print(f"Accuracy of training data is: {accuracyTrain * 100:.3f}%")

Accuracy of training data is: 82.353%

#Accuracy score for test data
YtestPred = model.predict(Xtest)
accuracyTest = accuracy_score(Ytest, YtestPred)
print(f"Accuracy of test data is: {accuracyTest * 100:.3f}%")
```

```
Accuracy of test data is: 85.714%
```

Prediction System

```
input_data =
(0.0209,0.0191,0.0411,0.0321,0.0698,0.1579,0.1438,0.1402,0.3048,0.3914
,0.3504,0.3669,0.3943,0.3311,0.3331,0.3002,0.2324,0.1381,0.3450,0.4428
,0.4890,0.3677,0.4379,0.4864,0.6207,0.7256,0.6624,0.7689,0.7981,0.8577
,0.9273,0.7009,0.4851,0.3409,0.1406,0.1147,0.1433,0.1820,0.3605,0.5529
,0.5988,0.5077,0.5512,0.5027,0.7034,0.5904,0.4069,0.2761,0.1584,0.0510
,0.0054,0.0078,0.0201,0.0104,0.0039,0.0031,0.0062,0.0087,0.0070,0.0042
)

#input_data to np array
InputDataNp = np.asarray(input_data)

#reshape the npArray as we are predicting for one instance
inputDataReshaped = InputDataNp.reshape(1, -1)

prediction = model.predict(inputDataReshaped)

if prediction[0] == 'M':
    print("The object is a Mine!")
else:
    print('The object is a Rock')
```

```
The object is a Mine!
```

```
import pickle

#model
with open('Log_Reg.pkl', 'wb') as f:
    pickle.dump(model, f)

#Load model
with open('Log_Reg.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

#Pred load
loaded_prediction = loaded_model.predict(Xtest)

# Validation
print("Prediction from loaded model:", loaded_prediction[:5])
```

```
Prediction from loaded model: ['R' 'M' 'M' 'R' 'M']
```