Rock Vs Mine Prediction Machine Learning Project Using Python

Name: Darsh Kumar

Data Description

The dataset used in this project contains sonar data used in Submarines. The sonar data is used to predict wheater the object outside the Submarines is a Rock or a Mine. The dataset is a labelled dataset which means that the input is associated with the output and a Taget column is also given. The dataset can be used for Supervised Machine Learning Algorithms. In total this dataset has 208 rows and 61 columns in which the first 60 columns has sonar data (input) and the last columns has the output as to wheater it is a rock or a mine. Here I have used Logistic Regression Machine Learning Model to train the model for Prediction as in this case the target column follows Binary classification. For Evaluation Metrics I have used the Accuracy Score.

In [1]:
```
# importing the required dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Next Step: Data Collection and Processing

In [2]:
```
# Loading the dataset
df = pd.read_csv('sonar_data.csv')
df
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	51	
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027	0.0(
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084	0.0(
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232	0.0
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121	0.0(
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031	0.0(
...	...	...	...	...	...	...	...	...	...	...	...	...	
203	0.0187	0.0346	0.0168	0.0177	0.0393	0.1630	0.2028	0.1694	0.2328	0.2684	...	0.0116	0.0(
204	0.0323	0.0101	0.0298	0.0564	0.0760	0.0958	0.0990	0.1018	0.1030	0.2154	...	0.0061	0.0(
205	0.0522	0.0437	0.0180	0.0292	0.0351	0.1171	0.1257	0.1178	0.1258	0.2529	...	0.0160	0.0(
206	0.0303	0.0353	0.0490	0.0608	0.0167	0.1354	0.1465	0.1123	0.1945	0.2354	...	0.0086	0.0(
207	0.0260	0.0363	0.0136	0.0272	0.0214	0.0338	0.0655	0.1400	0.1843	0.2354	...	0.0146	0.0

208 rows × 61 columns

In [3]:
```
# checking the type of dataset
type(df)
```

Out[3]:
```
pandas.core.frame.DataFrame
```

In [4]:
```
# fetching the first five rows of the dataset
df.head()
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	...	51	5
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027	0.006
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084	0.008
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232	0.016
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121	0.003
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031	0.005

5 rows × 61 columns

In [5]:
```
# fetching the last five rows of the dataset
df.tail()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	...	51	
203	0.0187	0.0346	0.0168	0.0177	0.0393	0.1630	0.2028	0.1694	0.2328	0.2684	...	0.0116	0.0(
204	0.0323	0.0101	0.0298	0.0564	0.0760	0.0958	0.0990	0.1018	0.1030	0.2154	...	0.0061	0.0(
205	0.0522	0.0437	0.0180	0.0292	0.0351	0.1171	0.1257	0.1178	0.1258	0.2529	...	0.0160	0.0(
206	0.0303	0.0353	0.0490	0.0608	0.0167	0.1354	0.1465	0.1123	0.1945	0.2354	...	0.0086	0.0(
207	0.0260	0.0363	0.0136	0.0272	0.0214	0.0338	0.0655	0.1400	0.1843	0.2354	...	0.0146	0.0

5 rows × 61 columns

In [6]:
```
# finding the total number of data points in the dataset
df.shape
```

Out[6]: (208, 61)

In [7]:
```
# fetching statistical information of the dataset
df.describe()
```

Out[7]:

	0	1	2	3	4	5	6	
count	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	0.13
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.08
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.00
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.08
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.11
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.16
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.45

8 rows × 60 columns

In [8]:
```
# fetching the general information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 208 entries, 0 to 207
Data columns (total 61 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       208 non-null    float64
 1   1       208 non-null    float64
 2   2       208 non-null    float64
 3   3       208 non-null    float64
 4   4       208 non-null    float64
 5   5       208 non-null    float64
 6   6       208 non-null    float64
 7   7       208 non-null    float64
 8   8       208 non-null    float64
 9   9       208 non-null    float64
 10  10      208 non-null    float64
 11  11      208 non-null    float64
 12  12      208 non-null    float64
 13  13      208 non-null    float64
 14  14      208 non-null    float64
 15  15      208 non-null    float64
 16  16      208 non-null    float64
 17  17      208 non-null    float64
 18  18      208 non-null    float64
 19  19      208 non-null    float64
 20  20      208 non-null    float64
 21  21      208 non-null    float64
 22  22      208 non-null    float64
 23  23      208 non-null    float64
 24  24      208 non-null    float64
 25  25      208 non-null    float64
 26  26      208 non-null    float64
 27  27      208 non-null    float64
 28  28      208 non-null    float64
 29  29      208 non-null    float64
 30  30      208 non-null    float64
 31  31      208 non-null    float64
 32  32      208 non-null    float64
 33  33      208 non-null    float64
 34  34      208 non-null    float64
 35  35      208 non-null    float64
 36  36      208 non-null    float64
 37  37      208 non-null    float64
 38  38      208 non-null    float64
 39  39      208 non-null    float64
 40  40      208 non-null    float64
 41  41      208 non-null    float64
 42  42      208 non-null    float64
 43  43      208 non-null    float64
 44  44      208 non-null    float64
 45  45      208 non-null    float64
 46  46      208 non-null    float64
 47  47      208 non-null    float64
```

```
 48  48      208 non-null    float64
 49  49      208 non-null    float64
 50  50      208 non-null    float64
 51  51      208 non-null    float64
 52  52      208 non-null    float64
 53  53      208 non-null    float64
 54  54      208 non-null    float64
 55  55      208 non-null    float64
 56  56      208 non-null    float64
 57  57      208 non-null    float64
 58  58      208 non-null    float64
 59  59      208 non-null    float64
 60  60      208 non-null    object
dtypes: float64(60), object(1)
memory usage: 99.2+ KB
```

In [9]:
```
# checking for any null values in the dataset
df.isnull().sum()
```

Out[9]:
```
0     0
1     0
2     0
3     0
4     0
     ..
56    0
57    0
58    0
59    0
60    0
Length: 61, dtype: int64
```

In [10]:
```
# finding the count of Rocks and Mine
# M stands for Mines
# R stands for Rocks
df['60'].value_counts()
```

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

In [11]:
```
# grouping the data based on Mines and Rocks
df.groupby('60').mean()
```

Out[11]:

60	0	1	2	3	4	5	6	7	8	
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492	0.25
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392	0.15

2 rows × 60 columns

In [12]:
```
# seperating the data and labels
x = df.drop(columns = '60',axis = 1)
y = df['60']
```

In [13]:
```
# printing the new varaibles
print(x)
```

```
          0       1       2       3       4       5       6       7       8  \
0    0.0200  0.0371  0.0428  0.0207  0.0954  0.0986  0.1539  0.1601  0.3109
1    0.0453  0.0523  0.0843  0.0689  0.1183  0.2583  0.2156  0.3481  0.3337
2    0.0262  0.0582  0.1099  0.1083  0.0974  0.2280  0.2431  0.3771  0.5598
3    0.0100  0.0171  0.0623  0.0205  0.0205  0.0368  0.1098  0.1276  0.0598
4    0.0762  0.0666  0.0481  0.0394  0.0590  0.0649  0.1209  0.2467  0.3564
..      ...     ...     ...     ...     ...     ...     ...     ...     ...
203  0.0187  0.0346  0.0168  0.0177  0.0393  0.1630  0.2028  0.1694  0.2328
204  0.0323  0.0101  0.0298  0.0564  0.0760  0.0958  0.0990  0.1018  0.1030
205  0.0522  0.0437  0.0180  0.0292  0.0351  0.1171  0.1257  0.1178  0.1258
206  0.0303  0.0353  0.0490  0.0608  0.0167  0.1354  0.1465  0.1123  0.1945
207  0.0260  0.0363  0.0136  0.0272  0.0214  0.0338  0.0655  0.1400  0.1843

          9  ...      50      51      52      53      54      55      56  \
0    0.2111  ...  0.0232  0.0027  0.0065  0.0159  0.0072  0.0167  0.0180
1    0.2872  ...  0.0125  0.0084  0.0089  0.0048  0.0094  0.0191  0.0140
2    0.6194  ...  0.0033  0.0232  0.0166  0.0095  0.0180  0.0244  0.0316
3    0.1264  ...  0.0241  0.0121  0.0036  0.0150  0.0085  0.0073  0.0050
4    0.4459  ...  0.0156  0.0031  0.0054  0.0105  0.0110  0.0015  0.0072
..      ...  ...     ...     ...     ...     ...     ...     ...     ...
203  0.2684  ...  0.0203  0.0116  0.0098  0.0199  0.0033  0.0101  0.0065
204  0.2154  ...  0.0051  0.0061  0.0093  0.0135  0.0063  0.0063  0.0034
205  0.2529  ...  0.0155  0.0160  0.0029  0.0051  0.0062  0.0089  0.0140
206  0.2354  ...  0.0042  0.0086  0.0046  0.0126  0.0036  0.0035  0.0034
207  0.2354  ...  0.0181  0.0146  0.0129  0.0047  0.0039  0.0061  0.0040

         57      58      59
0    0.0084  0.0090  0.0032
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]
```

In [14]:
```
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
```

Next Step: Splitting the Data for Training and Testing

In [15]:
```
# splitting the dataset into training and test data
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.1,stratify = y,rar
```

In [16]:
```
# finding the size of the training data after split
print(x.shape,xtrain.shape,xtest.shape)
```

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

In [17]:
```
# printing the size of test data after split
print(y.shape,ytrain.shape,ytest.shape)
```

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

Next Step: Modelling

In [18]:
```
# loding the variable with Machine Learning Model
regressor = LogisticRegression()
```

In [19]:
```
# training the logistic regression model with training data
regressor.fit(xtrain,ytrain)
```

Out[19]:
```
LogisticRegression()
```

Next Step: Model Evaluation

In [20]:
```
# evaluating the model using accuracy score for training data
x_predict = regressor.predict(xtrain)
accuracy = accuracy_score(x_predict,ytrain)
```

In [21]:
```
# fetching the accuracy score of the model for training data
print("The Accuracy of the training data is:",accuracy)
```

```
The Accuracy of the training data is: 0.8342245989304813
```

The Accuracy of the model on the Training data is: 83.42%

In [22]:
```
# evaluating the model using accuracy score for test data
xtest_predict = regressor.predict(xtest)
accu = accuracy_score(xtest_predict,ytest)
```

In [23]:
```
# fetching the accuracy score of test data
print("The Accuracy score of test data is:",accu)
```

```
The Accuracy score of test data is: 0.7619047619047619
```

The Accuracy of the model on the Test Data is: 76.19%

Next Step: Model Deployment

Creating the Predicting System to predict the Rock or Mines using Sonar Data

In [24]:
```
input_data = (0.0261,0.0266,0.0223,0.0749,0.1364,0.1513,0.1316,0.1654,0.1864,0.26
# changing the input data to numpy array
new_input_data = np.asarray(input_data)

# reshaping the array for prediction of 1 instance
new_input_data1 = new_input_data.reshape(1,-1)

#stroring the value of prediction
predicted_value = regressor.predict(new_input_data1)
print(predicted_value)

if(predicted_value=='R'):
    print("The obeject is a Rock")
else:
    print("The Obeject is a Mine")
```

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