# ARTIFICIAL INTELIGENCE & MACHINE LEARNING

## SUBJECT CODE: CS3491

## MINI PROJECT

## SONAR ROCK VS MINE PREDICTION

**INTRODUCTION:**

Underwater mines or naval mines are self-contained explosive devices placed in water to destroy enemies' surface ships or submarines. Underwater mines are used since the mid-19th Century. Sea mines were introduced by David Bushner in 1977 during the American civil war. Previously mines were only activated by physical contact but the newly created mines can be activated by various methods. Modern mines can be activated by acoustic, pressure, and magnetic changes in the water which provoke them to explode. These are called influence mines.

Generally, underwater mines are classified as offensive or defensive warfare.

1) Offensive mines are strewn across hostile shipping lanes in order to damage merchant ships and military boats.

2) Defensive mines are placed along coastlines to divert enemy submarines and ships away from critical locations and into more heavily guarded places. Usually, Mines are mistaken as rocks during their identification, as mines can have the same shape, length, and width as rocks. .

**ABSTRACT:**

The Naval Defence System's use of underwater mines offers excellent security but also poses a risk to marine life and submarine vessels because the mines can be mistaken for rocks. As a result, we require a system that can predict objects far more accurately because doing so could be quite risky. More accurate data is required in order to produce outcomes with a high degree of precision. In our proposal, we propose a technique for sonar signals-based underwater mine and rock prediction. At 60 different angles, sonar signals are employed to record the various frequencies of submerged objects. Based on their accuracy, we created three binary classifier models. The mine and rock categories are then predicted using prediction models.

## METHODOLOGY:

The approach we suggest uses a prediction system to provide precise results and outcomes. The dataset was taken from "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" by R. Paul Gorman and Terrence J. Sejnowski. They used SONAR to conduct trails in a mine-stimulated area.

Sonar signals were fired at the mines from 60 different angles, and the results were recorded. The dataset is then trained using the models that were assessed. The predictive system receives the frequencies from the sonar as input. To determine whether an object is a rock or a mine, we apply classification machine learning techniques.

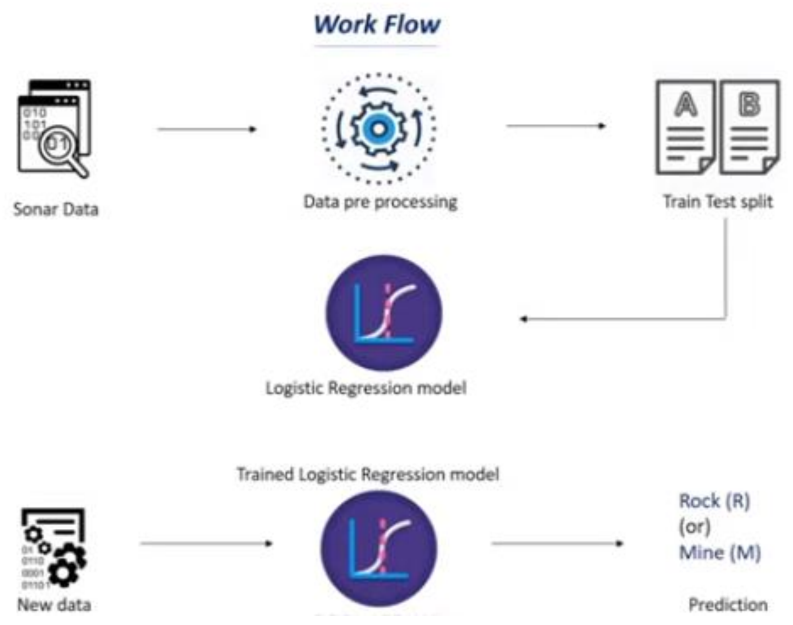


Diagram: Work Flow of the SONAR Rock VS Mine Prediction

## UNDERWATER MINES:

Naval mines, often known as underwater mines, are self-contained explosive weapons used to sink enemy surface vessels or submarines. Since the mid-19th century, underwater mines have been in use. During the American Civil War in 1977, David Bushner invented sea mines. The Adriatic Sea is thought to still contain 5,000 naval mines from the two world wars. Prior to their creation, mines could only be triggered by physical touch, but the new mines can be triggered in a variety of ways. Modern mines can be set off by changes in the water's acoustics, pressure, or magnetic field, which cause them to detonate. We refer to these as influence mines.

Underwater mines can be used in either offensive or defensive warfare. Mines are laid across hostile sea lanes to harm military and commercial vessels. In order to direct enemy submarines and ships away from strategic spots and towards areas that are better secured, defensive mines are positioned along coasts. Since mines can resemble rocks in size, form, and width, they are frequently mistaken for rocks when being identified. It is preferable to use a more exact input to get an accurate output in order to avoid this misconception. Sonar is one technique for finding mines.

## SONAR:

In order to steer and find objects, the sound navigation and ranging system uses sound waves. Acoustic mine detection, which falls under military applications, is the main use of SONAR. Other non-military uses include identifying sea divers, locating fish, and mapping the ocean floor. The frequency employed for a particular underwater sonar application (or range) is constrained by the sound wave attenuation, which increases rapidly with frequency and limits the distance that can be reached. The range of underwater SONARs used for mine detection is between 1 and 0.1 km, and their frequencies range from 0.1 to 1 MHz

Since infrasonic waves cannot travel under water and have long wavelengths, sonar favors ultrasonic waves because they can carry more energy. Sonar comes in active and passive forms. Passive SONAR, also known as listening SONAR, is only used to identify noises.

## WORKING PROCEDURE:

### Algorithm:

Step 1: We gather the dataset and perform data preparation and exploratory data analysis to clean the dataset.

Step 2: We split the data into training and test datasets. Using them, we evaluate the classification models.

Step 3: Following the evaluation, the top three performing models are determined to be KNN, SVM, and logistic regression.

Step 4: The accuracy of these models is evaluated, and a classification report is generated.
Step 5: We now fit the models to create a prediction system that's both accurate and efficient.

Step 6: Using the predictive systems, we can finally determine if the object is a mine or a rock.

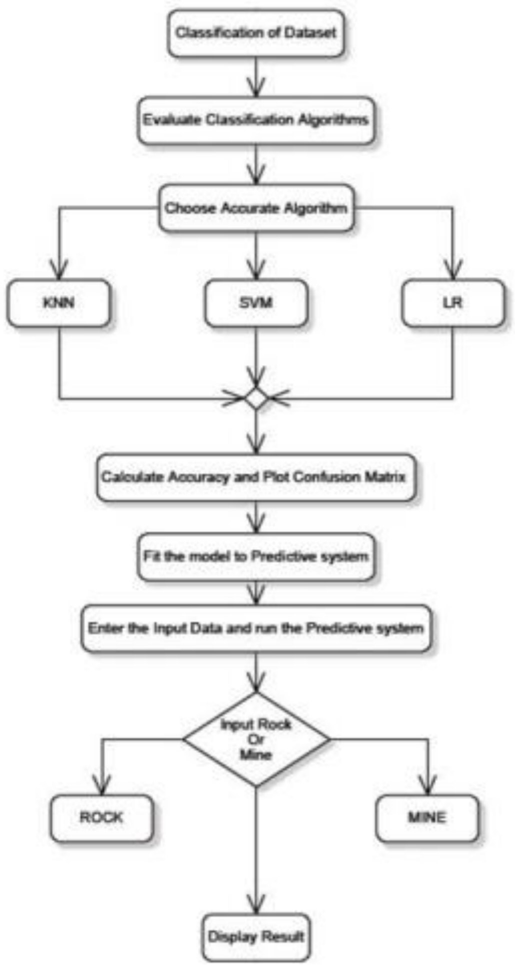

Classification of Dataset
Evaluate Classification Algorithms
Choose Accurate Algorithm
KNN
SVM
LR
Calculate Accuracy and Plot Confusion Matrix
Fit the model to Predictive system
Enter the Input Data and run the Predictive system
Input Rock Or Mine
ROCK
MINE
Display Result

Diagram: Working Procedure for SONAR Rock VS Mine Prediction

## <u>**SONAR Rock vs. Mine dataset:**</u>

The dataset has been collected from UCI Repository. It has come across 61 features which define and differentiate Rocks and Mines and comprises of 209 samples. This data is used for training and testing purpose. The Last column in this dataset indicates that, whether it's a mine or a rock, which is useful in prediction.

**Note:** This dataset can be downloaded from: https://drive.google.com/file/d/1pQxtljlNVh0DHYg-Ye7dtpDTlFceHVfa/view

You can download the Iris.csv file from the above link.

To Check our project:
https://colab.research.google.com/drive/1WJeq7WQYXZpeqB6gn1Fdx_Z9Vm2TWkQ7

**CODE:**

For any analysis, we need to have data. So, at the outset, we shall import data. Importing of data starts by following lines of code.

```
# 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
```

## 1. Load data:

The first step is Data Collection,

**CODE:**

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

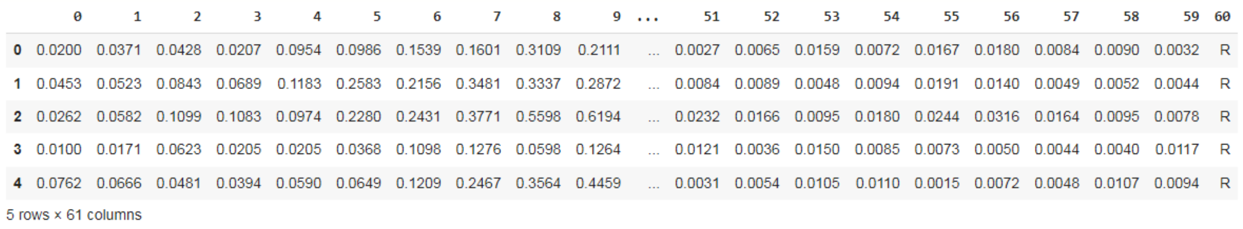
## 2. Print 5 Rows in the Dataset:

To check whether the data set is loaded or not print the first five rows by using the head method.

**CODE:**

```
sonar_data.head()
```

**OUTPUT:**



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 0.0072 | 0.0167 | 0.0180 | 0.0084 | 0.0090 | 0.0032 | R |
| **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 | 0.0094 | 0.0191 | 0.0140 | 0.0049 | 0.0052 | 0.0044 | R |
| **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 | 0.0180 | 0.0244 | 0.0316 | 0.0164 | 0.0095 | 0.0078 | R |
| **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 | 0.0085 | 0.0073 | 0.0050 | 0.0044 | 0.0040 | 0.0117 | R |
| **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 | 0.0110 | 0.0015 | 0.0072 | 0.0048 | 0.0107 | 0.0094 | R |

5 rows × 61 columns

## 3. Getting Information about the Dataset:

We will use the shape parameter to get the shape of the dataset.

**CODE:**

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

**OUTPUT:**

```
(208, 61)
```

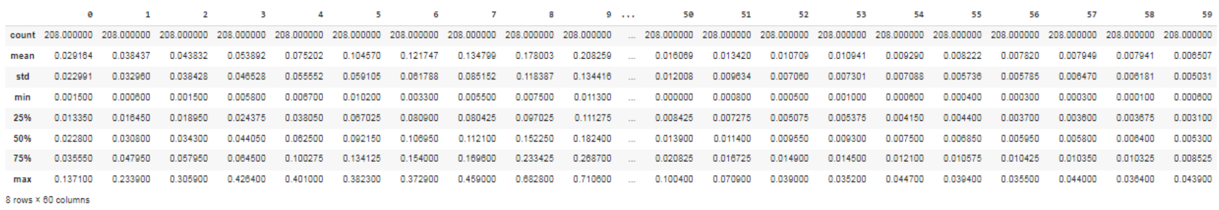
## 4. Statistical measures of the data:

Let’s get a quick statistical summary of the dataset using the describe () method. The Describe () function applies basic statistical computations on the dataset like extreme values, count of data points standard deviation, etc. Any missing value or NaN value is automatically skipped. describe () function gives a good picture of the distribution of data.

**CODE:**

```
sonar_data.describe()  #describe --> statistical measures of the data
```

**OUTPUT:**



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | ... | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 | 208.000000 |
| **mean** | 0.029164 | 0.038437 | 0.043832 | 0.053892 | 0.075202 | 0.104570 | 0.121747 | 0.134799 | 0.178003 | 0.208259 | ... | 0.016069 | 0.013420 | 0.010709 | 0.010941 | 0.009290 | 0.008222 | 0.007820 | 0.007949 | 0.007941 | 0.006507 |
| **std** | 0.022991 | 0.032960 | 0.038428 | 0.046528 | 0.055552 | 0.059105 | 0.061788 | 0.085152 | 0.118387 | 0.134416 | ... | 0.012008 | 0.009634 | 0.007060 | 0.007301 | 0.007088 | 0.005736 | 0.005785 | 0.006470 | 0.006181 | 0.005031 |
| **min** | 0.001500 | 0.000600 | 0.001500 | 0.005800 | 0.006700 | 0.010200 | 0.003300 | 0.005500 | 0.007500 | 0.011300 | ... | 0.000000 | 0.000800 | 0.000500 | 0.001000 | 0.000600 | 0.000400 | 0.000300 | 0.000300 | 0.000100 | 0.000600 |
| **25%** | 0.013350 | 0.016450 | 0.018950 | 0.024375 | 0.038050 | 0.067025 | 0.080900 | 0.080425 | 0.097025 | 0.111275 | ... | 0.008425 | 0.007275 | 0.005075 | 0.005375 | 0.004150 | 0.004400 | 0.003700 | 0.003600 | 0.003675 | 0.003100 |
| **50%** | 0.022800 | 0.030800 | 0.034300 | 0.044050 | 0.062500 | 0.092150 | 0.106950 | 0.112100 | 0.152250 | 0.182400 | ... | 0.013900 | 0.011400 | 0.009550 | 0.009300 | 0.007500 | 0.006850 | 0.005950 | 0.005800 | 0.006400 | 0.005300 |
| **75%** | 0.035550 | 0.047950 | 0.057950 | 0.064500 | 0.100275 | 0.134125 | 0.154000 | 0.169600 | 0.233425 | 0.268700 | ... | 0.020825 | 0.016725 | 0.014900 | 0.014500 | 0.012100 | 0.010575 | 0.010425 | 0.010350 | 0.010325 | 0.008525 |
| **max** | 0.137100 | 0.233900 | 0.305900 | 0.426400 | 0.401000 | 0.382300 | 0.372900 | 0.459000 | 0.682800 | 0.710600 | ... | 0.100400 | 0.070900 | 0.039000 | 0.035200 | 0.044700 | 0.039400 | 0.035500 | 0.044000 | 0.036400 | 0.043900 |

8 rows × 60 columns

## 5. Check How Many Rocks And Mines:

To check how many rocks and mines are present in the given dataset.

**CODE:**

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

**OUTPUT:**

```
M 111

R 97

Name: 60, dtype: int64
```

There are a total of 111 mines and 97 rocks in the given dataset.
Here m means mine and r means rock.

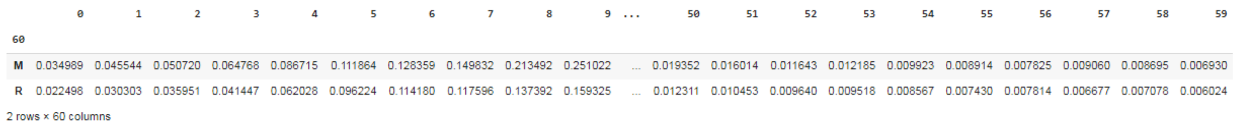
## 6. Predict Whether It Is Rock or Mine:

To check the mean of the mine and the rock because, with that difference of the values in each column, we are going to predict whether it is rock or mine in the ocean.

**CODE:**

```
sonar_data.groupby(60).mean()
```

**OUTPUT:**



| 60 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M | 0.034989 | 0.045544 | 0.050720 | 0.064768 | 0.086715 | 0.111864 | 0.128359 | 0.149832 | 0.213492 | 0.251022 | ... | 0.019352 | 0.016014 | 0.011643 | 0.012185 | 0.009923 | 0.008914 | 0.007825 | 0.009060 | 0.008695 | 0.006930 |
| R | 0.022498 | 0.030303 | 0.035951 | 0.041447 | 0.062028 | 0.096224 | 0.114180 | 0.117596 | 0.137392 | 0.159325 | ... | 0.012311 | 0.010453 | 0.009640 | 0.009518 | 0.008567 | 0.007430 | 0.007814 | 0.006677 | 0.007078 | 0.006024 |

2 rows × 60 columns

The mean value for the mine is 0.3 and rock is 0.02 the value is quite different with the help of these values we are going to predict whether it is rock or mine.

## 7. Split the Train And Test Data:

**CODE:**

```
# separating data and Labels
X = sonar_data.drop(columns=60, axis=1)
Y = sonar_data[60]

#Training and Test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0
.1,                                              stratify=Y, random_st
ate=1)
print(X.shape, X_train.shape, X_test.shape)
```

**OUTPUT:**

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

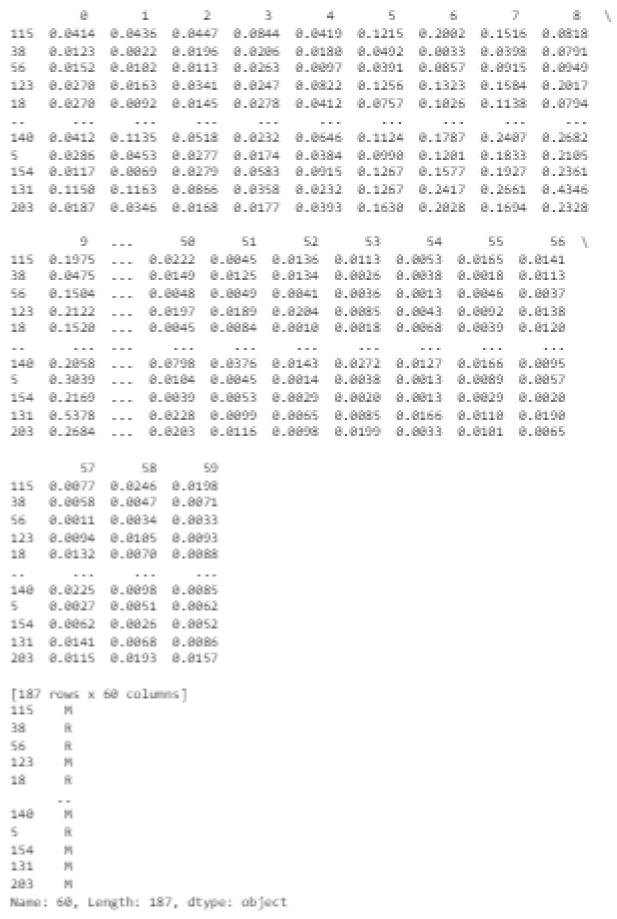
## 8. Print the Labels:

Now, let’s check the train split and the test split of the data.

**CODE:**

```
print(X_train)
print(Y_train)
```

**OUTPUT:**



```
         0       1       2       3       4       5       6       7       8  \
115  0.0414  0.0436  0.0447  0.0844  0.0419  0.1215  0.2002  0.1516  0.0818
38   0.0123  0.0022  0.0196  0.0206  0.0180  0.0492  0.0033  0.0398  0.0791
56   0.0152  0.0102  0.0113  0.0263  0.0097  0.0391  0.0857  0.0915  0.0949
123  0.0270  0.0163  0.0341  0.0247  0.0822  0.1256  0.1323  0.1584  0.2017
18   0.0270  0.0092  0.0145  0.0278  0.0412  0.0757  0.1026  0.1138  0.0794
..      ...     ...     ...     ...     ...     ...     ...     ...     ...
140  0.0412  0.1135  0.0518  0.0232  0.0646  0.1124  0.1787  0.2407  0.2682
5    0.0286  0.0453  0.0277  0.0174  0.0384  0.0990  0.1201  0.1833  0.2105
154  0.0117  0.0069  0.0279  0.0583  0.0915  0.1267  0.1577  0.1927  0.2361
131  0.1150  0.1163  0.0866  0.0358  0.0232  0.1267  0.2417  0.2661  0.4346
203  0.0187  0.0346  0.0168  0.0177  0.0393  0.1630  0.2028  0.1694  0.2328

         9  ...      50      51      52      53      54      55      56  \
115  0.1975  ...  0.0222  0.0045  0.0136  0.0113  0.0053  0.0165  0.0141
38   0.0475  ...  0.0149  0.0125  0.0134  0.0026  0.0038  0.0018  0.0113
56   0.1504  ...  0.0048  0.0049  0.0041  0.0036  0.0013  0.0046  0.0037
123  0.2122  ...  0.0197  0.0189  0.0204  0.0085  0.0043  0.0092  0.0138
18   0.1520  ...  0.0045  0.0084  0.0010  0.0018  0.0068  0.0039  0.0120
..      ...  ...     ...     ...     ...     ...     ...     ...     ...
140  0.2058  ...  0.0798  0.0376  0.0143  0.0272  0.0127  0.0166  0.0095
5    0.3039  ...  0.0104  0.0045  0.0014  0.0038  0.0013  0.0089  0.0057
154  0.2169  ...  0.0039  0.0053  0.0029  0.0020  0.0013  0.0029  0.0020
131  0.5378  ...  0.0228  0.0099  0.0065  0.0085  0.0166  0.0110  0.0190
203  0.2684  ...  0.0203  0.0116  0.0098  0.0199  0.0033  0.0101  0.0065

         57      58      59
115  0.0077  0.0246  0.0198
38   0.0058  0.0047  0.0071
56   0.0011  0.0034  0.0033
123  0.0094  0.0105  0.0093
18   0.0132  0.0070  0.0088
..      ...     ...     ...
140  0.0225  0.0098  0.0085
5    0.0027  0.0051  0.0062
154  0.0062  0.0026  0.0052
131  0.0141  0.0068  0.0086
203  0.0115  0.0193  0.0157

[187 rows x 60 columns]
115    M
38     R
56     R
123    M
18     R
      ..
140    M
5      R
154    M
131    M
203    M
Name: 60, Length: 187, dtype: object
```

**CODE:**

```
#Model Training --> Logistic Regression
model = LogisticRegression()
#training the Logistic Regression model with training data
model.fit(X_train, Y_train)
LogisticRegression(C=1.0, class_weight=None, dual=False,
                   fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001,
                   verbose=0,
                   warm_start=False)
```

**9. Model Training:**

Train the data from the given dataset.

**CODE:**

```
#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)
```

**OUTPUT:**

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

**10. Model Testing:**

Test the data to conclude the result.

**CODE:**

```
#accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy on test data : ', test_data_accuracy)
```

**OUTPUT:**

```
Accuracy on test data :  0.7619047619047619
```

**11. Making a Predictive System:**

Instead of predicting the entire dataset, we can check for only one row of the data. It is very useful,

**CODE:**

```
input_data = (0.0307,0.0523,0.0653,0.0521,0.0611,0.0577,0.0665,0.0664,0
.1460,0.2792,0.3877,0.4992,0.4981,0.4972,0.5607,0.7339,0.8230,0.9173,0.
9975,0.9911,0.8240,0.6498,0.5980,0.4862,0.3150,0.1543,0.0989,0.0284,0.1
008,0.2636,0.2694,0.2930,0.2925,0.3998,0.3660,0.3172,0.4609,0.4374,0.18
20,0.3376,0.6202,0.4448,0.1863,0.1420,0.0589,0.0576,0.0672,0.0269,0.024
5,0.0190,0.0063,0.0321,0.0189,0.0137,0.0277,0.0152,0.0052,0.0121,0.0124
,0.0055)

# changing the input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the np array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)
```

```
if (prediction[0]=='R'):
  print('The object is a Rock')
else:
  print('The object is a mine')
```

**OUTPUT:**

```
['M']
The object is a mine
```

For another row,

**CODE:**

```
input_data = (0.0260,0.0363,0.0136,0.0272,0.0214,0.0338,0.0655,0.1400,0
.1843,0.2354,0.2720,0.2442,0.1665,0.0336,0.1302,0.1708,0.2177,0.3175,0.
3714,0.4552,0.5700,0.7397,0.8062,0.8837,0.9432,1.0000,0.9375,0.7603,0.7
123,0.8358,0.7622,0.4567,0.1715,0.1549,0.1641,0.1869,0.2655,0.1713,0.09
59,0.0768,0.0847,0.2076,0.2505,0.1862,0.1439,0.1470,0.0991,0.0041,0.015
4,0.0116,0.0181,0.0146,0.0129,0.0047,0.0039,0.0061,0.0040,0.0036,0.0061
,0.0115)

# changing the input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the np array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]=='R'):
  print('The object is a Rock')
else:
  print('The object is a mine')
```

**OUTPUT:**

['M']

The object is a mine

## **CONCLUSION:**

Our project “Underwater mine and rock prediction by evaluation of machine learning algorithms” are used to detect rocks and mines in the ocean bed. Naval mines are an effective method for blocking ships and restricting naval operations which result the significant negative economic and environment impacts. There are two existing ways to detect a mine, one by using sonar signals and the other by using manpower. Using Sonar signals has been better option as the risk for the letter is more. The data is collected stored in a CSV file. By using different machine learning techniques we can observe and understand the nature of the predictive system. By the evaluation of algorithms, we get to check and compare the accuracies to build a better performing prediction model. A python in open-source software and the machine computation is also faster many others and the cost might decrease dependently .Through this project, we want to make the process a bit easy and simple to achieve and use.