SONAR ROCK OR MINE DETECTION

A Course Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

by

MD NABEEL AHMED	**(2103A51058)**
K. NIKHIL	**(2103A51130)**
G. VIGNESH	**(2103A51353)**

Under the guidance of

Mr. M THARUN REDDY
Assistant Professor,department of CSE.

Department of Computer Science and Artificial Intelligence

Department of Computer Science and Artificial Intelligence

CERTIFICATE

This is to certify that the project entitled **"SONAR ROCKS OR MINE DETECTION"** is the bonafied work carried out by **MD NABEEL AHMED, K. NIKHIL, G. VIGNESH** as a Course Project for the partial fulfilment to award the degree **BACHELOR OF TECHNOLOGY in ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** during the academic year 2022-2023 under our guidance and Supervision.

Mr. M THARUN REDDY

Asst. Professor,

S R University,

Ananthasagar, Warangal.

Dr. M.Sheshikala

Assoc. Prof. & HOD (CSE),

S R University,

Ananthasagar, Warangal.

ACKNOWLEDGEMENT

We express our thanks to Course co-coordinator **Mr. M. THARUN REDDY**, Asst. Prof. for guiding us from the beginning through the end of the Course Project. We express our gratitude to Head of the department CS&AI, **Dr. M.Sheshikala, Associate Professor** for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, School of Computer Science and Artificial Intelligence, **Dr C. V. Guru Rao**, for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

As we all know that our lives are changing drastically due to the evolution of Artificial Intelligence, We as a team of two decided to develop a machine which simulates human intelligence as a course project in the AIML course. Our major concern is to improve the technology in the food sector.

Our project helps the public The artificial intelligence and machine learning (AIML) classification algorithms are used in this research to suggest a sonar-based system for the identification of rocks or mines. The device uses sonar technology to gather acoustic signals from the underwater environment and then uses signal processing techniques to extract characteristics from the sounds. To detect the presence of rocks or mines in the water, these characteristics are then input into a variety of classification algorithms, such as k-Nearest Neighbours, Decision Trees, Support Vector Machines. The accuracy, sensitivity, and specificity of each algorithm's performance are assessed, and the algorithm with the best performance is chosen for the final implementation. The suggested technology might considerably increase the effectiveness and efficiency of rock and mine identification in underwater settings, improving the safety of maritime operations.

After our successful completion of gathering data. We have developed a model which is able to display the rating of calorie content. We have successfully completed the project by using the knowledge I have gained while learning the AIML course.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview:

This project is mainly a case study which aims to use artificial intelligence and machine learning techniques to detect the underwater mines present in the ocean using sonar technology and gives us the accurate result. The main aim of this project is to enhance the capabilities of naval forces in detecting underwater mines, which pose a significant threat to maritime security and can cause catastrophic damage to ships and submarines.

Additionally, the project involves the development of a system that uses sonar sensors to collect data about the underwater environment. This data is then processed using AI and ML algorithms to identify and classify underwater objects as either rocks or mines. The system is trained on a large dataset of sonar images to improve its accuracy in detecting and classifying objects.

1.2 Problem Statement:

To develop an artificial intelligence and machine learning based predictive model that can compare the performance of different machine learning models for the detection of underwater mines and non-mines using numerical datasets obtained from sonar images.

1.3 Existing Systems:

There are several existing systems and platforms that provide SONAR Mine or Rock detection but all of those are done using traditional methods. The predicted values are not of a high accuracy. The data rows are imbalanced, that is inconsistent in length.

However, these existing systems have limitations, including a lack of comprehensive data analysis and prediction models that accurately detect the kind of object with provided SONAR data. Therefore, this project aims to develop a comparision between comprehensive and accurate predictive models using machine learning techniques to provide valuable insights into the potential of machine learning algorithms for the detection of underwater mines, which has significant implications for maritime security and defence.

1.4 Proposed System:

The proposed system will focus on analysing and predicting the number of the mines or rocks using sonar. When we use sonar the waves which are sent from the ship are hit to ground and if there are any rocks or mines it will send back the frequencies which are stored as datasets.

These datasets are used by the system which will use 4 different types of classifications, Logistic Regression, KNN (K-Nearest Neighbour), SVM (Support Vector Machine) and Decision trees model to develop predictive models for different regions, and various dependent variables will be used to improve the accuracy of the predictions.

To better understand the data, the system will use various types of graphs, including line plots, box plots and bar plots. These plots will help visualise the data and identify any trends, patterns, or outliers.

Overall, the proposed system aims to provide accurate predictions for the detection of underwater mines. These insights have significant implications for maritime security and defence.

1.5 Objectives:

The main objectives of this project are to:

1. Visualise the data to see meaningful correlations.
2. Develop a predictive model by using different Machine learning algorithms.
3. Using this predictive model and data to find the accuracy which will help in detection of underwater mines.
4. To provide valuable insights into the potential of machine learning algorithms for the detection of underwater mines, which has significant implications for maritime security and defence.

1.6 Architecture

2. LITERATURE SURVEY

2.1 Survey Documentation

In this section, we documented the literary survey conducted related to the problem statement of detecting and analysing the mine or rock by sonar using artificial intelligence and machine learning techniques.We conducted a comprehensive research of various databases, including Google scholar and ScienceDirect. There are many articles and research papers available on the topic of sonar mines vs rocks and the history of the sonar mine or rocks ML project. Here are a few resources that may be useful:

[1]"Using a Support Vector Machine to Classify Sonar Returns from Underwater Mines and Rocks" by Stephen D. Brown and Robert M. Haralick. This research paper published in 2000 describes the use of support vector machines (SVM) to classify sonar returns from underwater mines and rocks.

[2]"Mines vs. Rocks Revisited: A Modern Perspective on a Classic Dataset" by Jesse Davis and Mark Goadrich. This paper published in 2006 presents a modern perspective on the classic sonar mines vs. rocks dataset, comparing the performance of various ML algorithms, including KNN and logistic regression.

[3]"Performance Analysis of Sonar Data Classification using Machine Learning Techniques" by M. Ramesh and P. Balakrishnan. This research paper published in 2017 compares the performance of various ML algorithms, including KNN, decision tree, SVM, and neural networks, for sonar data classification.

[4]"Sonar Mines vs Rocks Dataset: A Brief Analysis" by M. Arif Wani and M. Muzamil Bhat. This paper published in 2019 provides a brief analysis of the sonar mines vs rocks dataset, including a comparison of the performance of KNN and logistic regression.

[5]"A Survey of Sonar-Based Underwater Object Detection and Classification Techniques" by Arun Solanki and Abhijit Mitra. This survey paper published in 2021

provides an overview of various techniques for underwater object detection and classification, including sonar-based approaches.

[6]These resources provide a glimpse into the history of the sonar mines vs. rocks ML project and the various techniques that have been used to classify sonar data. They also highlight the ongoing research in this area and the challenges that remain in developing accurate and reliable ML models for underwater object detection and classification.

[7]Mines versus rocks revisited: Distributed representations and deep networks" by Ryan Spring and Pedro Domingos. This paper published in 2016 investigates the use of deep neural networks for the sonar mines vs. rocks classification task.

[8]"Sonar data classification using deep learning: A review" by Sushma Mishra and S. S. Limaye. This review paper published in 2020 provides a comprehensive survey of deep learning approaches for sonar data classification, including the sonar mines vs. rocks dataset.

[9]"An Investigation of Machine Learning Algorithms for Sonar-based Mine Detection" by Alexander J. Smola, Robert M. Haralick, and Mark E. Oxley. This research paper published in 1998 compares the performance of various machine learning algorithms, including KNN, decision trees, neural networks, and SVMs, for sonar-based mine detection.

[10]"A Comparative Study of Data Mining Techniques for Detection of Underwater Mines using Sonar Data" by Muhammad Salman Khan and Tae-Sun Chung. This research paper published in 2014 compares the performance of various data mining techniques, including KNN, decision trees, and SVMs, for the detection of underwater mines using sonar data.

[11]"Hybrid neural network for sonar target recognition" by J. R. Hodge and E. J. Bowles. This research paper published in 1997 proposes a hybrid neural network approach for sonar target recognition, which combines a self-organising map and a multi-layer perceptron.

3. DATA PRE-PROCESSING

3.1 Description of Dataset

The dataset in .csv format is obtained through SONAR(Sound Navigation and Ranging) techniques. Sonar (sound navigation and ranging) is a technique based on the principle of reflection of ultrasonic sound waves. These waves propagate through water and reflect on hitting the ocean bed or any object obstructing its path.

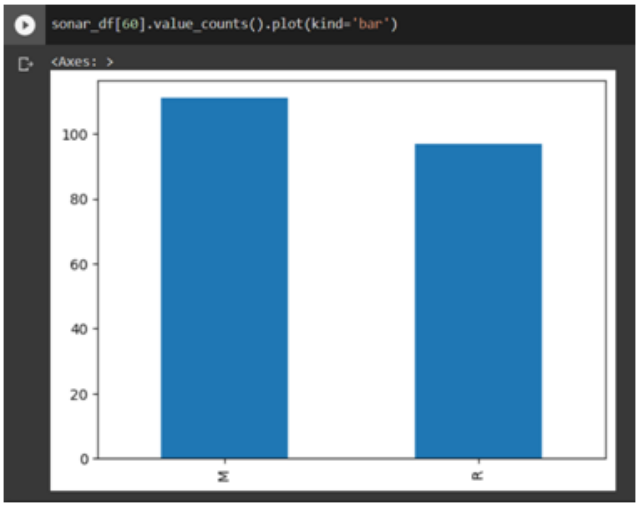
The CSV files contain data regarding sonar signals bounced off a metal cylinder (mines - M) and a roughly cylindrical rock (rock - R) at various angles and under various conditions.

There are a total of 60 features or attributes obtained through this technique which are the frequencies of the bounced back sonar signal. The dataset contains 208 rows of data items each associated with the classifier attribute containing two distinct classes M - mine and R - rock.

```
np.shape(sonar_df)
(208, 61)
```

```
np.shape(sonar_df)
(208, 61)
```

```
sonar_df[60].value_counts().plot(kind='bar')
<Axes: >
```



3.2 Data Cleaning

The purpose of data cleaning is to prepare the dataset for analysis and modelling by identifying and correcting errors, inconsistencies, and inaccuracies. The data cleaning process involves several steps, including but not limited to:

1. Removing duplicate values: Duplicate values in the dataset can distort the results and lead to incorrect conclusions. Therefore, it is important to remove duplicate values before analysis.

```
sonar_df.duplicated()

0      False
1      False
2      False
3      False
4      False
       ...
203    False
204    False
205    False
206    False
207    False
Length: 208, dtype: bool
```

2. Handling missing or null values: Null or missing values can be due to various reasons such as data entry errors, system failure, or non-response. Null values can adversely affect the analysis and may result in incorrect conclusions. Therefore, it is important to remove them from the dataset or impute them with reasonable values.

```
sonar_df.isnull()
```

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57	58	59	60
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
203	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
204	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
205	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
206	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
207	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

208 rows × 61 columns

Since there are no duplicate or null values in the dataset, it is considered to be clean and hence can be proceeded for data visualisation.

3.3 Data Augmentation

The number of data rows associated with each classifier, that is with R (Rock) are 97 while the data rows associated with M (Mine) are 111.

```
sonar_df[60].value_counts()

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

Even though there is only a difference of only 14 samples, in comparison to the total number of data samples available, this difference is significant and needs to be balanced. In order to balance the dataset, there are two options:

1. upsampling - resample the values to make their count equal to the class label with the higher count (here, 111). Here, we will be upsampling. First, we divide the whole dataset into 2, one for each label.

```
df=sonar_df
df0 = df[df[60] == 'R']
df1 = df[df[60] == 'M']
```

2. downsampling - pick n samples from each class label where n = number of samples in class with least count (here, 97). The sample() function of the data frame is used to resample and obtain 9200 samples. The append() function of the data frame is used to combine the rows in both the datasets

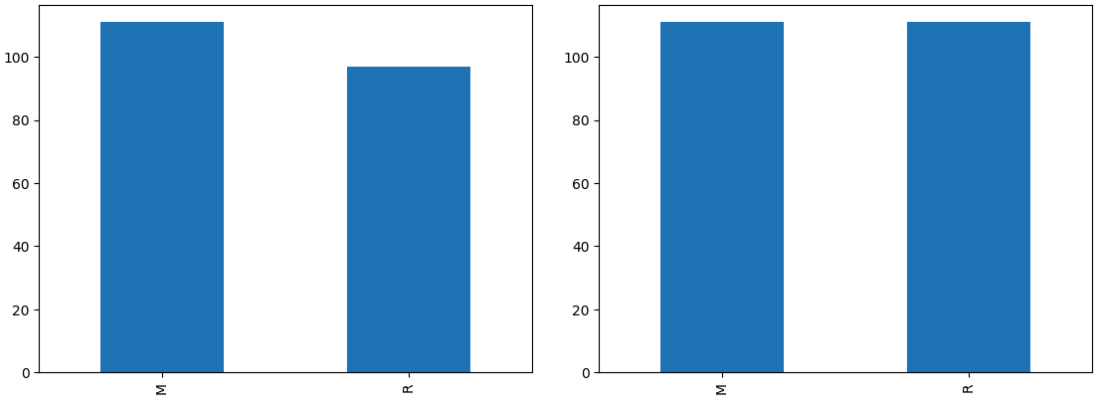
```
print("Number of samples in:")
print("Class label 0 - ", len(df0))
print("Class label 1 - ", len(df1))

df0 = df0.sample(len(df1), replace = True)

print('\nAfter resampling - ')
print("Number of samples in:")
print("Class label 0 - ", len(df0))
print("Class label 1 - ", len(df1))
```

```
Number of samples in:
Class label 0 -  97
Class label 1 -  111

After resampling -
Number of samples in:
Class label 0 -  111
Class label 1 -  111
```

```
[28] df = df1.append(df0)
     print('Total number of samples - ', len(df))

     Total number of samples -  222
```

3.4 Data Visualization

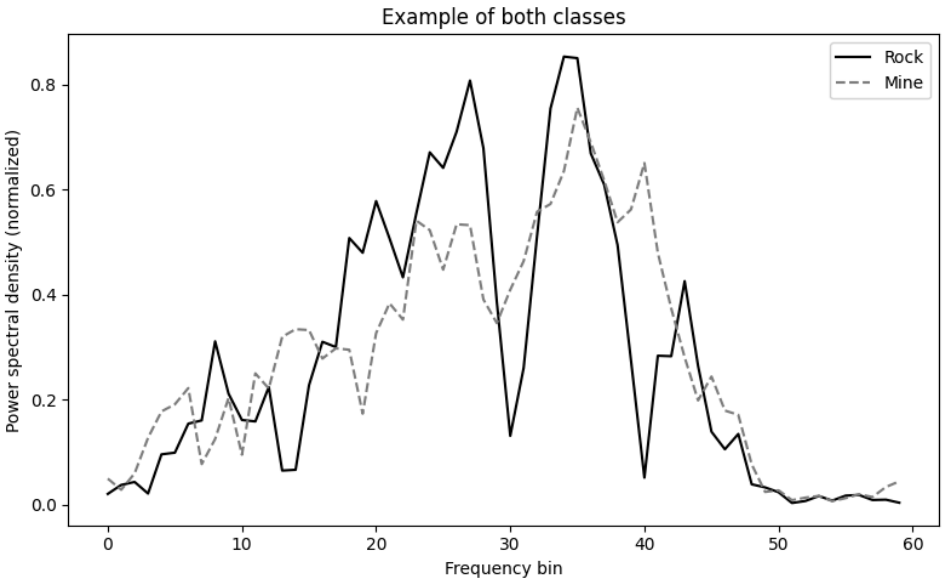
The comparison of graph before and after resampling the number of data rows associated with each classifier:



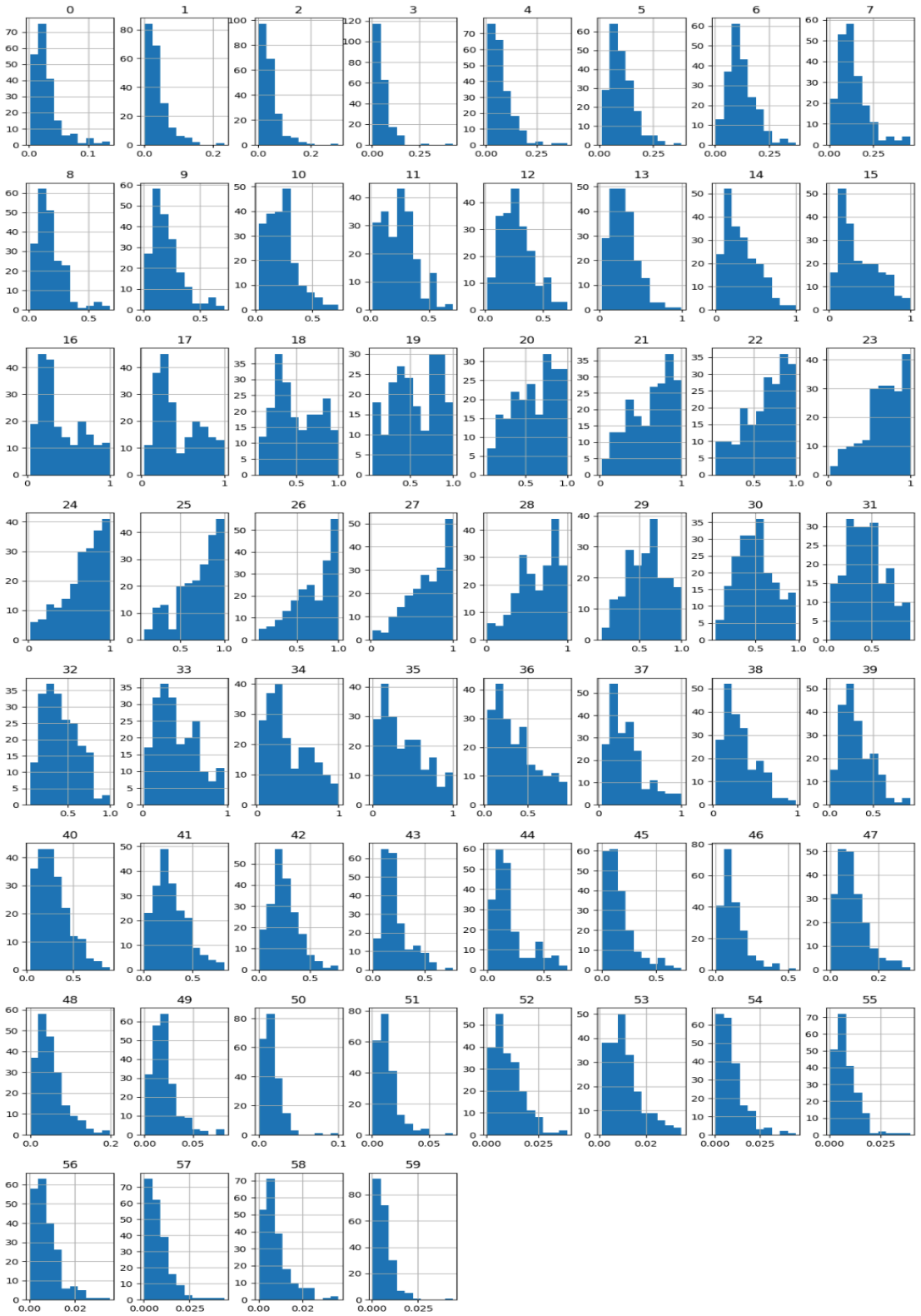
Graph(a) Before **Graph(b) After**

This is how an example spectral envelope looks for each of the classes:

```
plt.figure(figsize=(8,5))
plt.plot(df[df[60] == 'R'].values[0][:-1], label='Rock', color='black')
plt.plot(df[df[60] == 'M'].values[0][:-1], label='Mine', color='gray', linestyle='--')
plt.legend()
plt.title('Example of both classes')
plt.xlabel('Frequency bin')
plt.ylabel('Power spectral density (normalized)')
plt.tight_layout()
plt.show()
```



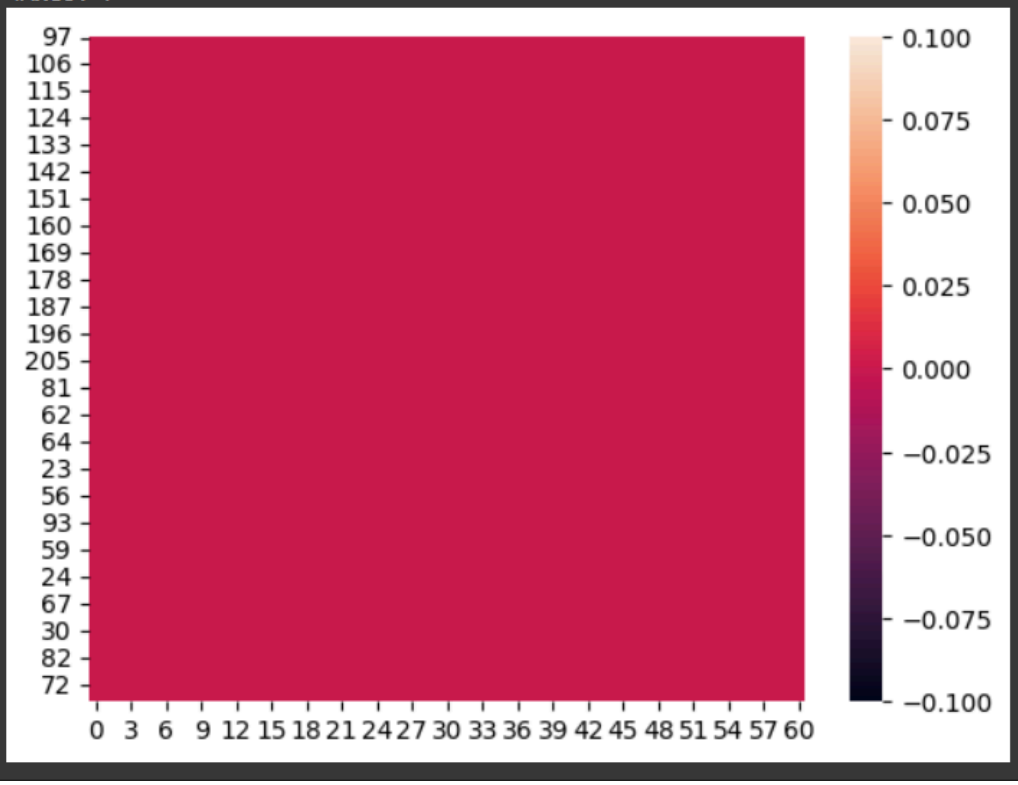
Histogram of all features:



Heat Maps of data values:

```
sns.heatmap(df.isna())
```

```
<Axes: >
```



4. METHODOLOGY

4.1 Procedure to solve the given problem.

Splitting the data

After data preprocessing and visualisation the dependent and independent data values must be split so that they can be fit in a machine learning model. In this case we split the first 60 features or attributes into the independent variable 'X' and the classifier attribute in the dependent variable 'Y'.

```
X = df.drop(columns=60,axis=1)
Y = df[60]
```

Training and testing the data

Now we train the data using the train_test_split module from sklearn. It splits the data into training and testing data variables which will later be used in fitting into models.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, random_state=42)
```

Using appropriate machine learning model

To get optimal precision and accuracy, we train the training data to different machine learning models. These models will train the data and produce an accuracy score using the testing data. The models to be used are:

1. Logistic Regression
2. K Nearest Neighbour
3. Decision Tree
4. Support Vector Machine

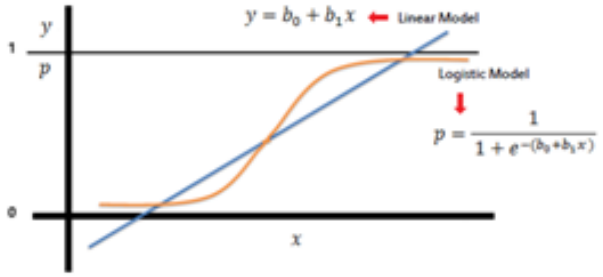
By comparing the accuracy score, confusion matrix and classification report we can deduce as to which model generates optimal accuracy.

1. Logistic Regression

Logistic Regression is a Machine Learning method that is used to solve classification issues. The classification algorithm Logistic Regression is used to predict the likelihood of a categorical dependent variable. The dependent variable in logistic regression is a binary variable with data coded as 1 (yes, True, normal, success, etc.) or 0 (no, False, abnormal, failure, etc.).

The logistic function in linear regression is a type of sigmoid, a class of functions with the same specific properties.

$$f(x) = \frac{1}{1 + e^{-x}}$$



To implement the logistic regression model we first import the LogisticRegression module from sklearn. Then the X_train and Y_train fit into the model.

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,Y_train)
```

```
LogisticRegression()
```

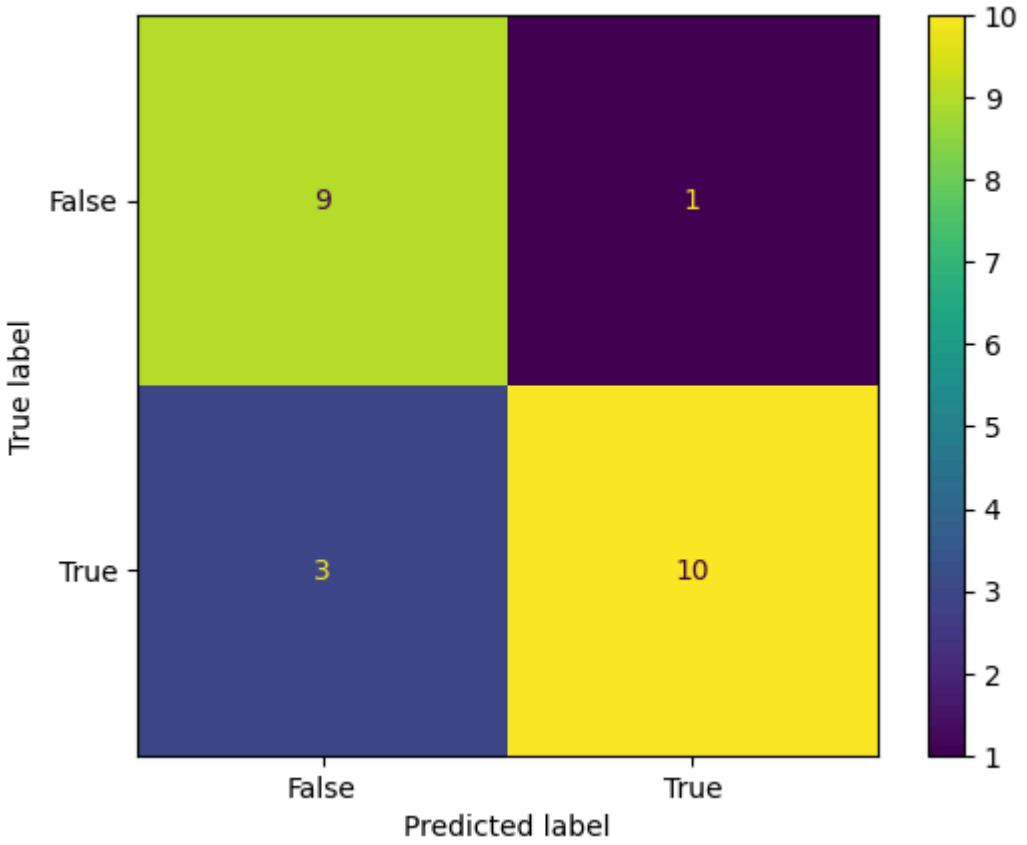
Now we obtain an accuracy score using the testing data to compare it with other models.

```
test_pred = lr.predict(X_test)
lra = accuracy_score(test_pred,Y_test)
print(lra)
```

```
0.8260869565217391
```

Also Using the testing data we obtain a confusion matrix and classification report of logistic regression model.

```
cm = metrics.confusion_matrix(test_pred,Y_test)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [False, True])
cm_display.plot()
plt.show()
```



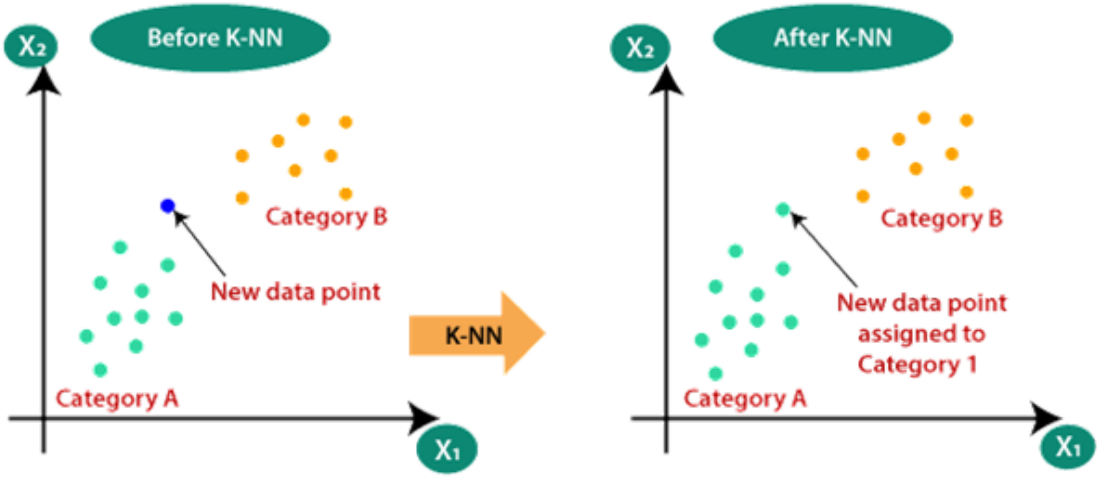

```
print(classification_report(test_pred, Y_test))
```

	precision	recall	f1-score	support
M	0.75	0.90	0.82	10
R	0.91	0.77	0.83	13
accuracy			0.83	23
macro avg	0.83	0.83	0.83	23
weighted avg	0.84	0.83	0.83	23

2. K Nearest Neighbour

K-nearest neighbours (KNN) is a type of supervised learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data.

The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and which class holds the highest probability will be selected.In the case of regression, the value is the mean of the 'K' selected training points.



Following the same methodology used for logistic regression, we first import KNearestNeighbot from the sklearn module. Then the training data X_train and Y_train fit into the model.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, Y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

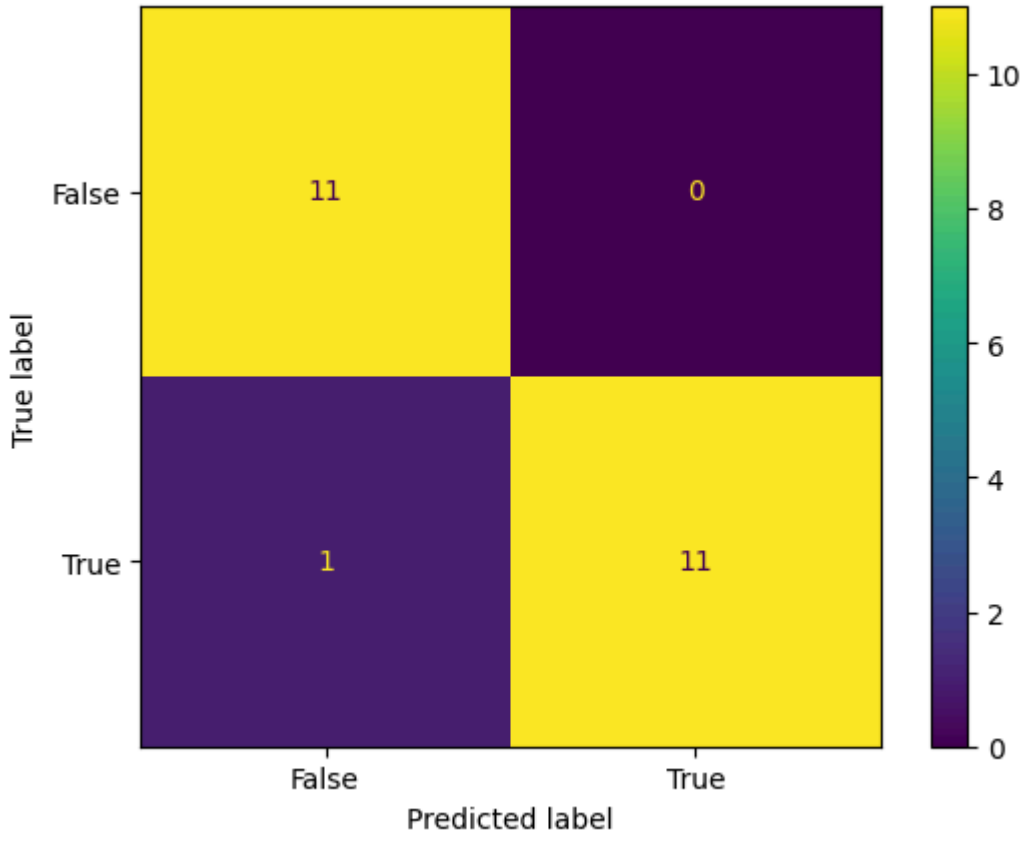
And now we predict the accuracy based on testing data by comparing it to trained data.

```
test_pred = knn.predict(X_test)
knna = accuracy_score(test_pred,Y_test)
print(knna)
```

```
0.9565217391304348
```

Again obtaining a confusion matrix and classification report of the model after training and testing phase.

```
cm = metrics.confusion_matrix(test_pred,Y_test)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [False, True])
cm_display.plot()
plt.show()
```




```
print(classification_report(test_pred, Y_test))
```

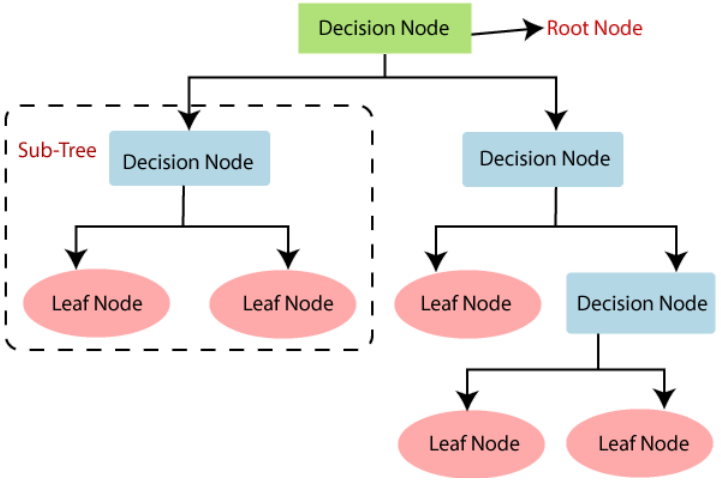
	precision	recall	f1-score	support
M	0.92	1.00	0.96	11
R	1.00	0.92	0.96	12
accuracy			0.96	23
macro avg	0.96	0.96	0.96	23
weighted avg	0.96	0.96	0.96	23

3. Decision tree

A decision tree algorithm is a type of machine learning algorithm used for classification and regression tasks. It works by creating a tree-like model of decisions and their possible consequences.

In a decision tree, each node represents a decision based on a feature or attribute of the data, and each branch represents a possible outcome of that decision. The leaf nodes of the tree represent the final classification or prediction.

Decision trees are relatively easy to interpret and visualize, and they can handle both categorical and numerical data. However, they can be sensitive to small variations in the data and prone to overfitting if the tree is too complex or the training data is noisy. Ensemble methods like random forests and gradient boosting can help to mitigate these issues.



Importing DecisionTreeClassifier from sklearn module and fitting the training data into it.

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion = 'entropy')
dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(criterion='entropy')
```

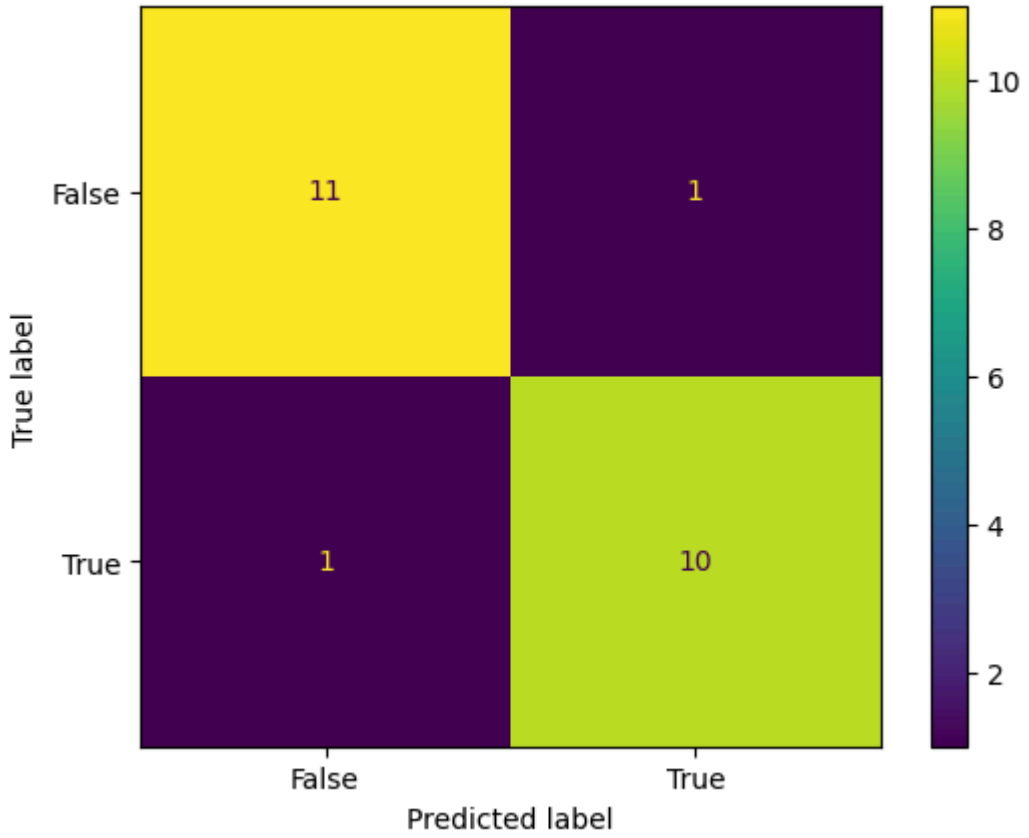
Obtaining the accuracy score using the testing data by comparing with trained data.

```
test_pred = dt.predict(X_test)
dta = accuracy_score(test_pred,Y_test)
print(dta)
```

```
0.8695652173913043
```

Finally Obtaining confusion matrix and classification report.

```
cm = metrics.confusion_matrix(test_pred, Y_test)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [False, True])
cm_display.plot()
plt.show()
```



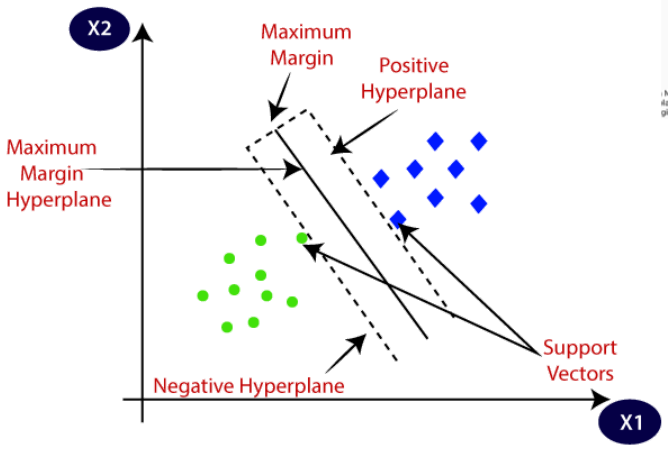

```
print(classification_report(test_pred, Y_test))
```

	precision	recall	f1-score	support
M	0.92	0.85	0.88	13
R	0.82	0.90	0.86	10
accuracy			0.87	23
macro avg	0.87	0.87	0.87	23
weighted avg	0.87	0.87	0.87	23

4. Support Vector Machine

Support Vector Machines (SVM) is a popular machine learning algorithm used for classification and regression analysis. It is based on the concept of finding a hyperplane that maximally separates the data points into different classes.

In an SVM model, the data is represented as points in a high-dimensional space. The algorithm then tries to find the hyperplane that best separates the data into different classes by maximising the margin between the hyperplane and the nearest data points. The data points that lie closest to the hyperplane are called support vectors.



Firstly importing SVC from sklearn and then fitting the trained data into the model.

```
from sklearn.svm import SVC
svm = SVC(kernel = 'rbf')
svm.fit(X_train,Y_train)
```

```
SVC()
```

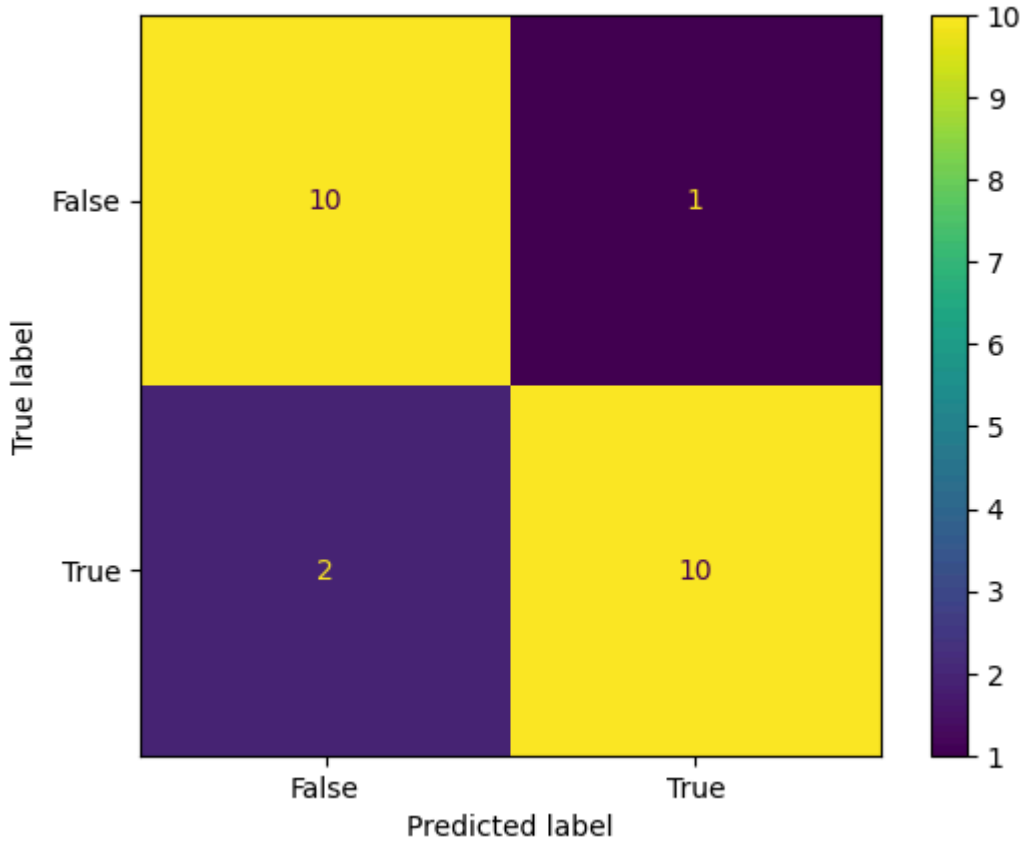
Now obtaining accuracy scores using the testing data by comparing with trained data.

```
test_pred = svm.predict(X_test)
svma = accuracy_score(test_pred,Y_test)
print(svma)
```

```
0.8695652173913043
```

Finally Obtaining confusion matrix and classification report.

```
cm = metrics.confusion_matrix(test_pred, Y_test)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [False, True])
cm_display.plot()
plt.show()
```




```
print(classification_report(test_pred, Y_test))
```

	precision	recall	f1-score	support
M	0.83	0.91	0.87	11
R	0.91	0.83	0.87	12
accuracy			0.87	23
macro avg	0.87	0.87	0.87	23
weighted avg	0.87	0.87	0.87	23

4.2 Model Architecture

Data collection: We have obtained SONAR data from reliable sources, such as the KAGGLE. This dataset was used in Gorman, R. P., and Sejnowski, T. J. (1988). "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75–89.

Data preprocessing: The data comes with no null values or duplicate values so there was no need to clean it. Although, it had an imbalance in the number of data values of the classifier attribute which was later fixed by sampling.

Train-test split: Split the data into training and testing sets using the train_test_split function from the sklearn library. We have chosen 90% of the data as train data and the remaining 10% for the test data.

Model selection and training: We have chosen different machine learning models and trained them. Then we obtained the accuracy scores and confusion matrices of those models to compare.

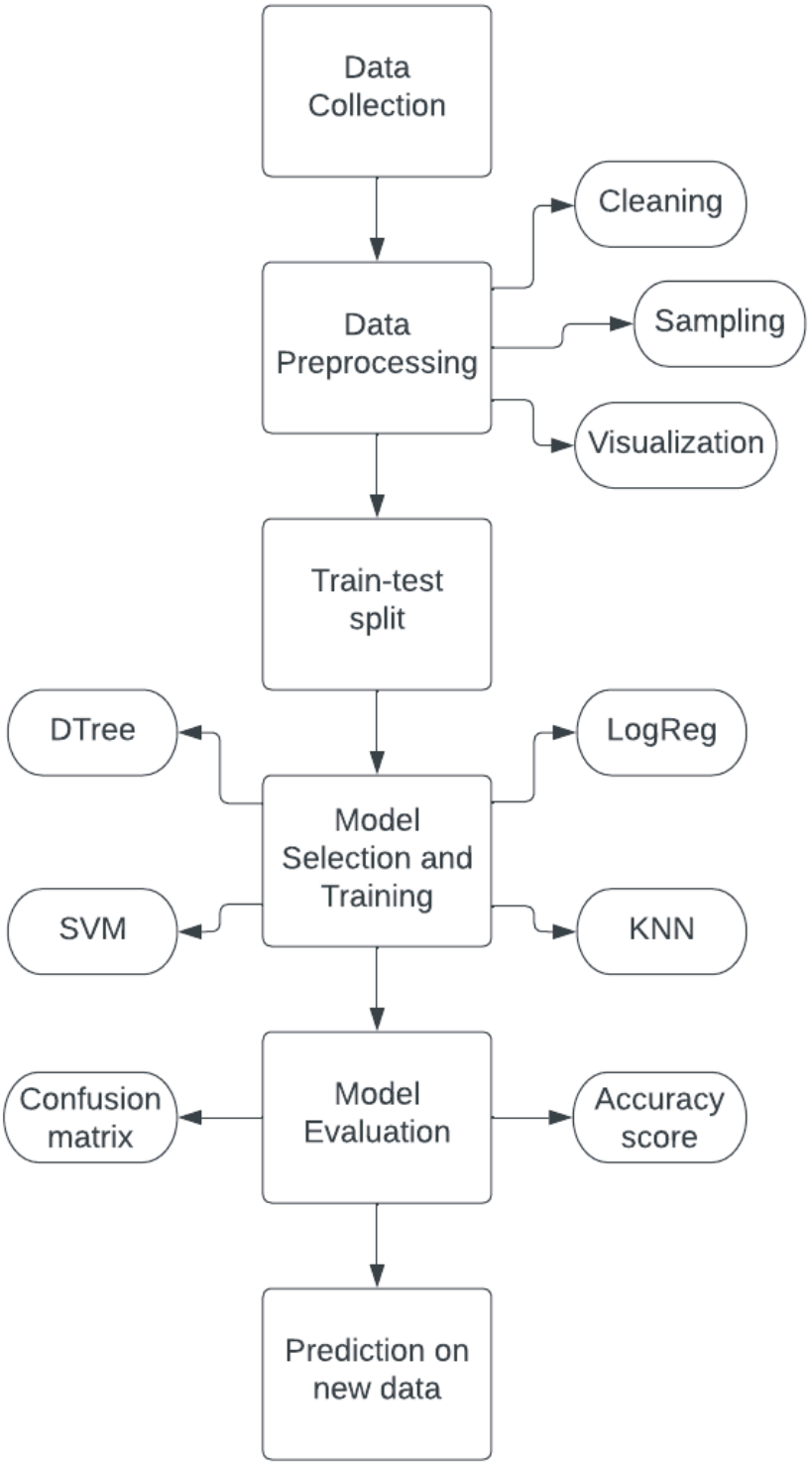
Model evaluation: We have evaluated the model's performance on the testing set by calculating various metrics.

Prediction on new data: We have tested the model on new data by making predictions using the predict method. We have created a new dataset of predicted variables and used the model to predict new cases.

```
input_data = (0.0201,0.0376,0.0438,0.0207,0.0954,0.0906,0.1539,0.1601,0.3109,
              0.2111,0.1609,0.1582,0.2238,0.0645,0.0660,0.2273,0.3100,0.2999,
              0.5078,0.4797,0.5783,0.5071,0.4328,0.5550,0.6711,0.6415,0.7104,
              0.8080,0.6791,0.3857,0.1307,0.2604,0.5121,0.7547,0.8537,0.8507,
              0.6692,0.6097,0.4943,0.2744,0.0510,0.2834,0.2825,0.4256,0.2641,
              0.1386,0.1051,0.1343,0.0383,0.0324,0.0232,0.0027,0.0065,0.0159,
              0.0072,0.0167,0.0180,0.0084,0.0090,0.0032)
input_data_np_array = np.asarray(input_data)
reshaped_input = input_data_np_array.reshape(1,-1)
prediction = model.predict(reshaped_input)

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

```
The object is a Mine
```

The model architecture can also be represented with the help of a block diagram.



4.3 Software description

This project is developed using Jupyter Notebook, which is a popular web-based interactive development environment for creating and sharing data science projects. The code is written in Python programming language and uses several Python libraries, including Pandas, NumPy, Scikit-learn, Matplotlib, and Seaborn.

Pandas is a library that is used for data manipulation and analysis. It provides a set of data structures and functions to work with structured data, such as data frames and series. In this project, Pandas is used to load, clean, and manipulate the COVID-19 dataset.

NumPy is a library that is used for numerical computing with Python. It provides a powerful array processing capability and mathematical functions to work with large, multi-dimensional arrays and matrices. In this project, NumPy is used to perform numerical operations, such as calculating mean and standard deviation of the data.

Scikit-learn is a library that is used for machine learning tasks, such as building and evaluating models. It provides a wide range of tools for supervised and unsupervised learning, including regression, classification, clustering, and dimensionality reduction. In this project, Scikit-learn is used to build a linear regression model to predict the number of new COVID-19 cases based on several input variables.

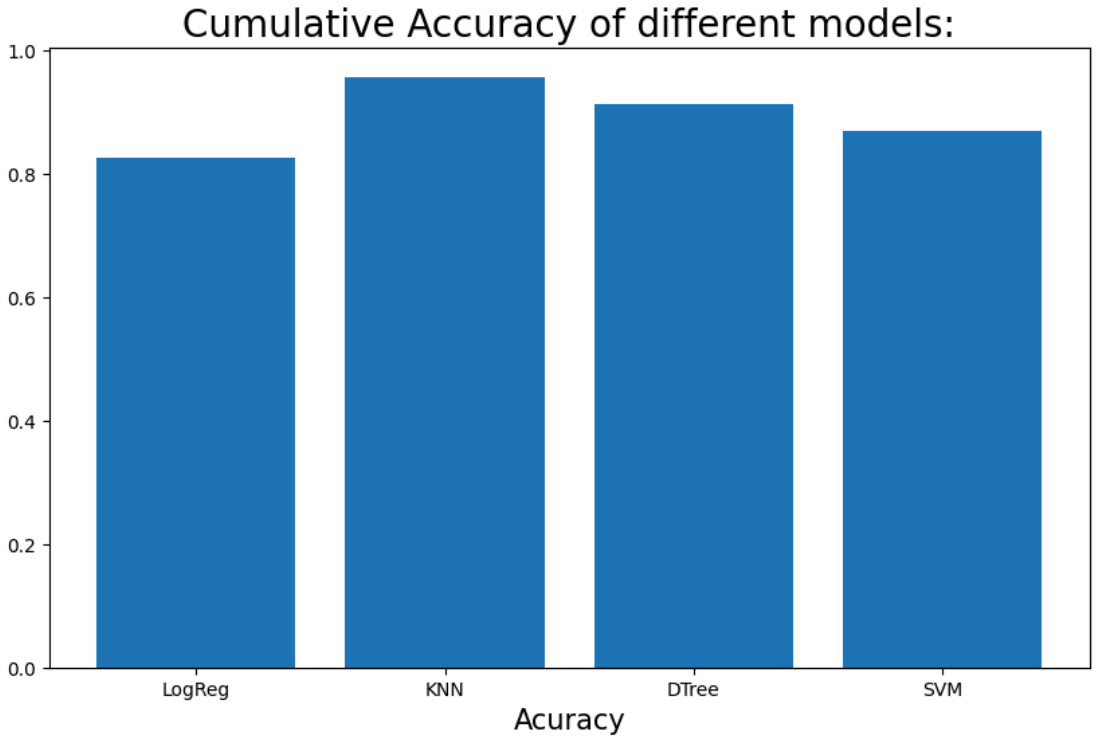
Matplotlib is a library that is used for creating visualisations and plots. It provides a set of functions to create various types of plots, such as line, bar, scatter, and histogram. In this project, Matplotlib is used to create scatter plots and regression lines to visualize the relationship between the input variables and the number of new COVID-19 cases.

Seaborn is a library that is used for creating more advanced visualisations and plots. It provides a high-level interface to create complex visualisations, such as heatmaps, pair plots, and violin plots. In this project, Seaborn is used to create a scatter plot with a regression line and a residual plot to evaluate the performance of the linear regression model.

5. RESULTS AND DISCUSSION

The main objective of the project was to evaluate which model will generate an optimal precision. After obtaining the metrics from training and testing different models let us have a comparison over them. The accuracy scores of the models can be plotted on a bar graph.

```
plt.figure(figsize= (10,6))
acc = [lra,knna,dta,svma]
name = ['LogReg','KNN','DTree','SVM']
plt.title('Cumulative Accuracy of different models:', fontsize = 20)
plt.xlabel('Acuracy', fontsize = 15)
plt.bar(name,acc)
plt.show()
```



We have also obtained other metrics like confusion matrix and classification report. From the above results it is clear that KNN generated the highest amount of accuracy while Logistic Regression generated lower accuracy compared to other models.

6. CONCLUSION AND FUTURE SCOPE

Based on the metrics obtained after fitting the training data and comparing the testing data, accuracy scores of each model have been obtained. Out of which KNN generated the highest accuracy. Hence, KNN can be used for accurate prediction.

Even Though the accuracy generated by KNN model is reasonable enough, it can further be developed by reducing the number of features with feature reduction techniques like Principal Component Analysis, Backward Elimination, Forward Selection, Score comparison, Missing Value Ratio etc.

Furthermore, this project can be expanded by using more powerful machine learning models such as Random Forest, Gradient Boosting, or Neural Networks to improve prediction accuracy.

Overall, This model will have a significant impact on maritime security and defense. This model can be further used to detect any kind of mineral, if their frequency data is provided.

7. REFERENCES

Dataset from https://www.kaggle.com/datasets used in Gorman, R. P., and Sejnowski, T. J. (1988). "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75–89.

Pandas documentation. (2021). Retrieved from https://pandas.pydata.org/docs/

NumPy documentation. (2021). Retrieved from https://numpy.org/doc/stable/

Seaborn documentation. (2021). Retrieved from https://seaborn.pydata.org/

Scikit-learn documentation. (2021). Retrieved from

https://scikit-learn.org/stable/documentation.html

Matplotlib documentation. (2021). Retrieved from

https://matplotlib.org/stable/contents.html

.