Abstract:

Underwater mines pose extreme danger for ships and submarines. Therefore, navies around the world use mine countermeasure (MCM) units to protect against them. One of the measures used by MCM units is mine hunting, which requires searching for all the mines in a suspicious area. It is generally divided into four stages: detection, classification, identification and disposal. The detection and classification steps are usually performed using a sonar mounted on a ship's hull or on an underwater vehicle. After retrieving the sonar data, military personnel scan the seabed images to detect targets and classify them as mine-like objects (MLOs) or benign objects. To reduce the technical operator's workload and decrease post-mission analysis time, computer-aided detection (CAD), computer-aided classification (CAC) and automated target recognition (ATR) algorithms have been introduced. This paper reviews mine detection and classification techniques used in the aforementioned systems. The author considered current and previous generation methods starting with classical image processing, and then machine learning followed by deep learning. This review can facilitate future research to introduce improved mine detection and classification algorithms.

Keywords:
mine detection; **mine classification**; **sonar imagery**; **mine countermeasure**; **mine-like object.**

• Introduction to Underwater naval mine dectection with Machine Learning

Underwater mines are a strategic military tool to protect any country’s naval borders. They constitute fully autonomous devices composed of an explosive charge, sensing device and fuse mechanism. Previous generation mines needed physical contact with the ship to trigger an explosion. Navies around the world use mine countermeasure (MCM) units to protect against mines that involve both passive and active tactics. Passive countermeasures modify the specific target vessel characteristics or signatures which are used to trigger mines. These include building vessels with fibreglass or altering a steel vessel’s magnetic field through degaussing. Alternatively, active countermeasures aim to find mines using specially designed ships or underwater vehicles. The first active measure, minesweeping, utilises a contact sweep or wire drag to cut the mooring wires of floating mines. In other scenarios, it uses a distance sweep to mimic a ship. The second active measure, mine hunting, requires searching for all the mines in a suspicious area. It is generally divided into four stages:

- Detection—finding targets from different signals (acoustic, magnetic);
- Classification—determining if the target is a mine-like object (MLO) or a benign object;
- Identification—using additional information (diver, underwater vehicle equipped with a camera) to validate classification results;
- Disposal—neutralising a mine.

To reduce the technical operator’s workload and decrease post-mission analysis time, computer-aided detection (CAD), computer-aided classification (CAC) and automated target recognition (ATR) algorithms have been introduced. They are based on the analysis of different types of image characteristics, which can be grouped into three categories [1]:

- Texture-based features—patterns and local variations of the image intensity;

- Geometrical features—e.g., length, area;
- Spectral features—e.g., colour, energy.

History of Machine Learning (ML) :

The history of ML dates back to the mid – 20th century when pioneers like ALAN TURING and JOHN McCarthy laid the theoretical foundations. However, it gained significant momentum with the advent of digital computers and the development of early learning – algorithms. The field has evolved through several phases, including rule – based systems, expert systems, and the emergence of neural networks. In recent years, ML has witnessed unprecedented growth, thanks to advances in data availability, computing power, and algorithmic innovations.

Types of Machine Learning (ML):

1. Supervised Learning: In this approach, the algorithm learns from labeled data and aims to discover predictions or decisions. It's commonly used for stock price prediction, classification of stocks , and risk assessment.

2. Un-supervised Learning: This deals with unlabelled data and aims to discover patterns, clusters, or hidden structure with in the data. Clustering algorithms can help in portfolio diversification, and identifying market trends.

3. Reinforcement Learning: This type of ML involves training models to make sequence of decisions that maximize a reward. In the stock market, it can be used for portfolio management and trading strategy optimization.

Problem statement

Background: Naval mines pose a severe threat to maritime security, potentially causing significant damage to ships and submarines. Traditional methods of underwater naval mine detection, including sonar systems and specialized sensors, face challenges in accurately and efficiently distinguishing between mines and natural elements in complex and dynamic underwater environments.

Challenges:

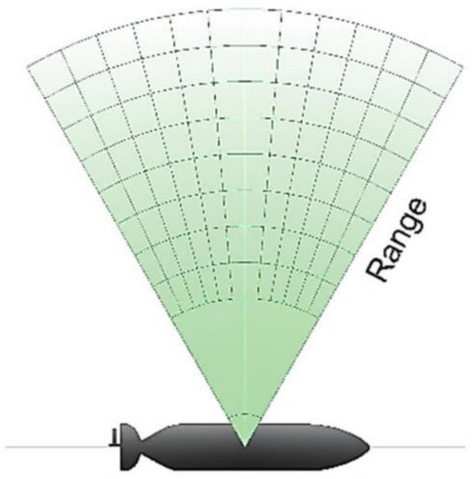
- **False Positives and Negatives:** Current detection methods often result in false positives, where natural features are mistaken for mines, and false negatives, where actual mines go undetected. These errors compromise the reliability of naval mine detection systems.
- **Adaptability to Varied Mine Characteristics:** Mines exhibit diverse shapes, sizes, and materials, making it difficult for existing detection technologies to provide a comprehensive and adaptable solution. A one-size-fits-all approach is not sufficient.
- **Environmental Variability:** Underwater conditions, including variations in water temperature, salinity, and acoustic properties, introduce uncertainties that affect the performance of detection systems. The ability to operate effectively under diverse environmental conditions is crucial.
- **Limited Processing Speed:** The processing speed of traditional detection methods may not be sufficient for real-time decision-making, potentially delaying timely responses to mine threats during naval operations.
- **Insufficient Data Integration:** The integration of data from multiple sensors, including sonar, magnetic sensors, and environmental sensors, can be challenging. Achieving seamless integration is essential for a holistic and accurate understanding of the underwater environment.
- **Cost and Resource Constraints:** Developing and maintaining effective underwater naval mine detection systems can be resource-intensive. Cost-effective solutions that do not compromise on performance are needed.
- Underwater environments are complex and dynamic, making it challenging to distinguish between mines and natural features.

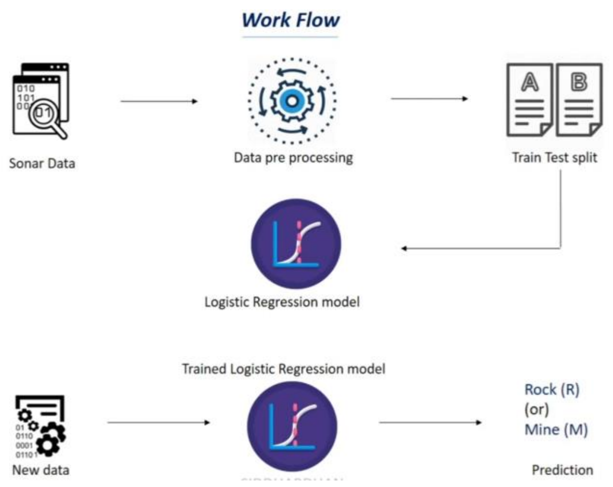
- Mines can have various shapes, sizes, and materials, further complicating detection.
- Environmental conditions such as salinity, temperature, and acoustic properties can affect the performance of traditional detection methods.

Object Classification

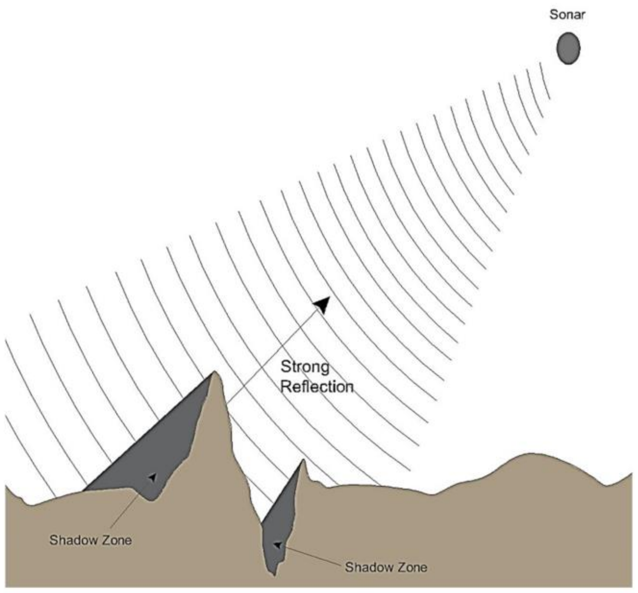
For object classification, a set of features needs to be extracted. Based on the extracted features, the classification can be performed by calculating distances between them [42]. Other approaches utilise parallel lines via the Hough transform [29] or shape and signal-to-noise ratio analyses [41]. For example, Dobeck et al. [50] proposed 25 features due to their uniqueness. The most significant ones are listed below:

- Maximum highlight pixel value in the ROI;
- Minimum pixel value in the ROI;
- Size of highlight region;
- Size of shadow region;
- Mean of highlight region;
- Mean of shadow region;
- Distance from shadow to highlight;
- Number of shadow pixels below threshold;
- Number of highlight pixels above the threshold.





Highlights and Shadows:



Objectives:

- Develop a robust underwater naval mine detection system that minimizes both false positives and false negatives, ensuring high accuracy in identifying mines while avoiding unnecessary alarms.
- Create a detection system that can adapt to the diverse characteristics of different types of mines, including variations in shape, size, and material, thereby improving overall effectiveness.
- Enhance the system's ability to operate in varied underwater environments by accounting for factors such as temperature, salinity, and acoustic conditions, ensuring reliable performance under different circumstances.
- Improve processing speed to enable real-time decision-making, providing naval personnel with timely information to navigate safely through potentially mined areas.
- Integrate data from multiple sensors seamlessly, ensuring a comprehensive understanding of the underwater environment and enhancing the overall accuracy of mine detection.
- Explore cost-effective solutions that leverage advancements in technology and machine learning, addressing resource constraints without compromising the system's performance.

Description of the problem on which the project focuses

The project focuses on the development and implementation of an advanced underwater naval mine detection system leveraging machine learning techniques. The primary challenge is to enhance the accuracy, adaptability, and real-time capabilities of mine detection in complex underwater environments. Traditional methods, while effective to a certain extent, are limited by issues such as false positives, difficulties in adapting to diverse mine characteristics, and the need for real-time decision-making. The integration of machine learning aims to address these challenges and revolutionize the field of underwater mine detection.

- **Data Collection:** Gathering labeled datasets consisting of underwater sensor data with known mine locations is essential for training ML models.
- **Data Preprocessing:** Preparing the data involves cleaning, normalizing, and extracting relevant features to facilitate effective training.
- **Feature Extraction:** Identifying and extracting meaningful features from the sensor data, such as acoustic signatures and shape characteristics, is crucial for training accurate ML models.
- **Model Training:** Employing supervised learning techniques, ML models are trained on labeled datasets to recognize patterns associated with mines.
- **Deployment:** Trained ML models are integrated into existing naval mine detection systems, working alongside traditional methods to enhance overall detection capabilities.
- **Real-time Decision-making:** ML models contribute to real-time decision-making, providing naval personnel with timely information to navigate safely through potentially mined areas.
- **Continuous Learning:** ML models can continuously learn and adapt based on new data, improving their accuracy and effectiveness over time.

- **Collaboration and Research:** Collaboration between naval experts and ML researchers is crucial for advancing the field. Ongoing research contributes to the development of more sophisticated algorithms and the incorporation of emerging sensor technologies.

In summary, the integration of machine learning into underwater naval mine detection represents a significant advancement in maritime security. The adaptability, pattern recognition, and real-time processing capabilities of ML models contribute to more accurate and efficient detection, ultimately enhancing the safety of naval operations in challenging underwater environments.

Significance of the Project:

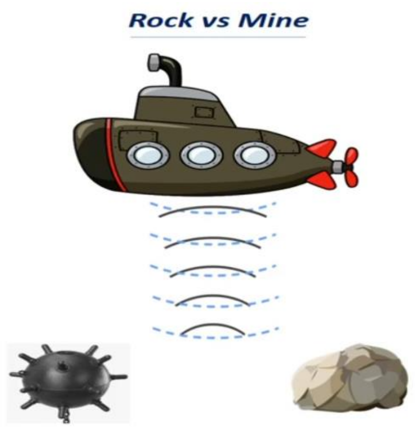
Underwater naval mine detection is a critical aspect of naval operations to ensure the safety of ships and submarines. Traditional methods of mine detection often involve human divers or specialized equipment, which can be time-consuming, expensive, and risky. Machine learning (ML) has the potential to significantly enhance the efficiency and effectiveness of underwater mine detection. Here are some key aspects highlighting the significance of using machine learning in this domain:

- **Increased Efficiency:** Machine learning algorithms can process large amounts of data quickly and efficiently. This is particularly important in underwater mine detection, where vast areas need to be scanned. ML models can analyze sonar data, images, and other sensor information much faster than traditional methods, leading to quicker identification of potential threats.
- **Automation:** ML models can automate the detection process, reducing the need for constant human supervision. This can be crucial in hazardous underwater environments, where the presence of mines poses a significant risk to human divers.
- **Enhanced Accuracy:** Machine learning algorithms can be trained on diverse datasets, enabling them to learn complex patterns and improve

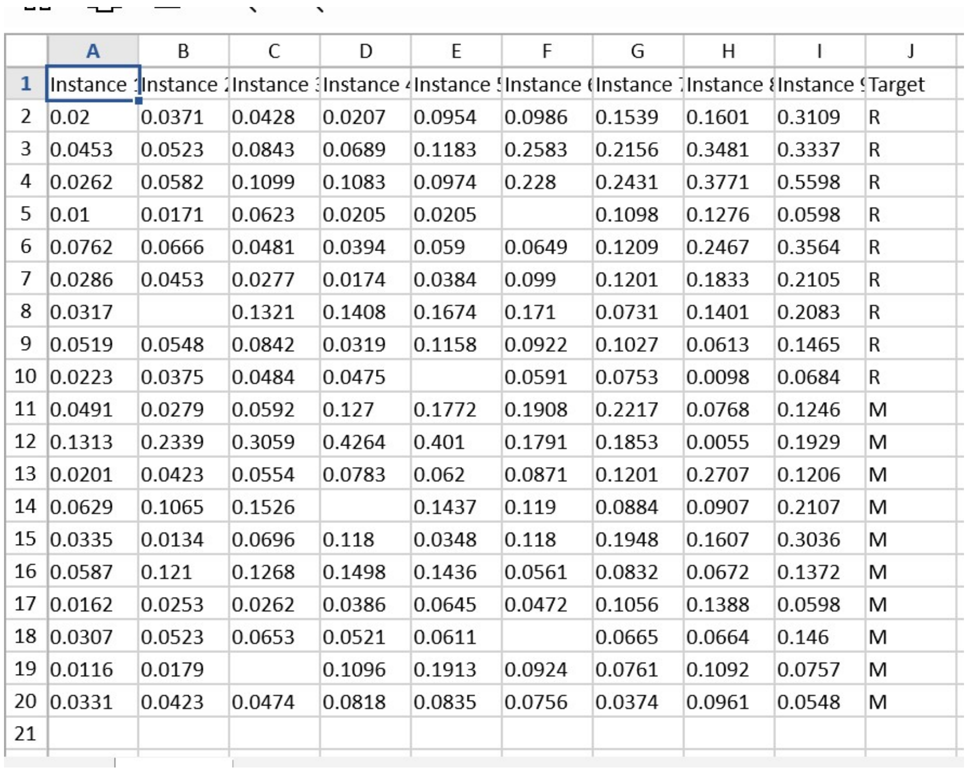
accuracy over time. This can lead to a reduction in false positives and false negatives, ensuring that potential threats are accurately identified.

- **Adaptability:** ML models can adapt to changing conditions and evolving threats. As new types of mines are developed, machine learning algorithms can be trained to recognize these emerging threats, providing a level of adaptability that may be challenging for traditional detection methods.
- **Real-time Decision Making:** Machine learning enables real-time analysis of sensor data. This rapid processing allows for quicker decision-making in identifying and responding to potential threats, thereby improving the overall situational awareness and response time.
- **Cost-effectiveness:** While implementing machine learning systems initially incurs costs, the long-term benefits include reduced reliance on expensive manual methods, lower operational costs, and potentially enhanced fleet safety, making it a cost-effective solution.
- **Integration with Sensor Networks:** ML models can integrate with various sensor networks, such as sonar and imaging systems, to provide a comprehensive and multi-modal approach to mine detection. This integration allows for a more holistic understanding of the underwater environment.
- **Reduced Risk to Personnel:** By automating the mine detection process, machine learning reduces the need for human divers to engage in high-risk activities. This significantly enhances the safety of naval personnel involved in mine clearance operations.

In summary, the application of machine learning in underwater naval mine detection brings about improvements in efficiency, accuracy, adaptability, and overall safety. As technology continues to advance, the integration of machine learning into naval operations is likely to become increasingly crucial for maintaining maritime security.



Data Collection , Data set and Description of the Dataset Data sources:



	A	B	C	D	E	F	G	H	I	J
1	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Instance 8	Instance 9	Target
2	0.02	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	R
3	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	R
4	0.0262	0.0582	0.1099	0.1083	0.0974	0.228	0.2431	0.3771	0.5598	R
5	0.01	0.0171	0.0623	0.0205	0.0205		0.1098	0.1276	0.0598	R
6	0.0762	0.0666	0.0481	0.0394	0.059	0.0649	0.1209	0.2467	0.3564	R
7	0.0286	0.0453	0.0277	0.0174	0.0384	0.099	0.1201	0.1833	0.2105	R
8	0.0317		0.1321	0.1408	0.1674	0.171	0.0731	0.1401	0.2083	R
9	0.0519	0.0548	0.0842	0.0319	0.1158	0.0922	0.1027	0.0613	0.1465	R
10	0.0223	0.0375	0.0484	0.0475		0.0591	0.0753	0.0098	0.0684	R
11	0.0491	0.0279	0.0592	0.127	0.1772	0.1908	0.2217	0.0768	0.1246	M
12	0.1313	0.2339	0.3059	0.4264	0.401	0.1791	0.1853	0.0055	0.1929	M
13	0.0201	0.0423	0.0554	0.0783	0.062	0.0871	0.1201	0.2707	0.1206	M
14	0.0629	0.1065	0.1526		0.1437	0.119	0.0884	0.0907	0.2107	M
15	0.0335	0.0134	0.0696	0.118	0.0348	0.118	0.1948	0.1607	0.3036	M
16	0.0587	0.121	0.1268	0.1498	0.1436	0.0561	0.0832	0.0672	0.1372	M
17	0.0162	0.0253	0.0262	0.0386	0.0645	0.0472	0.1056	0.1388	0.0598	M
18	0.0307	0.0523	0.0653	0.0521	0.0611		0.0665	0.0664	0.146	M
19	0.0116	0.0179		0.1096	0.1913	0.0924	0.0761	0.1092	0.0757	M
20	0.0331	0.0423	0.0474	0.0818	0.0835	0.0756	0.0374	0.0961	0.0548	M
21										

Creating a dataset for underwater naval mine detection involves collecting relevant data from various sources. The dataset should ideally include diverse examples of underwater environments, different types of mines, and variations in sensor readings. Below is an outline of the data collection process, the components of the dataset, and a description of potential data attributes:

Data Collection:

- **Sonar Data:**
- Collect sonar data from different underwater locations, including coastal areas, open seas, and harbors.
- Vary the depth of the water to capture different underwater terrains.
- Use both side-scan and forward-looking sonar systems to cover a wide range of detection scenarios.
- **Imaging Data:**
- Capture images using underwater cameras to supplement sonar data.
- Include images of different mine types and environmental conditions.
- Ensure variations in lighting, water clarity, and backgrounds.
- **Environmental Factors:**
- Record environmental data such as water temperature, salinity, and currents.
- Include data on the seabed composition, which can affect mine detection.
- **Mines:**
- Collect information on different types of naval mines, including both historical and modern variations.
- Include mines made of various materials, sizes, and shapes.
- Simulate scenarios with multiple mines in close proximity.
- **False Positives/Negatives:**
- Include instances that mimic false positives and false negatives to train the model on realistic scenarios.
- Introduce natural elements that might be mistaken for mines and vice versa.

Dataset Components:

- **Sonar Data:**
- A collection of sonar readings in standard file formats.
- Each record should include information on the location, timestamp, and sensor specifications.
- **Images:**
- A set of underwater images in common image formats.
- Annotations or labels indicating the presence or absence of mines in each image.
- **Environmental Data:**
- A dataset containing environmental parameters corresponding to each sonar and image record.
- **Mine Information:**
- A catalog of different types of naval mines, including specifications such as size, weight, and composition.
- Metadata linking each mine type to instances in the dataset.
- **Annotations:**
- Ground truth annotations indicating the location and type of mines in the dataset.
- Annotations for false positives/negatives to train the model on different scenarios.

Description of Dataset Attributes:

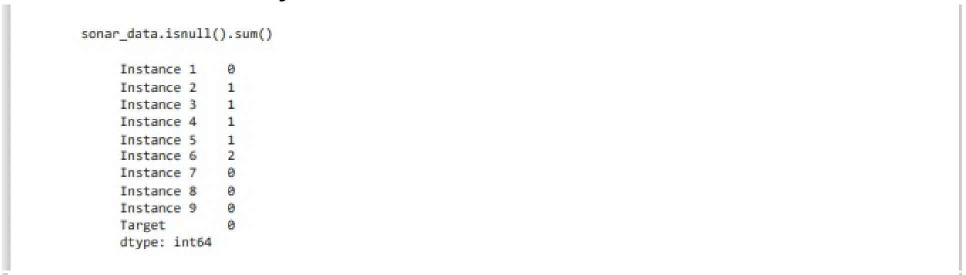
- **Sonar Data Attributes:**
- Sonar frequency, beam angle, and range.
- Timestamp and geographic coordinates.
- Amplitude and phase information.
- **Image Data Attributes:**
- Image resolution, format, and color depth.
- Timestamp and geographic coordinates.
- Annotations indicating mine presence or absence.
- **Environmental Data Attributes:**
- Water temperature, salinity, and current speed.
- Seabed composition and terrain details.

- **Mine Information Attributes:**
- Mine type, size, weight, and composition.
- Historical information about the mine's use.
- **Annotations Attributes:**
- Location (coordinates) of mines in sonar and image data.
- Type of mine (for positive instances) or annotation indicating a false positive/negative.

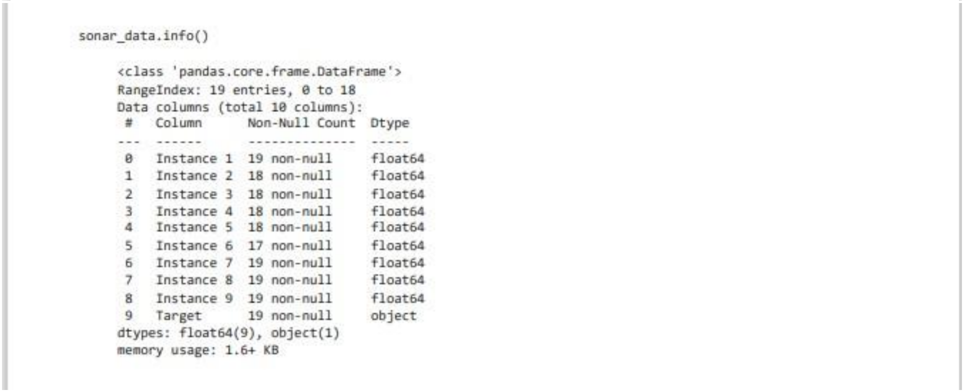
When building and using the dataset, it's crucial to ensure a balance of positive and negative examples, representativeness of different scenarios, and a diverse set of environmental conditions. Additionally, proper data preprocessing steps, such as normalization and cleaning, should be applied to ensure the quality of the dataset.

Data cleaning and Preprocessing – Description of the code Handing missing data:

Handing missing data: Checking for missing values in the dataset and decide on an appropriate strategies for handling them, such as filling missing values or removing rows with missing data Outlier detection: Identify and handle outliers in the temperature and other variables. Outliers can significantly affect the accuracy of the model.

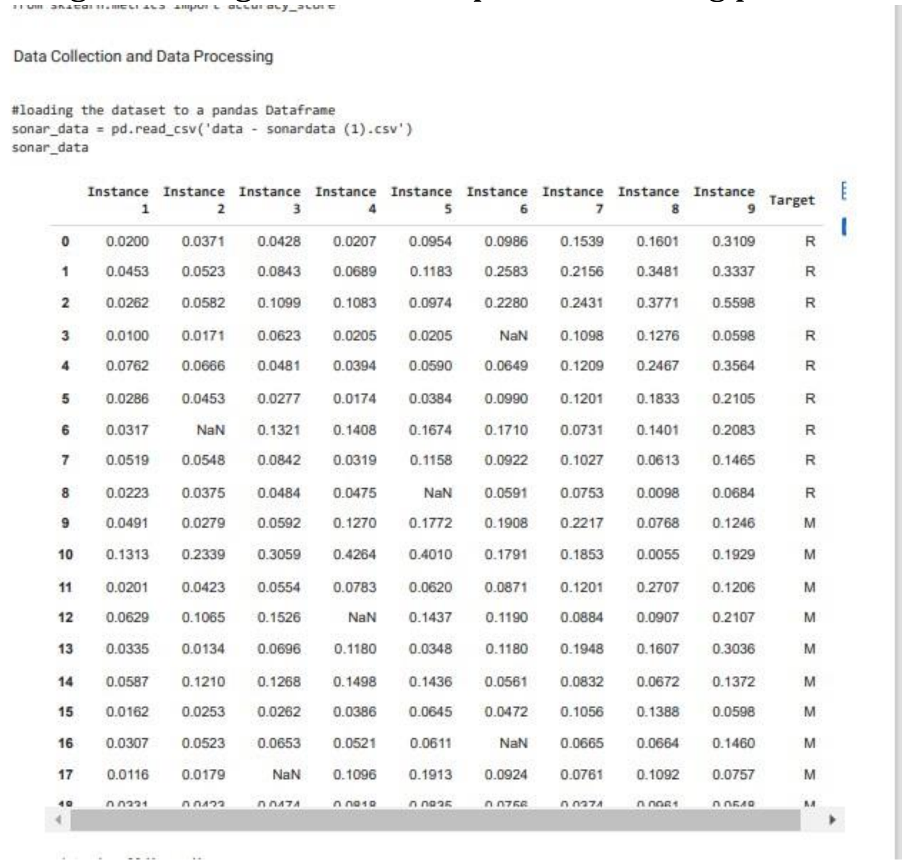


```
sonar_data.isnull().sum()

    Instance 1    0
    Instance 2    1
    Instance 3    1
    Instance 4    1
    Instance 5    1
    Instance 6    2
    Instance 7    0
    Instance 8    0
    Instance 9    0
    Target        0
    dtype: int64
```



```
sonar_data.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 19 entries, 0 to 18
    Data columns (total 10 columns):
     #   Column      Non-Null Count  Dtype
    ---  ------      --------------  -----
     0   Instance 1  19 non-null     float64
     1   Instance 2  18 non-null     float64
     2   Instance 3  18 non-null     float64
     3   Instance 4  18 non-null     float64
     4   Instance 5  18 non-null     float64
     5   Instance 6  17 non-null     float64
     6   Instance 7  19 non-null     float64
     7   Instance 8  19 non-null     float64
     8   Instance 9  19 non-null     float64
     9   Target      19 non-null     object
    dtypes: float64(9), object(1)
    memory usage: 1.6+ KB
```

Data Encoding: When there is a categorical data in your dataset it's impossible to work any operations on the dataset. To overcome the problem we can encode our data set in the form on numeric values. Here I used Ordinal encoding according to that it will provide floating point values.



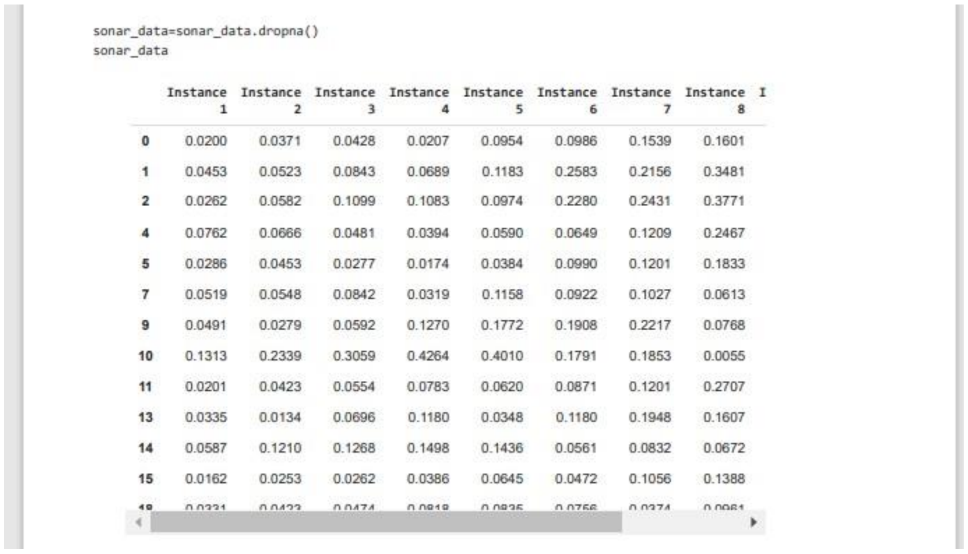
```
from sklearn.metrics import accuracy_score
```

Data Collection and Data Processing

```
#loading the dataset to a pandas Dataframe
sonar_data = pd.read_csv('data - sonardata (1).csv')
sonar_data
```

	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Instance 8	Instance 9	Target
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	R
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	R
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	R
3	0.0100	0.0171	0.0623	0.0205	0.0205	NaN	0.1098	0.1276	0.0598	R
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	R
5	0.0286	0.0453	0.0277	0.0174	0.0384	0.0990	0.1201	0.1833	0.2105	R
6	0.0317	NaN	0.1321	0.1408	0.1674	0.1710	0.0731	0.1401	0.2083	R
7	0.0519	0.0548	0.0842	0.0319	0.1158	0.0922	0.1027	0.0613	0.1465	R
8	0.0223	0.0375	0.0484	0.0475	NaN	0.0591	0.0753	0.0098	0.0684	R
9	0.0491	0.0279	0.0592	0.1270	0.1772	0.1908	0.2217	0.0768	0.1246	M
10	0.1313	0.2339	0.3059	0.4264	0.4010	0.1791	0.1853	0.0055	0.1929	M
11	0.0201	0.0423	0.0554	0.0783	0.0620	0.0871	0.1201	0.2707	0.1206	M
12	0.0629	0.1065	0.1526	NaN	0.1437	0.1190	0.0884	0.0907	0.2107	M
13	0.0335	0.0134	0.0696	0.1180	0.0348	0.1180	0.1948	0.1607	0.3036	M
14	0.0587	0.1210	0.1268	0.1498	0.1436	0.0561	0.0832	0.0672	0.1372	M
15	0.0162	0.0253	0.0262	0.0386	0.0645	0.0472	0.1056	0.1388	0.0598	M
16	0.0307	0.0523	0.0653	0.0521	0.0611	NaN	0.0665	0.0664	0.1460	M
17	0.0116	0.0179	NaN	0.1096	0.1913	0.0924	0.0761	0.1092	0.0757	M
18	0.0331	0.0423	0.0474	0.0818	0.0835	0.0756	0.0374	0.0961	0.0548	M

`dropna()` **:**function is a method commonly used in data manipulation libraries like Pandas in Python. This function is used to remove missing (NaN) values from a DataFrame or Series. Missing values can occur in datasets for various reasons, such as incomplete data collection or data corruption, and handling them is an essential step in data preprocessing.



```
sonar_data=sonar_data.dropna()
sonar_data
```

	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Instance 8	I
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	
5	0.0286	0.0453	0.0277	0.0174	0.0384	0.0990	0.1201	0.1833	
7	0.0519	0.0548	0.0842	0.0319	0.1158	0.0922	0.1027	0.0613	
9	0.0491	0.0279	0.0592	0.1270	0.1772	0.1908	0.2217	0.0768	
10	0.1313	0.2339	0.3059	0.4264	0.4010	0.1791	0.1853	0.0055	
11	0.0201	0.0423	0.0554	0.0783	0.0620	0.0871	0.1201	0.2707	
13	0.0335	0.0134	0.0696	0.1180	0.0348	0.1180	0.1948	0.1607	
14	0.0587	0.1210	0.1268	0.1498	0.1436	0.0561	0.0832	0.0672	
15	0.0162	0.0253	0.0262	0.0386	0.0645	0.0472	0.1056	0.1388	
[illegible]	0.0331	0.0423	0.0474	0.0818	0.0835	0.0756	0.0374	0.0961	

FEATURE SCALING:

• Feature scaling is a data preprocessing technique used to transform the values of features or variables in a dataset to a similar scale.

• The purpose is to ensure that all features contribute equally to the model and to avoid the domination of features with larger values.

• Feature scaling becomes necessary when dealing with datasets containing features that have different ranges, units of measurement, or orders of magnitude.

• In such cases, the variation in feature values can lead to biased model performance or difficulties during the learning process.

• Scaling facilitates meaningful comparisons between features, improves model convergence, and prevents certain features from overshadowing others based solely on their magnitude.

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

```
(13, 10)
```

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

	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Ins
count	13.000000	13.000000	13.000000	13.000000	13.000000	13.000000	13.000000	13.0
mean	0.045400	0.063108	0.083654	0.100500	0.114685	0.122685	0.146492	0.1

```
sonar_data['Target'].value_counts()
```

```
M    7
R    6
Name: Target, dtype: int64
```

M --> Mine

R --> Rock

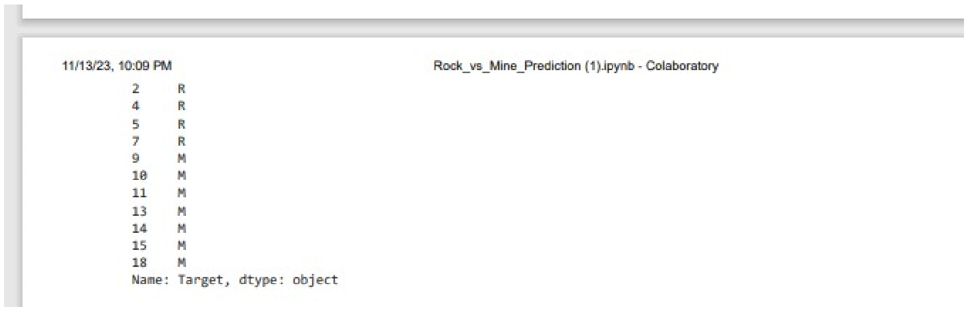
```
sonar_data.groupby('Target').mean()
```

	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Instanc
Target								
M	0.048857	0.072300	0.098643	0.145700	0.138086	0.107700	0.135443	0.11654
R	0.041367	0.052383	0.066167	0.047767	0.087383	0.140167	0.159383	0.22943

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

```
print(X)
print(Y)
```

```
    Instance 1  Instance 2  Instance 3  Instance 4  Instance 5  Instance 6  \
0       0.0200      0.0371      0.0428      0.0207      0.0954      0.0986
1       0.0453      0.0523      0.0843      0.0689      0.1183      0.2583
2       0.0262      0.0582      0.1099      0.1083      0.0974      0.2280
4       0.0762      0.0666      0.0481      0.0394      0.0590      0.0649
5       0.0286      0.0453      0.0277      0.0174      0.0384      0.0990
7       0.0519      0.0548      0.0842      0.0319      0.1158      0.0922
9       0.0491      0.0279      0.0592      0.1270      0.1772      0.1908
10      0.1313      0.2339      0.3059      0.4264      0.4010      0.1791
11      0.0201      0.0423      0.0554      0.0783      0.0620      0.0871
13      0.0335      0.0134      0.0696      0.1180      0.0348      0.1180
14      0.0587      0.1210      0.1268      0.1498      0.1436      0.0561
15      0.0162      0.0253      0.0262      0.0386      0.0645      0.0472
18      0.0331      0.0423      0.0474      0.0818      0.0835      0.0756

    Instance 7  Instance 8  Instance 9
0       0.1539      0.1601      0.3109
1       0.2156      0.3481      0.3337
2       0.2431      0.3771      0.5598
4       0.1209      0.2467      0.3564
5       0.1201      0.1833      0.2105
7       0.1027      0.0613      0.1465
9       0.2217      0.0768      0.1246
10      0.1853      0.0055      0.1929
11      0.1201      0.2707      0.1206
13      0.1948      0.1607      0.3036
14      0.0832      0.0672      0.1372
15      0.1056      0.1388      0.0598
18      0.0374      0.0961      0.0548
0     R
1     R
```



11/13/23, 10:09 PM Rock_vs_Mine_Prediction (1).ipynb - Colaboratory

```
2     R
4     R
5     R
7     R
9     M
10    M
11    M
13    M
14    M
15    M
18    M
Name: Target, dtype: object
```

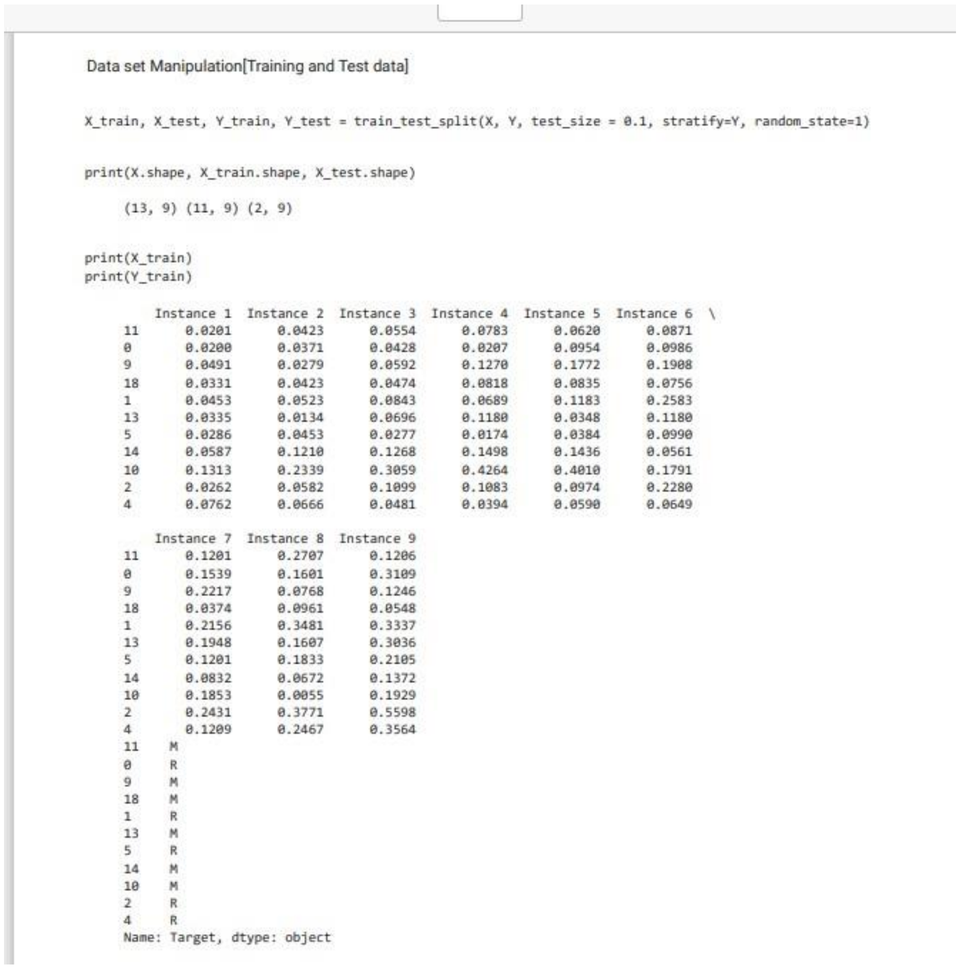
Data set manipulation

Data manipulation is a critical step in the data analysis process. It involves cleaning, transforming, and reshaping data to make it suitable for analysis or modeling. In Python, the pandas library is widely used for data manipulation tasks. Here are some common data manipulation tasks and how to perform them using pandas.

# Import necessary libraries

 import numpy as np

import pandas as pd



Data set Manipulation[Training and Test data]

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, stratify=Y, random_state=1)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(13, 9) (11, 9) (2, 9)
```

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

```
    Instance 1  Instance 2  Instance 3  Instance 4  Instance 5  Instance 6  \
11      0.0201      0.0423      0.0554      0.0783      0.0620      0.0871
0       0.0200      0.0371      0.0428      0.0207      0.0954      0.0986
9       0.0491      0.0279      0.0592      0.1270      0.1772      0.1908
18      0.0331      0.0423      0.0474      0.0818      0.0835      0.0756
1       0.0453      0.0523      0.0843      0.0689      0.1183      0.2583
13      0.0335      0.0134      0.0696      0.1180      0.0348      0.1180
5       0.0286      0.0453      0.0277      0.0174      0.0384      0.0990
14      0.0587      0.1210      0.1268      0.1498      0.1436      0.0561
10      0.1313      0.2339      0.3059      0.4264      0.4010      0.1791
2       0.0262      0.0582      0.1099      0.1083      0.0974      0.2280
4       0.0762      0.0666      0.0481      0.0394      0.0590      0.0649

    Instance 7  Instance 8  Instance 9
11      0.1201      0.2707      0.1206
0       0.1539      0.1601      0.3109
9       0.2217      0.0768      0.1246
18      0.0374      0.0961      0.0548
1       0.2156      0.3481      0.3337
13      0.1948      0.1607      0.3036
5       0.1201      0.1833      0.2105
14      0.0832      0.0672      0.1372
10      0.1853      0.0055      0.1929
2       0.2431      0.3771      0.5598
4       0.1209      0.2467      0.3564
11    M
0     R
9     M
18    M
1     R
13    M
5     R
14    M
10    M
2     R
4     R
Name: Target, dtype: object
```

Algorithm description and usage in the code

Logistic Regression is a classification algorithm commonly used in machine learning. Despite its name, it's primarily employed for binary classification problems, where the target variable has two possible outcomes (classes). In the context of underwater naval mine detection, Logistic Regression can be applied to predict whether a given set of features corresponds to a mine (positive class) or a non-mine (negative class).

Logistic Regression Algorithm Description:

Logistic Regression models the probability that a given input belongs to a particular class using the logistic function (sigmoid function). The logistic function

maps any real-valued number into the range [0, 1], which is suitable for representing probabilities.
The logistic regression model calculates the weighted sum of the input features, applies the logistic function, and outputs a probability. If the probability is greater than a chosen threshold (commonly 0.5), the instance is predicted as belonging to the positive class; otherwise, it is predicted as belonging to the negative class.

Usage in Code for Underwater Naval Mine Detection:

Assuming you have a dataset with features (sonar readings, environmental conditions, etc.) and corresponding labels indicating whether each instance is a mine or not, you can use Logistic Regression in Python with a machine learning library like Scikit-learn. Below is a simplified example:

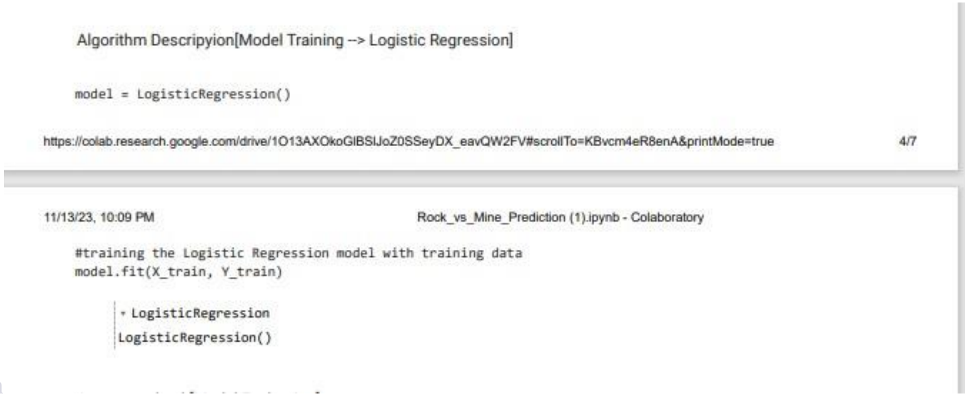
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score



Algorithm Descripyion[Model Training --> Logistic Regression]

```
model = LogisticRegression()
```

https://colab.research.google.com/drive/1O13AXOkoGIBSIJoZ0SSeyDX_eavQW2FV#scrollTo=KBvcm4eR8enA&printMode=true 4/7

11/13/23, 10:09 PM Rock_vs_Mine_Prediction (1).ipynb - Colaboratory

```
#training the Logistic Regression model with training data
model.fit(X_train, Y_train)

    ▾ LogisticRegression
    LogisticRegression()
```

Python

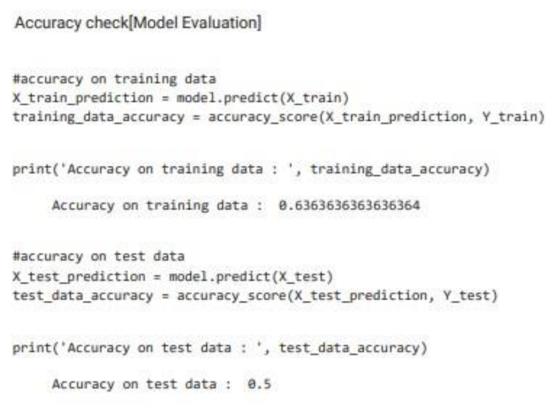
In this code snippet:

- `X` represents the feature matrix.
- `y` represents the target variable (labels).
- The dataset is split into training and testing sets using `train_test_split`.
- A Logistic Regression model is created using `LogisticRegression()`.

- The model is trained on the training set using **`fit`**.
- Predictions are made on the test set using **`predict`**.
- The accuracy and a classification report are printed to evaluate the model's performance.

Remember to customize the code based on your specific dataset and features. Additionally, feature scaling and other preprocessing steps might be necessary depending on the nature of your data.

Accuracy check



Accuracy check[Model Evaluation]

```
#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)

    Accuracy on training data :  0.6363636363636364

#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)

    Accuracy on test data :  0.5
```

Visualization:

In the context of underwater naval mine detection, visualizations can help you gain insights into the performance of your machine learning model and the characteristics of the dataset. Below are some relevant visualizations you might consider:

1. Distribution of Classes:

Visualizing the distribution of mine and non-mine instances in your dataset can help you understand if the classes are balanced or if there's a class imbalance.

2. Feature Distributions:

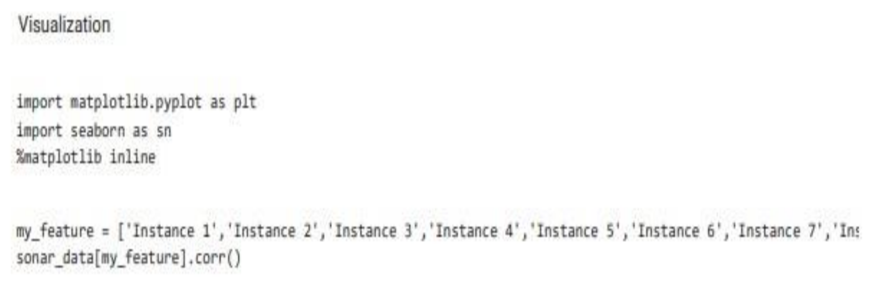
Understanding the distribution of individual features for both classes can provide insights into the separability of the classes.

3. ROC Curve:

The Receiver Operating Characteristic (ROC) curve provides a visual representation of the trade-off between true positive rate (sensitivity) and false positive rate.

4. Confusion Matrix:

A confusion matrix provides a detailed breakdown of the model's performance in terms of true positives, true negatives, false positives, and false negatives.



Visualization

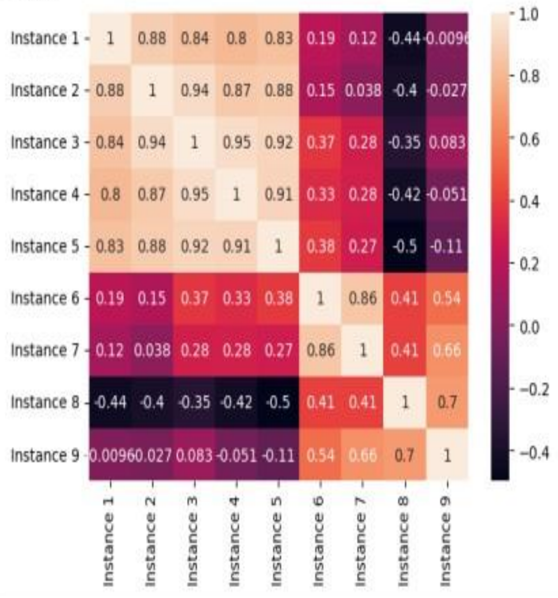
```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

```
my_feature = ['Instance 1','Instance 2','Instance 3','Instance 4','Instance 5','Instance 6','Instance 7','Ins
sonar_data[my_feature].corr()
```

```
<ipython-input-23-2c72f6901ec7>:2: FutureWarning: The default value of numeric_only :
  sonar_data[my_feature].corr()

          Instance  Instance  Instance  Instance  Instance  Instance  Instance  Inst
```

```
sn.heatmap(sonar_data[my_feature].corr(),annot=True)
```

```
<ipython-input-24-0d9169935ce0>:1: FutureWarning: The default value of numeric_only :
  sn.heatmap(sonar_data[my_feature].corr(),annot=True)
<Axes: >
```



Prediction:

Making a Predictive System

```
input_data = (0.0200,0.0371,0.0428,0.0207,0.0954,0.0986,0.1539,0.1601,0.3109)

# 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')
```

```
['M']
The object is a mine
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature
  warnings.warn(
```

Conclusion:

Underwater mine detection and classification in sonar imagery pose a challenging task due to the low quality of sonar images. Sonar images are complicated to analyse due to speckle noise and environmental conditions causing spurious shadows, sidelobe effects and multipath return. This results in significant variability in targets, clutter and background signatures. To deal with these obstacles, numerous image processing techniques have been developed. They can be divided into classical image processing, machine learning and deep learning techniques.

The most conventional approach to MLO detection and classification is classical image processing. It incorporates highlight and shadow regions to spot suspicious objects on the seafloor. This process involves the segmentation step, which divides the image into highlights, shadows and background regions. Then, the templates are used for detecting highlight and shadow combinations. The flaw in classical image processing is that it demands more workforce and training efforts. It requires experts to design image features necessary for object detection and classification. Additionally, its performance is image quality dependent. Therefore, an extra enhancement step is needed to ensure its accuracy. However, a feature that compensates for its flaws is that it does not need an extensive database.

The upcoming trend is to combine classical image processing with DL. A classical method distinguishes ROIs in the detection step while DL classifies ROIs as MLOs or benign objects in this combined technique. This combination can improve the performance of the classification step since the neural network analyses only the ROIs' areas in the image. As a result, the computationally expensive DL algorithms are functional in analysing just small parts of sonar images.

This paper examined the current and previous generations of classical image processing, machine learning and deep learning methods. It can help upcoming research in developing new detection and classification algorithms, thus laying the groundwork for the next generation of advanced techniques.

Funding
This paper was funded by the Polish Ministry of Defence Research Grant entitled “Vision system for object detection and tracking in the marine environment”.

References:

• https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
• https://www.geeksforgeeks.org/machine-learning/
•https://www.geeksforgeeks.org/ml-understanding-dataprocessing/?ref=lbp
• https://files.eric.ed.gov/fulltext/EJ1333895.pdf
•https://www.javatpoint.com/traffic-prediction-using-machine-learning .
•https://www.statology.org/excel-convert-categorical-to-numeric/