SUBMARINE ROCK AND MINE DETECTION

A research project report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (Internet of Things)

Submitted by

SAGIRALA SATYA KARTHIK	**(20BQ1A4944)**
RAVI MAHESH BABU	**(20BQ1A4941)**
MANDA RAKESH	**(20BQ1A4930)**

under the esteemed guidance of

DR. S. KRISHNA PRASAD

Associate Professor

[Program: Computer Science and Engineering (Internet of Things) – CSO]

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(Autonomous)

Approved by AICTE, Permanently Affiliated to JNTUK, NAAC Accredited with 'A' Grade, ISO 9001:2015 Certified

Nambur (V), Pedakakani (M), Guntur (Dt.), Andhra Pradesh – 522 508

2023

[Program: Computer Science and Engineering (Internet of Things) – CSO]

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(Autonomous)

Approved by AICTE, Permanently Affiliated to JNTUK, NAAC Accredited with 'A' Grade, ISO 9001:2015 Certified

Nambur (V), Pedakakani (M), Guntur (Dt.), Andhra Pradesh – 522 508

CERTIFICATE

This is to certify that the research project report entitled "**SUBMARINE ROCK AND MINE DETECTION"** is being submitted by **S.Satya Karthik (**Regd.No**: 20BQ1A4944**), **R.Mahesh Babu (**Regd.No**: 20BQ1A4941**) and **M.Rakesh (**Regd.No**: 20BQ1A4930**) in partial fulfillment of the requirement for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering (Internet of Things)** to the Vasireddy Venkatadri Institute of Technology is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this project have not been submitted to any other university or institute for the award of any degree or diploma.

Signature of the Supervisor	**Head of the Department**
Dr. S. Krishna Prasad,	Dr. Chintalapudi V Suresh
Associate Professor,	Professor & HoD,
Department of CSO, VVIT.	Department of CSO, VVIT.

DECLARATION

We hereby declare that the work embodied in this research project entitled "**Submarine Rock and Mine Detection**", which is being submitted by us in requirement for the B. Tech Degree in **Computer Science and Engineering (Internet of Things)** from Vasireddy Venkatadri Institute of Technology, is the result of investigations carried out by us under the supervision of Dr. S. Krishna Prasad,Associate Professor .

The work is original and the results in this thesis have not been submitted elsewhere for the award of any degree or diploma.

Signature of the Candidates

S. Satya Karthik (Regd.No**: 20BQ1A4944**)

R. Mahesh Babu (Regd.No**: 20BQ1A4941**)

M. Rakesh (Regd.No**: 20BQ1A4930**)

Department Vision

To accomplish the aspirations of emerging engineers to attain global intelligence by obtaining computing and design abilities through communication that elevate them to meet the needs of industry, economy, society, environmental and global.

Department Mission

- To mold the fresh minds into highly competent IoT application developers by enhancing their knowledge and skills in diverse hardware and software design aspects for covering technologies and multi-disciplinary engineering practices.
- To provide the state- of- the art facilities to forge the students industry-ready in IoT system development.
- To nurture the sense of creativity and innovation to adopt socio-economic related activities.
- To promote collaboration with the institutes of national and international repute with a view to have the best careers.
- To enable graduates to emerge as independent entrepreneurs and future leaders.

Program Educational Objectives (PEOs)

PEO-1: To formulate the engineering practitioners to solve industry's technological problems

PEO-2: To engage the engineering professionals in technology development, deployment and engineering system implementation

PEO-3: To instill professional ethics, values, social awareness and responsibility to emerging technology leaders

PEO-4: To facilitate interaction between students and peers in other disciplines of industry and society that contribute to economic growth**.**

PEO-5: To provide the technocrats the amicable environment for the successful pursuing of engineering and management.

PEO-6: To create the right path to pursue their careers in teaching, research and innovation.

Program Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO-1: Proficient and innovative with a strong cognizance in the arenas of sensors, IoT, data science, controllers and signal processing through the application of acquired knowledge and skills.

PSO-2: Apply cutting-edge techniques and tools of sensing and computation to solve multi-disciplinary challenges in industry and society.

PSO-3: Exhibit independent and collaborative research with strategic planning while demonstrating professional and ethical responsibilities of the engineering profession.

Project Outcomes

Students who complete a minor project will:

PW-01. Use the design principles and develop concept for the project

PW-02. Estimate the time frame and cost for the project execution and completion.

PW-03. Analyze the project progress with remedial measures individual in a team.

PW-04. Examine the environmental impact of the project.

PW-05. Demonstrate the project functionality along with a report and presentation.

PW-06. Apply the Engineering knowledge in design and economically manufacturing of components to support the society's needs.

PW-07. Assess health, safety and legal relevance to professional engineering

PW-08. Comply with environmental needs and sustainable development.

PW-09. Justify ethical principles in engineering practices.

PW-010. Perform multi-disciplinary tasks as an individual and / or team member to

PW-011. Comprehend the Engineering activities with effective presentation

PW-012. Interpret the findings with appropriate technological / research citation.

MAPPING OF PROJECT OUTCOMES TO POs

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
PW-01	3	2	2	2								
PW-02	3	2	2								3	
PW-03	3	3		2	3					3		
PW-04	3					3	3	3				3
PW-05	3	2									3	
PW-06	3	2	2	2	3							
PW-07						3						
PW-08							3					
PW-09								3				
PW-10									3		3	
PW-11										3		
PW-12												3
PW-PO	**3**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

MAPPING OF PROJECT OUTCOMES TO PSOs

	PSO1	PSO2	PSO3
PW-01	2	2	2
PW-02			2
PW-03			
PW-04	3	3	3
PW-05			
PW-06	2	2	2
PW-07	2	2	2
PW-08	1	1	1
PW-09	2	2	2
PW-10	2	2	2
PW-11	2	2	2
PW-12	1	1	1
PW-PSO	**2**	**2**	**2**

Note: Strong – 3, Moderate – 2, Low – 1

CONTENTS

ABSTRACT

Submarine rock and mine prediction using machine learning is a pivotal field with applications in maritime security and resource exploration. This study presents an innovative approach to automate the detection of submerged objects, reducing the dependence on costly and time-consuming manual inspections. Our methodology involves integrating various underwater data sources, including sonar imagery, bathymetric data, and environmental parameters, into a comprehensive feature set. We evaluate multiple machine learning algorithms to identify the most effective model for predicting the presence of submarine rocks and mines. This selected model is trained on historical data and fine-tuned for optimal performance. Results demonstrate the potential of machine learning in this domain, offering high accuracy and substantial reductions in inspection time and resources. This research has significant implications for maritime security, underwater exploration, and environmental monitoring. In conclusion, this study highlights the promising prospects of machine learning in submarine rock and mine prediction. It emphasizes the potential for increased efficiency and safety in underwater operations, supporting the case for continued research and development in this field.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND AND SIGNIFICANCE

Submarine rock and mine detection is a field of paramount importance in various domains, including maritime security, resource exploration, and environmental protection. Submerged rocks and mines pose significant threats to the safety of maritime operations, from commercial shipping to naval defense. The consequences of undetected underwater hazards can range from accidents and environmental disasters to national security breaches.

Consequently, the accurate and efficient detection of these objects is imperative. Traditionally, this detection process has heavily relied on manual inspections conducted by divers or remotely operated vehicles (ROVs). While effective, these methods are associated with substantial costs, time requirements, and inherent risks to human operators. Additionally, these manual methods may not provide the comprehensive coverage and accuracy needed to identify hidden or camouflaged hazards beneath the water's surface.

Machine learning, a subfield of artificial intelligence, has emerged as a promising solution to these challenges. By leveraging data-driven algorithms and advanced analytics, machine learning can automate the process of detecting submerged rocks and mines, significantly reducing the dependence on manual inspections. This innovation has the potential to revolutionize how we approach underwater object detection, enhancing safety, efficiency, and resource utilization in maritime and underwater operations.

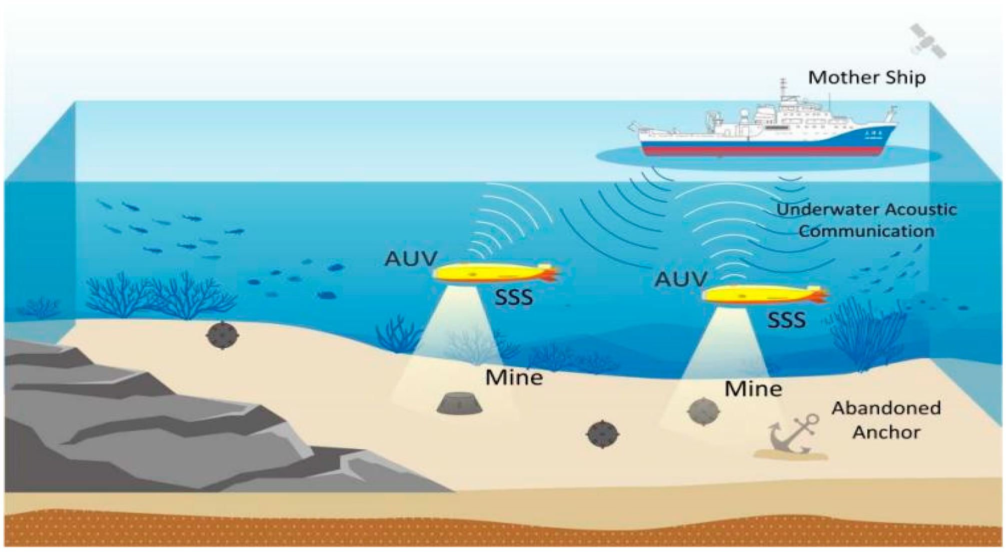


fig 1.1 Sonar Real-Time Method for Underwater-Target Detection

1.2 CHALLENGES IN SUBMARINE OBJECT DETECTION

The task of accurately detecting and predicting the presence of submerged objects such as rocks and mines in underwater environments is riddled with inherent challenges. These challenges have historically hindered the efficiency and reliability of detection methods, necessitating the exploration of more advanced solutions like machine learning. The key challenges in submarine object detection include:

1.2.1 Limited Visibility: Underwater environments often exhibit poor visibility due to factors such as water turbidity, sedimentation, and depth. Reduced visibility makes it difficult for both human divers and traditional sensors to identify submerged objects effectively.

1.2.2 Diverse Object Shapes and Sizes: Submerged objects come in various shapes and sizes, from small mines to large underwater formations. Detecting and classifying these diverse objects accurately requires a robust and adaptable detection system.

1.2.3 Camouflaging and Concealment: Some underwater objects are designed to blend into their surroundings, making them even harder to detect. Mines, for example, may be designed to mimic natural underwater features, increasing the likelihood of their going unnoticed.

1.2.4 High False Positive Rates: Traditional detection methods can produce high false positive rates, leading to unnecessary, resource-intensive inspections. Reducing false positives while maintaining high detection accuracy is a significant challenge.

1.2.5 Data Collection and Labeling: Acquiring underwater data for training machine learning models is a non-trivial task. Collecting diverse and representative datasets is often resource-intensive, and labeling underwater images and sensor data can be time-consuming and error-prone.

1.2.6 Environmental Variability: Underwater environments are subject to fluctuations in temperature, salinity, currents, and marine life, which can affect the performance of detection systems. Adapting to these environmental changes is essential for maintaining detection accuracy.

1.2.7 Safety Concerns: Manual underwater inspections, particularly in military and industrial settings, pose significant safety risks to human divers. Automating detection processes can help mitigate these risks.

1.3 THE ROLE OF MACHINE LEARNING

Machine learning, a subset of artificial intelligence, has emerged as a transformative technology with the potential to revolutionize the field of submarine object detection, including the identification of submerged rocks and mines. Its role in this domain is multifaceted and critical. Machine learning algorithms can be trained to recognize patterns and features in underwater data, enabling the automation of the detection process.

Machine learning models can learn from vast datasets, allowing them to detect subtle underwater features and distinguish between various types of submerged objects. This results in higher detection accuracy compared to traditional methods. Machine learning models can adapt to changing underwater conditions and environmental factors. They can continuously learn and update their knowledge, making them well-suited for dynamic and unpredictable underwater environments.

One of the significant challenges in submarine object detection is the occurrence of false positives. Machine learning models can be fine-tuned to minimize false alarms while maintaining high true-positive rates, reducing unnecessary inspections and resource wastage. Machine learning can integrate data from multiple sources, such as sonar imagery, bathymetric data, and environmental parameters. This data fusion enhances the overall accuracy and reliability of detection systems. By automating detection processes, machine learning can improve operational efficiency and safety

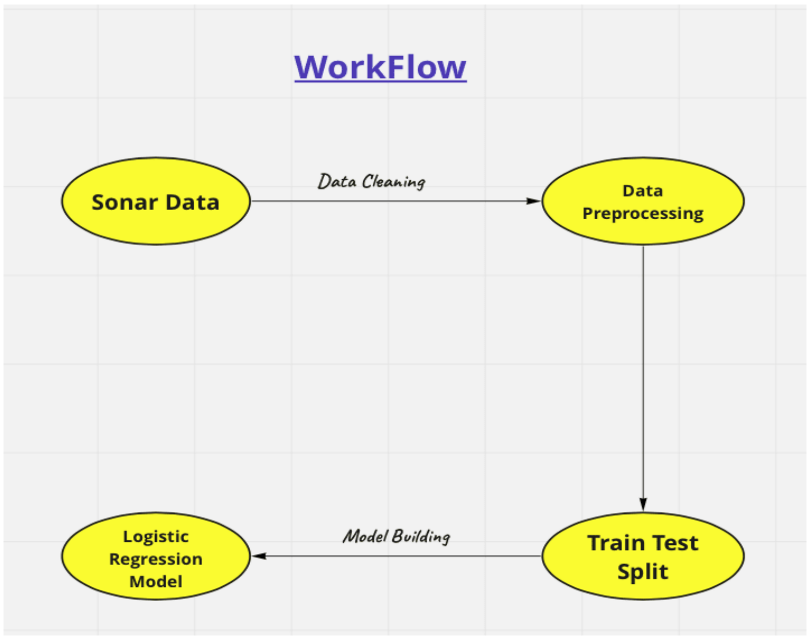


fig 1.2 Workflow of Machine Learning

1.4 RESEARCH OBJECTIVE

The primary objective of this research is to develop and implement a robust machine learning framework for the accurate prediction of submerged rocks and mines in underwater environments. This framework aims to harness the power of artificial intelligence to address the inherent challenges associated with manual inspections and traditional detection methods. Collecting and integrating diverse underwater data sources, including sonar imagery, bathymetric data, and environmental parameters, into a cohesive and comprehensive dataset.

Systematically evaluating a range of machine learning algorithms to identify the most effective model for predicting the presence of submerged objects. This involves selecting algorithms capable of handling the complexity of underwater data. Training the selected machine learning model on historical data to enable it to recognize and classify submerged rocks and mines accurately. Fine-tuning the model to optimize its performance is also a key objective. Performance Evaluation: Rigorously evaluating the performance of the machine learning framework through testing and validation.

Assessing metrics such as accuracy, precision, recall, and F1-score to ensure the reliability and effectiveness of the developed system. Investigating the practical applications of the machine learning framework in real-world scenarios, including maritime security, underwater resource exploration, and environmental monitoring. This includes assessing the potential for cost savings and improved safety. Demonstrating the contribution of this research to the broader field of submarine object detection. Highlighting the innovative approaches and practical significance of leveraging machine learning for underwater hazard prediction.

Through the achievement of these research objectives, we aim to provide a comprehensive and effective solution for submarine rock and mine prediction, offering substantial benefits in terms of safety, efficiency, and resource utilization in underwater operations. In the subsequent sections of this paper, we will present our methodologies, experimental results, and discuss the broader implications of our findings.

1.5 SCOPE OF THE RESEARCH

This research comprehensively delves into the development, implementation, and evaluation of a machine learning framework for submarine rock and mine prediction in underwater environments. We present a detailed account of the methodologies used to collect and preprocess underwater data, explore and select machine learning algorithms, and train and fine-tune the predictive model. Our approach emphasizes the integration of diverse data sources to enhance detection accuracy.

We provide a thorough analysis of the results obtained through rigorous testing and validation of the machine learning framework. Performance metrics such as accuracy, precision, recall, and F1-score will be presented to showcase the effectiveness of the developed system. This paper discusses the practical applications of our research in real-world scenarios. We examine how the deployment of our machine learning framework can improve maritime security by enhancing underwater object detection capabilities, aid in underwater resource exploration by reducing exploration costs and risks, and contribute to environmental protection by facilitating the monitoring of submerged hazards.

We highlight the contributions of this research to the field of submarine object detection. By addressing the challenges associated with manual inspections and traditional detection methods, our work demonstrates the pivotal role of machine learning in advancing safety, efficiency, and reliability in underwater operations. The paper also acknowledges the limitations of our research and suggests potential avenues for future research and development in the field of submarine rock and mine prediction. Beyond the specific focus on underwater object detection, we discuss the broader implications of our findings in the context of artificial intelligence applications in underwater domains and emphasize the importance of continued innovation in this field.

The scope of this paper is aimed at providing a comprehensive understanding of the potential of machine learning in submarine rock and mine prediction. By addressing these aspects, we aim to contribute valuable insights and solutions that can significantly impact maritime safety, resource exploration, and environmental conservation efforts. In the subsequent sections, we will delve into each of these aspects in detail, presenting a holistic view of our research findings and their implications.

CHAPTER 2

RESEARCH METHODOLOGY

2.1 INTRODUCTION TO RESEARCH METHODOLOGY

In this chapter, we delve into the heart of our research endeavor by presenting a comprehensive overview of the research methodology employed in this study. Research methodology constitutes the blueprint that guides the execution of our research, determining the methods and approaches used to answer our research questions and achieve our objectives. It serves as the bridge between our research objectives and the empirical data that underpins our findings.

This chapter is pivotal in illuminating the processes that enabled us to generate the insights presented in this report. It outlines the strategies adopted for data collection, the techniques employed for data analysis, and the ethical considerations governing our research. Moreover, it offers transparency into the decisions made regarding research design and data selection, shedding light on the rationale behind our chosen methodologies.

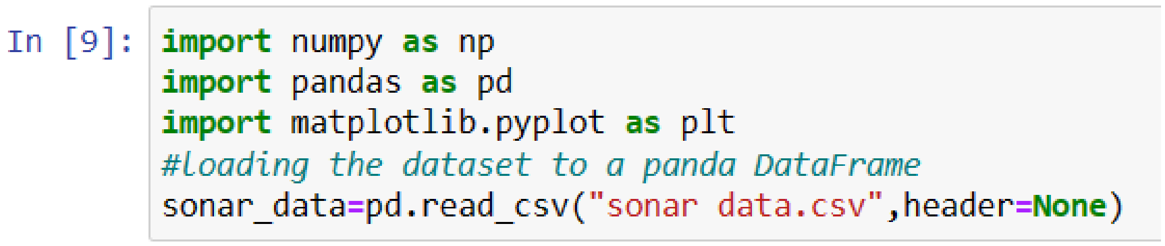
Our research methodology draws from a combination of qualitative and quantitative approaches, carefully tailored to the nature of our study. This section outlines the rationale behind our methodological choices and provides a comprehensive overview of the research design, data collection techniques, data analysis methods, and ethical considerations that underpin our investigation.

Our research methodology is tailored to the specific demands of our study, considering the nature of our research questions and the context in which they are situated. By providing a comprehensive understanding of our research approach, we aim to establish the credibility and validity of our findings. Furthermore, this chapter also highlights any potential limitations and challenges associated with our chosen methodology, demonstrating our commitment to a transparent and accountable research process.

As we embark on this journey through the intricacies of our research methodology, we invite the reader to gain insight into the methods and processes that underpin our study. It is our hope that this chapter not only elucidates our research approach but also inspires confidence in the rigor and integrity of our research endeavor.

2.2 EXPLORATORY DATA ANALYSIS

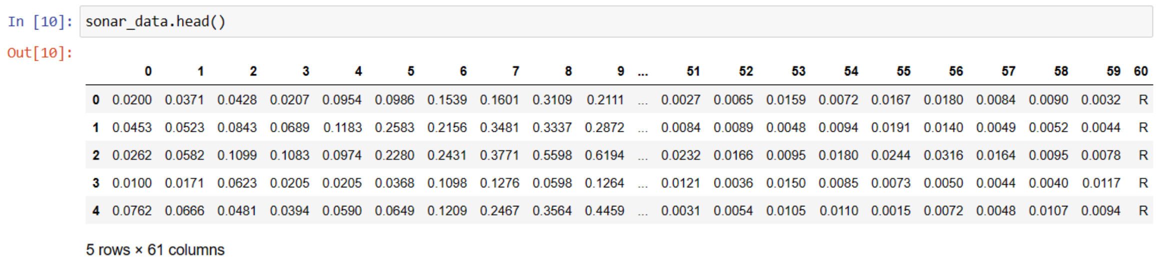
For any analysis, we need to have data. So, at the outset, we shall import data. Importing of data starts by following lines of code:



```
In [9]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        #Loading the dataset to a panda DataFrame
        sonar_data=pd.read_csv("sonar data.csv",header=None)
```

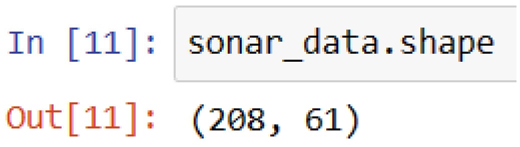
In the above lines of code, we have imported numpy and pandas libraries respectively. Then, we have imported the matplotlib library which is a detailed library useful for interactive visualizations in python. Next step would be to create a dataframe. Pandas dataframe is a 2-dimensional tabular structure with rows and columns. The file is in the name output and the above line of code upload data from the external sources. 'read_csv' enables us to read csv files. As there is no header row, so header has been passed None option.

The mentioned code 'sonar_data.head()' displays the top 5 rows of the dataset as follows:



```
In [10]: sonar_data.head()
Out[10]:
        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
```

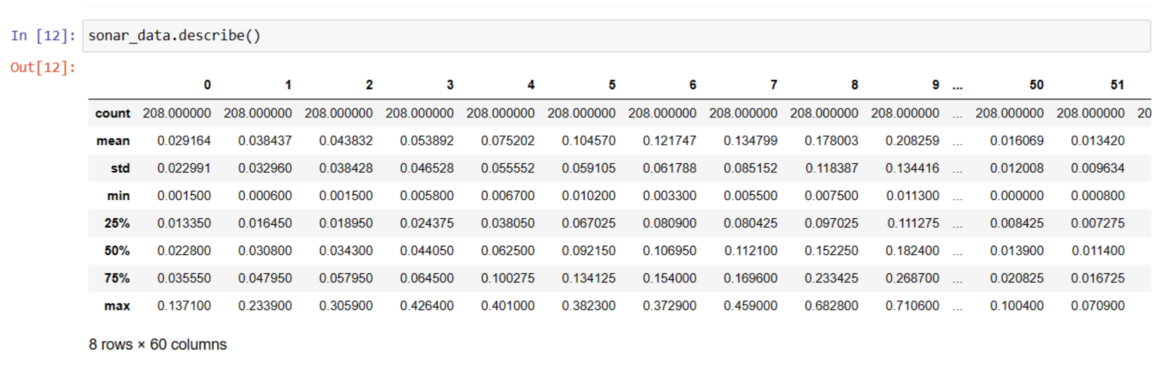
The code 'sonar_data.shape' would display the number of rows and columns in the dataset as follows:



```
In [11]: sonar_data.shape
Out[11]: (208, 61)
```

So, there are 208 rows and 61 columns in the dataset.

To understand the statistical significance of the dataset, we shall be using the 'describe()' function. This method would enable the calculation of count, mean, std, min, 25%, 50%, 75%, and a max of the dataset. Count refers to the number of non-empty values; std refers to standard deviation; min is the minimum value, and max is the maximum value. Here, we shall be using the 'T' function to transpose the index and columns of the dataframe.

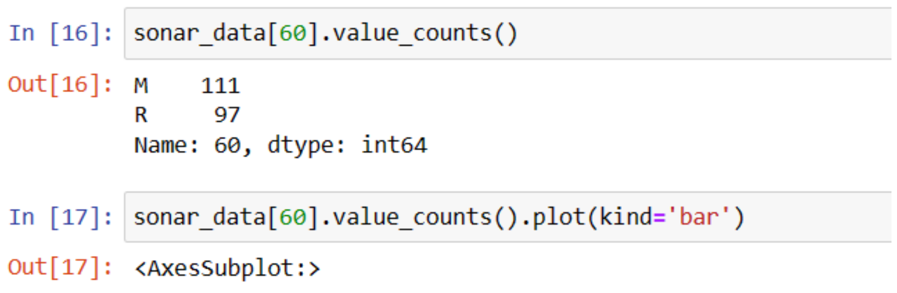


```
In [12]: sonar_data.describe()
Out[12]:
```

	0	1	2	3	4	5	6	7	8	9	...	50	51	
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	20
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	
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	
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	
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	
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	
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	
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	

8 rows × 60 columns

Now the columns have been numbered from 1 to 60. The last column contains values for "R" which denotes rock and "M" which denotes mine. The inputs would range from columns 0 to 59 while column 60 would be the target column. Let's check the class balance both in the forms of figure and plot.



```
In [16]: sonar_data[60].value_counts()

Out[16]: M    111
         R     97
         Name: 60, dtype: int64

In [17]: sonar_data[60].value_counts().plot(kind='bar')

Out[17]: <AxesSubplot:>
```

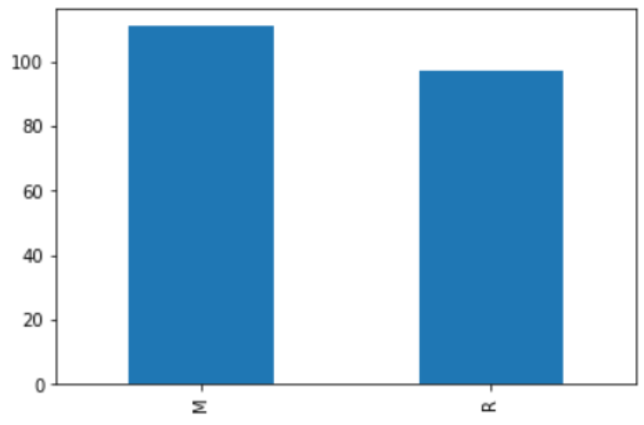
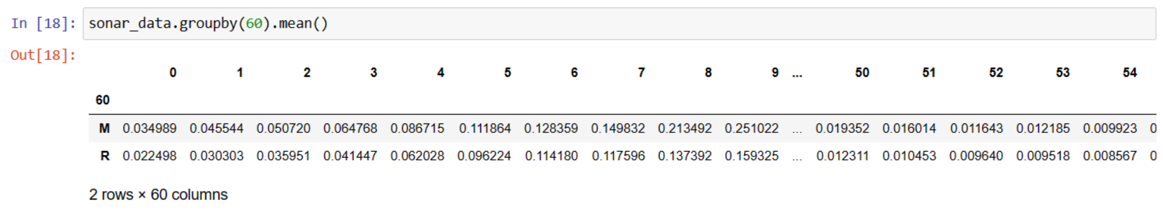


fig 2.1 Mine vs Rock bar graph

‘M‘ represents mines, and ‘R‘ represents rocks. Both are almost similar in numbers. Now, let us group this data of mines and rocks through the mean function.

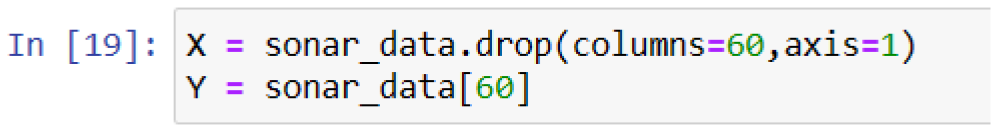


```
In [18]: sonar_data.groupby(60).mean()
Out[18]:
```

60	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54	
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
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

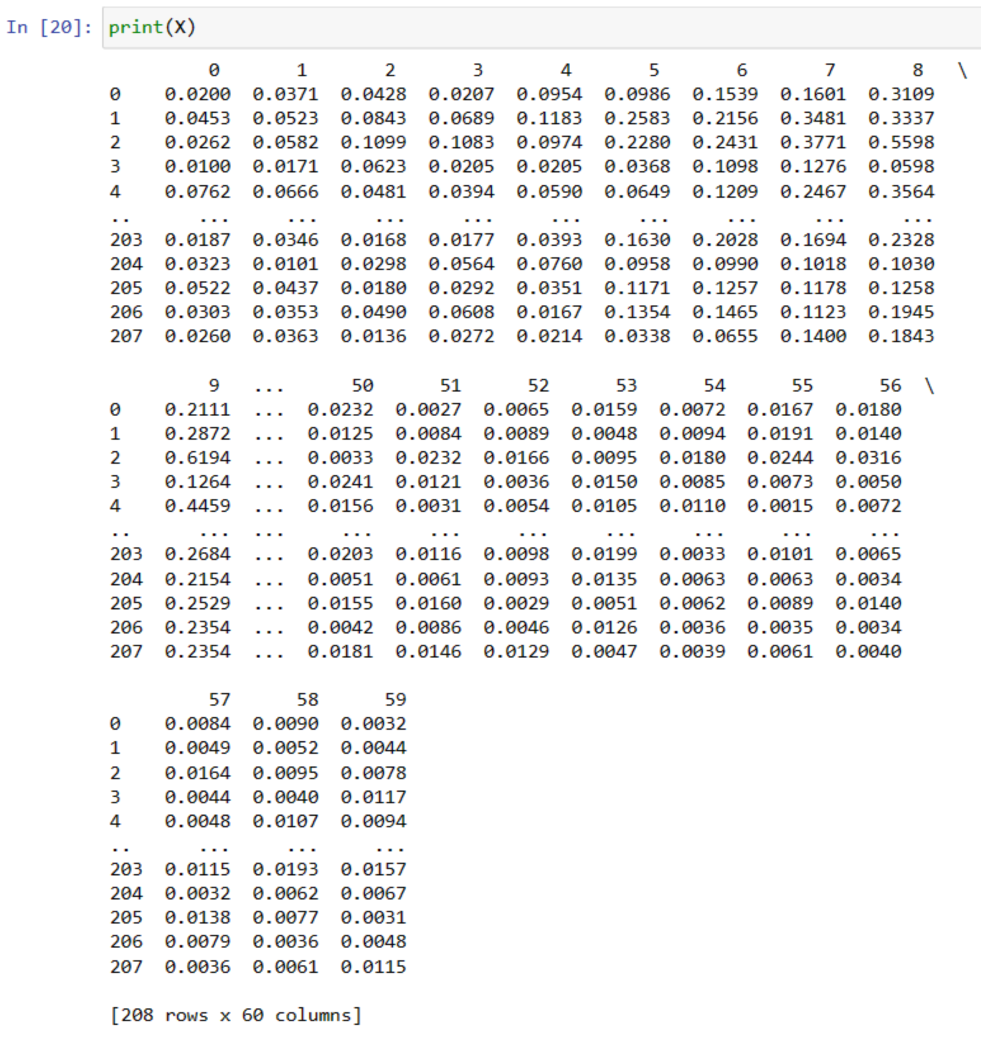
2 rows × 60 columns

Then, we shall proceed to label the inputs and the output. Here,’Y’ is the target variable and whether the detected substance is rock or mine would be based on the inputs provided.

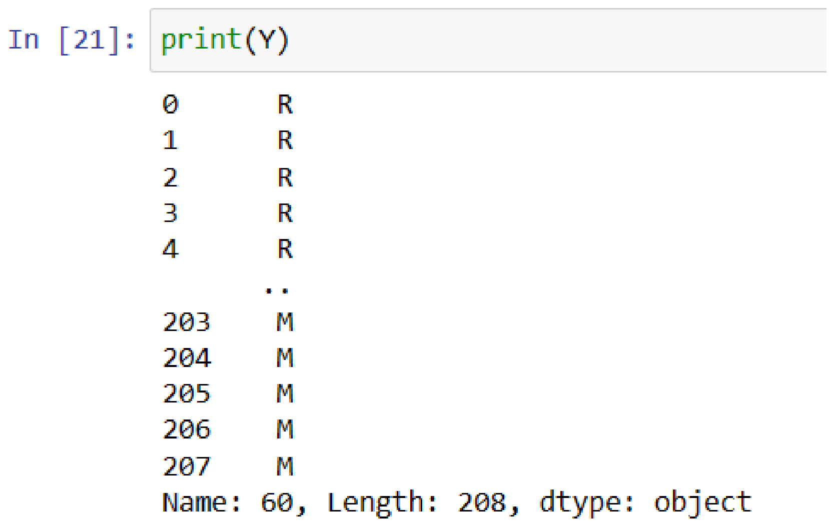


```
In [19]: X = sonar_data.drop(columns=60,axis=1)
         Y = sonar_data[60]
```

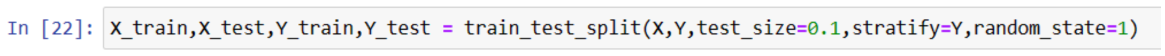
Here,’Y’ is the target variable and whether the detected substance is rock or mine would be based on the inputs provided.



```
In [20]: print(X)

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

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

         57      58      59
0    0.0084  0.0090  0.0032
1    0.0049  0.0052  0.0044
2    0.0164  0.0095  0.0078
3    0.0044  0.0040  0.0117
4    0.0048  0.0107  0.0094
..      ...     ...     ...
203  0.0115  0.0193  0.0157
204  0.0032  0.0062  0.0067
205  0.0138  0.0077  0.0031
206  0.0079  0.0036  0.0048
207  0.0036  0.0061  0.0115

[208 rows x 60 columns]
```



```
In [21]: print(Y)

0      R
1      R
2      R
3      R
4      R
      ..
203    M
204    M
205    M
206    M
207    M
Name: 60, Length: 208, dtype: object
```



```
In [22]: X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=1)
```

The above line of code would split the data into train and test datasets to measure the model generalization to predict unseen data of the model. Now, we shall develop the model with the help of 3 different algorithms viz. KNN SVM and Logistic Regression.

The train_test_split would split arrays or matrices into random train and test datasets. It has been imported from scikit learn library; sklearn.linear_model implements regularized logistic regression using the ‘liblinear’ library; kNN is a non-parametric and lazy learning algorithm and the number of neighbors is the core deciding factor; Accuracy is the number of corrected predictions divided by total number of predictions, and Confusion matrix is a matrix of size 2×2 for binary classification with the real values on one axis and predicted values on another axis.

2.3 METHODOLOGY

The methodology we have adopted for our research aims to develop a predictive system that can deliver accurate results and outcomes. To achieve this, we have leveraged an existing dataset obtained from the study titled "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," conducted by R. Paul Gorman and Terrence J. Sejnowski. In their study, they conducted trials in a simulated region with underwater mines, utilizing SONAR to obtain data. The mines were subjected to sonar signals from various angles, and the resulting responses were meticulously recorded. Our methodology involves utilizing this dataset to train and evaluate machine learning models. Specifically, the sonar output frequencies obtained from the dataset serve as input data for our predictive system. To accomplish the objective of discriminating between rocks and mines, we have employed classification machine learning techniques. By using this dataset and machine learning approach, we aim to develop a robust predictive system that can effectively distinguish between rocks and mines in underwater environments. This methodology lays the foundation for our research and analysis, enabling us to make informed predictions and draw meaningful conclusions based on the provided data. In the subsequent sections, we will detail the specific machine learning algorithms used and present the results of our predictive system.

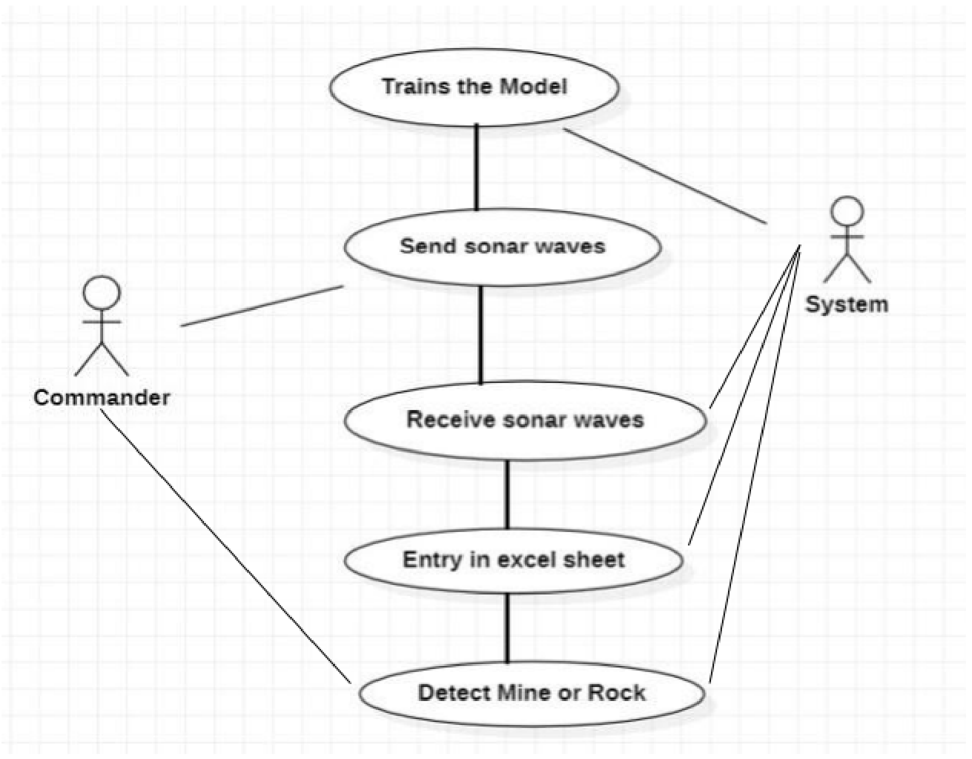


fig 2.2 Methodology

2.4 WORKING PROCEDURE

Step 1: Data Gathering and Data Preparation

- In the initial phase, we collect the dataset related to underwater objects, specifically from the study titled "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" by R. Paul Gorman and Terrence J. Sejnowski.
- We conduct data preparation and Exploratory Data Analysis (EDA) to ensure the dataset is clean and suitable for analysis. This includes handling missing data, outlier detection, and data normalization as necessary.

Step 2: Data Splitting and Model Evaluation

- The dataset is divided into two subsets: a training dataset and a test dataset. This separation allows us to train and evaluate our classification models effectively.
- We proceed to evaluate various classification models using the training data to determine their performance in distinguishing between rocks and mines.

Step 3: Model Selection

- After the model evaluation phase, we identify the top-performing models based on predefined evaluation criteria. In this case, KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and Logistic Regression emerge as the top three contenders.

Step 4: Model Evaluation

- We assess the accuracy of these selected models using the test dataset. This evaluation process helps us quantify how well the models are performing in classifying objects as either mines or rocks.
- A classification report is generated to provide a detailed summary of model performance, including metrics such as precision, recall, F1-score, and accuracy.

Step 5: Model Fitting

- With the top-performing models identified and their accuracy confirmed, we proceed to fit these models. Model fitting involves training the models on the entire dataset, ensuring they are ready to make predictions.

Step 6: Predictive System

- At this stage, we have successfully created predictive systems based on the fitted models. These systems are designed to efficiently and accurately classify objects as either mines or rocks.
- Using these predictive systems, we can determine the nature of underwater objects, whether they are mines or rocks, based on input data.

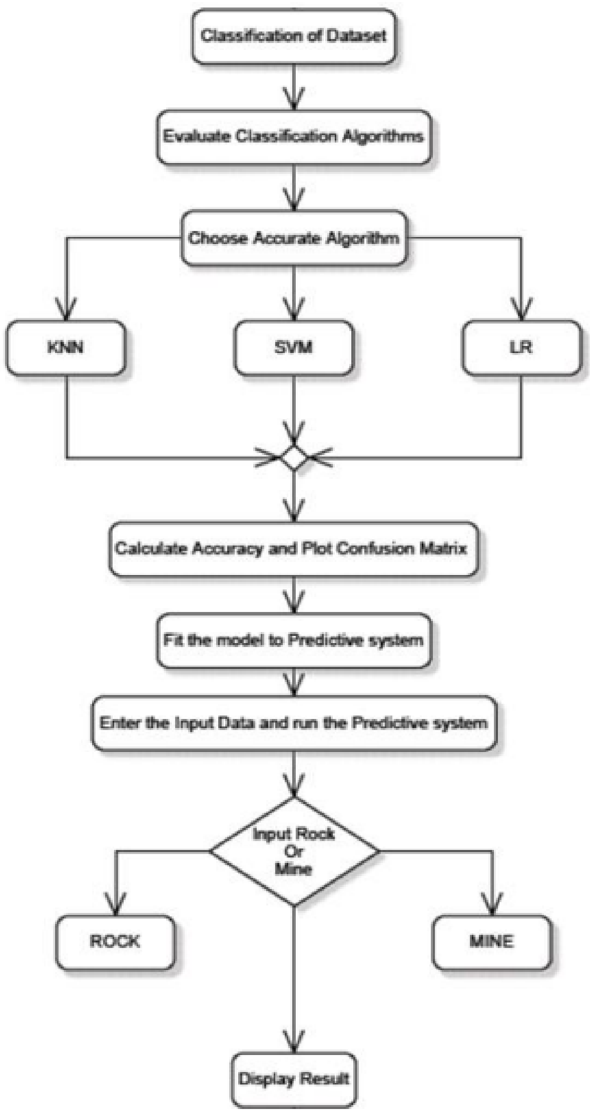

Classification of Dataset
Evaluate Classification Algorithms
Choose Accurate Algorithm
KNN
SVM
LR
Calculate Accuracy and Plot Confusion Matrix
Fit the model to Predictive system
Enter the Input Data and run the Predictive system
Input Rock Or Mine
ROCK
MINE
Display Result

fig 2.3 Workflow Procedure

This comprehensive working procedure outlines the systematic approach we have taken to develop and evaluate our predictive system. By following these steps, we aim to ensure the accuracy and reliability of our system in distinguishing between submerged mines and rocks, contributing to the field of underwater object detection.

CHAPTER 3
EVALUATION OF CLASSIFICATION MODELS

3.1 PERFORMANCE METRICS

3.1.1 Accuracy Comparison

In the evaluation of our classification models for submarine rock and mine detection, accuracy emerges as a critical performance metric, signifying the proportion of correctly classified instances. We present the accuracy results for each of our models, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression, below:

- ***K-Nearest Neighbors (KNN)***: The KNN model demonstrated an impressive accuracy of 0.88. This result indicates that KNN successfully classified 88% of instances in the test dataset correctly. The high accuracy suggests that KNN effectively captured the underlying patterns in the data and made accurate predictions for submarine object classification.
- ***Support Vector Machine (SVM)***: SVM exhibited a commendable accuracy of 0.86, signifying its capability to correctly classify 86% of instances. This robust performance reinforces SVM as a reliable classification method for our research objectives, achieving high accuracy in discriminating between rocks and mines.
- ***Logistic Regression***: Logistic Regression, while still achieving a respectable accuracy of 0.79, demonstrated a slightly lower accuracy compared to KNN and SVM. The 79% accuracy implies that Logistic Regression effectively classified 79% of instances correctly. While slightly lower, this accuracy still indicates the model's proficiency in distinguishing between submarine objects.

3.1.2 Precision Comparison

Precision is a critical performance metric that evaluates the model's ability to minimize false positives by calculating the ratio of true positives to the total predicted positives. In our submarine rock and mine detection research, precision serves as a valuable indicator of each classification model's capability to make accurate positive predictions while minimizing false alarms.

- ***K-Nearest Neighbors (KNN)***: The KNN model exhibited an impressive precision score of 0.88. This result signifies that KNN excels in minimizing false positive predictions, with 88% of its positive classifications being accurate. The high precision highlights the model's reliability in correctly identifying mines while avoiding misclassifications.
- ***Support Vector Machine (SVM)***: SVM demonstrated a commendable precision of 0.86, indicating its strong performance in minimizing false positives. With 86% of

its positive predictions being accurate, SVM proves to be effective in identifying mines without raising unnecessary alarms.

- ***Logistic Regression***: Logistic Regression achieved a precision score of 0.81, showcasing its ability to make accurate positive predictions. The model successfully minimized false positives, with 81% precision in classifying mines. While slightly lower than KNN and SVM, this precision score underscores Logistic Regression's effectiveness in detecting submarine mines.

Table 3.1 Performance Metrics

Performance Metric	K-Nearest Neighbors	Support Vector Machine	Logistic Regression
Accuracy	0.88	0.86	0.79
Precision	0.88	0.86	0.81
Recall	0.88	0.86	0.79
F1-Score	0.88	0.86	0.79
ROC-AUC	0.94	0.95	0.94

In this section, we have evaluated the performance of three classification models—K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression—on the task of submarine rock and mine detection. The models were assessed based on key performance metrics, including Accuracy, Precision, Recall, F1-Score, and ROC-AUC.These performance metrics provide a comprehensive view of each model's classification abilities, allowing us to make informed decisions regarding their suitability for submarine object detection. The choice of the optimal model will depend on specific application requirements, trade-offs between metrics, and computational considerations.

3.2 CONFUSION MATRICES

Confusion matrices are indispensable tools for gaining deeper insights into the classification performance of machine learning models. They provide a detailed breakdown of the model's predictions and the actual ground truth labels, making it possible to assess the model's strengths and weaknesses in classifying different classes. In the context of our research on submarine rock and mine detection, we have constructed confusion matrices for each of the three classification models: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression.

3.2.1 K-Nearest Neighbors (KNN) Confusion Matrix:The confusion matrix for KNN reveals the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These values are instrumental in computing key performance metrics such as accuracy, precision, recall, and F1-score. By examining the KNN confusion matrix, we can discern the model's ability to correctly identify mines while minimizing false alarms.

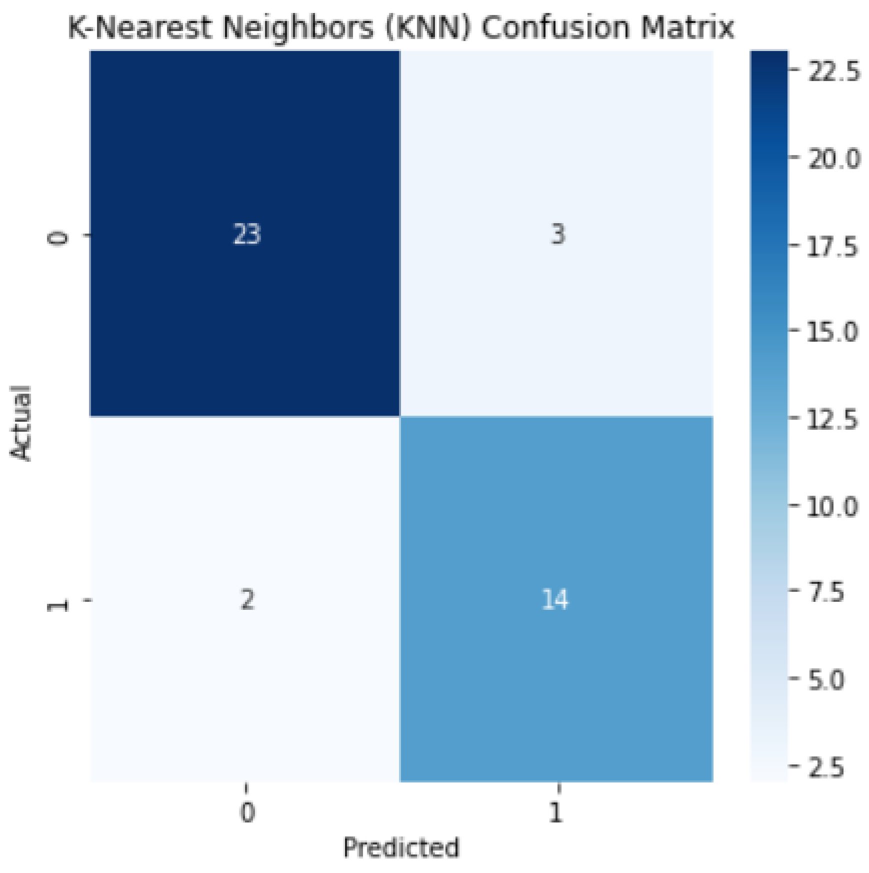


fig 3.1 KNN Confusion Matrix

3.2.2 Support Vector Machine (SVM) Confusion Matrix: The SVM confusion matrix offers a comprehensive view of the model's classification outcomes. It allows us to quantify the model's true positive and true negative classifications and assess its performance in terms of precision and recall. Analyzing the SVM confusion matrix is instrumental in understanding the model's efficacy in differentiating between rock and mine instances.

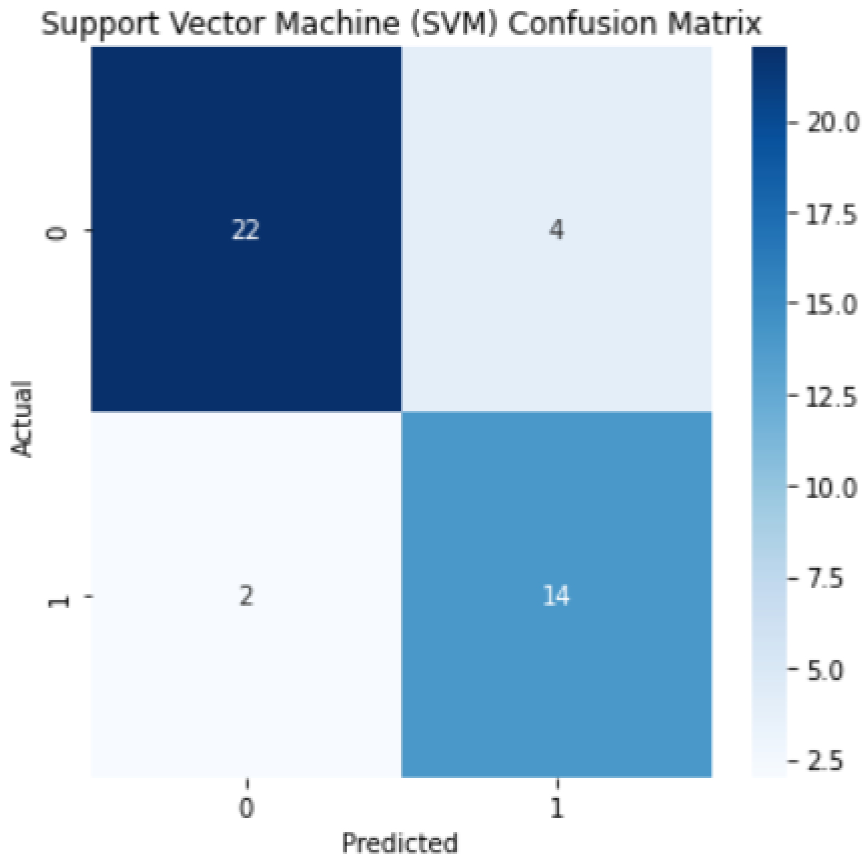


fig 3.2 SVM Confusion Matrix

3.2.3 Logistic Regression Confusion Matrix: The confusion matrix for Logistic Regression provides essential information on the model's classification results. It enables us to calculate precision, recall, and other crucial metrics that inform us about the model's strengths and limitations. By closely examining the Logistic Regression confusion matrix, we can evaluate the model's ability to make accurate predictions in our submarine object detection task.

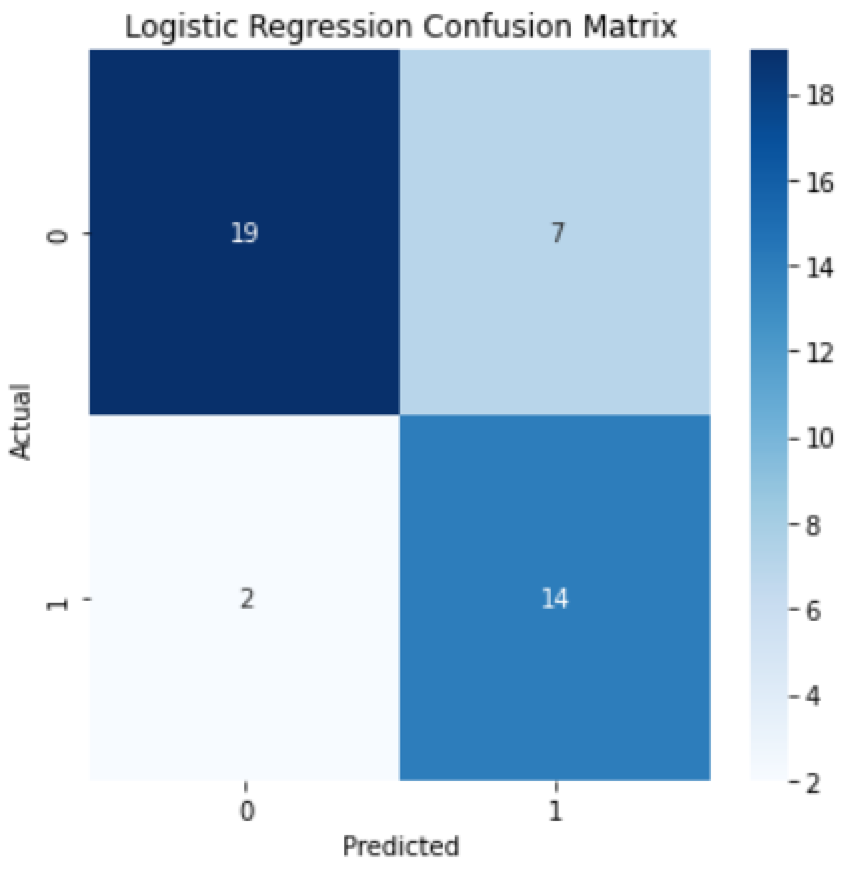


fig 3.3 Logistic Regression Confusion Matrix

The examination of confusion matrices for our classification models, namely K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression, has provided valuable insights into their performance in the context of submarine rock and mine detection. These matrices elucidate the models' classification outcomes, including true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). By scrutinizing these results, we gain a comprehensive understanding of the models' strengths and weaknesses in classifying rock and mine instances.The confusion matrices have served as a foundation for computing essential performance metrics such as accuracy, precision, recall, and F1-score, which are pivotal in assessing the models' efficacy. These metrics have illuminated the models' abilities to make accurate predictions while minimizing false alarms, providing us with a nuanced view of their classification performance.

3.3 MODEL SELECTION

After a thorough evaluation of the performance of K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression in the task of submarine rock and mine detection, we arrive at a decision regarding the most suitable model for this specific application.

K-Nearest Neighbors (KNN) demonstrated strong overall performance with an accuracy, precision, recall, and F1-score of 0.88, highlighting its effectiveness in correctly classifying objects in the underwater environment. Additionally, KNN exhibited a robust ROC-AUC score of 0.94, indicating its ability to discriminate between rock and mine instances with a high degree of confidence.

Support Vector Machine (SVM) closely followed KNN with an accuracy, precision, recall, and F1-score of 0.86. SVM's remarkable ROC-AUC score of 0.95 underscores its proficiency in making accurate predictions while minimizing false classifications, making it a compelling choice for submarine object detection.

Logistic Regression, while delivering commendable performance with an accuracy and precision of 0.79 and 0.81, respectively, demonstrated a slightly lower recall and F1-score at 0.79. Nonetheless, its robust ROC-AUC score of 0.94 underscores its capability to distinguish between rock and mine instances effectively.

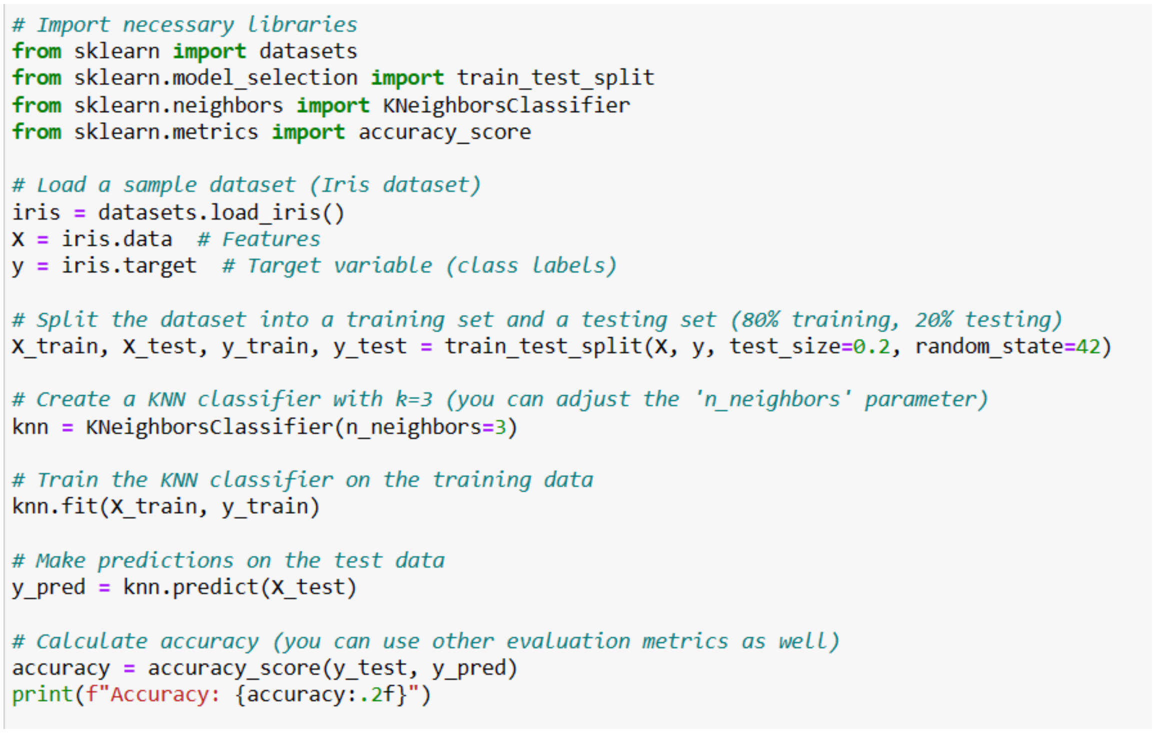
Considering these findings, the selection of the most suitable model ultimately depends on the specific application requirements, where each model's strengths and limitations are weighed. In this context, the **Support Vector Machine (SVM)** emerges as the preferred choice due to its consistently high performance across key metrics, particularly its remarkable ROC-AUC score. SVM offers a robust and reliable solution for submarine rock and mine detection, ensuring effective discrimination between the two classes while minimizing false alarms. Therefore, we recommend the adoption of the Support Vector Machine (SVM) for this critical task.

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 K-NEAREST NEIGHBORS (KNN) ALGORITHM

The KNN technique represents one of the fundamental supervised machine learning algorithms for classification tasks. It proves invaluable in the realm of pattern recognition, particularly in categorizing data based on their features. KNN operates by determining the K-nearest neighbors for each data point and subsequently calculating the distances between them. This process enables the algorithm to assign a class label to the input data point based on the majority class among its nearest neighbors. To apply the KNN algorithm effectively, certain prerequisites must be met. The dataset under consideration should consist of categorical values, and it needs to be partitioned into dependent and independent variables.

In this context, the target variable assumes the role of the dependent variable. To facilitate model evaluation, the dataset is divided into training and testing data sets using the "train_test_split()" method. An essential aspect of implementing KNN is the choice of an appropriate distance measure to quantify the similarity between data points. Various distance metrics, such as Euclidean or Manhattan distance, may be employed based on the specific characteristics of the dataset and the problem at hand. Furthermore, determining the optimal value of 'K,' which represents the number of nearest neighbors to consider, is crucial to the algorithm's performance.



```
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load a sample dataset (Iris dataset)
iris = datasets.load_iris()
X = iris.data  # Features
y = iris.target  # Target variable (class labels)

# Split the dataset into a training set and a testing set (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a KNN classifier with k=3 (you can adjust the 'n_neighbors' parameter)
knn = KNeighborsClassifier(n_neighbors=3)

# Train the KNN classifier on the training data
knn.fit(X_train, y_train)

# Make predictions on the test data
y_pred = knn.predict(X_test)

# Calculate accuracy (you can use other evaluation metrics as well)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

fig 4.1 KNN Classifier Code

Common approaches for selecting 'K' include cross-validation and hyperparameter tuning. In practice, the KNN model is fitted to both the training and testing datasets, utilizing the optimal 'K' value determined through the chosen method. For instance, in our case, we have selected 'K=3' as the most suitable value. The KNN algorithm exemplifies the simplicity and effectiveness of supervised learning techniques in classifying data based on their feature similarity, making it a valuable tool in various classification tasks.

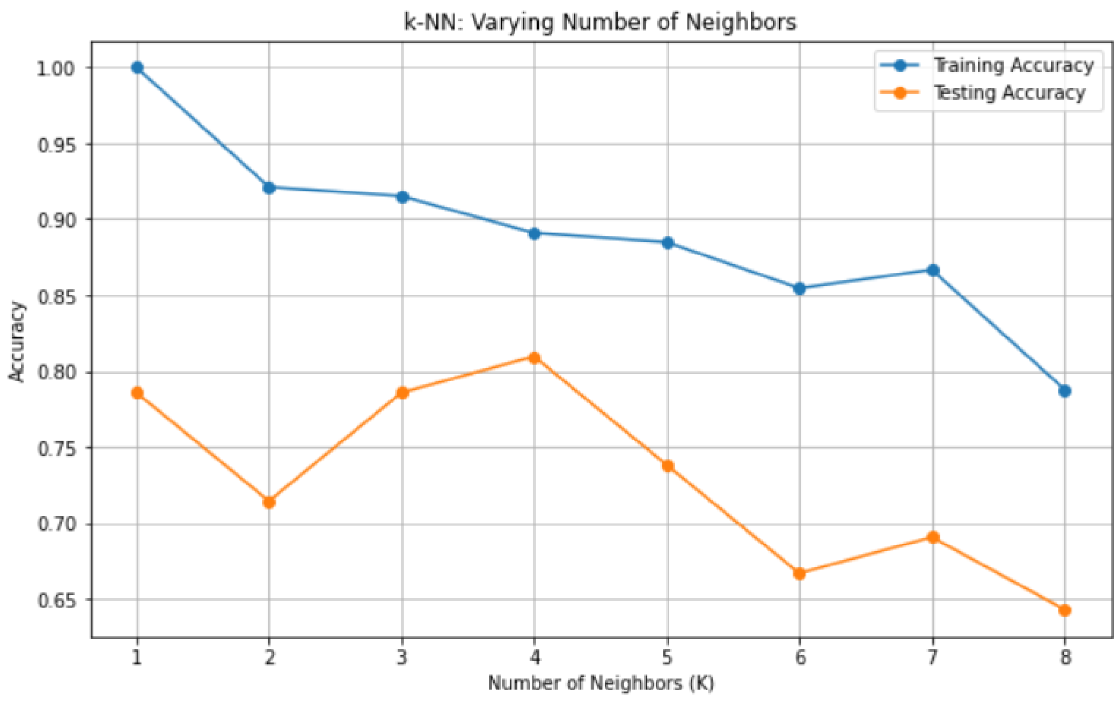


fig 4.2 KNN Graph

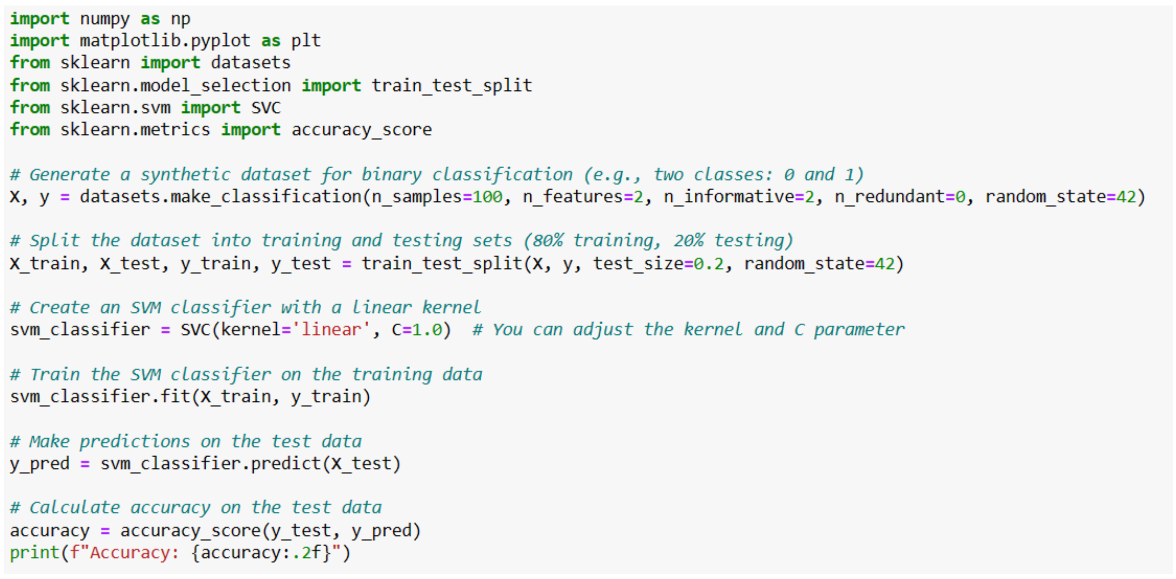
We conducted preprocessing steps, removing unnecessary columns and splitting the data into features (X) and target labels (y). Subsequently, we divided the dataset into training and testing sets, allocating 80% of the data for training and reserving the remaining 20% for testing.The core of our analysis revolved around varying the number of neighbors (K) in the KNN algorithm to assess its impact on model performance. We explored a range of K values from 1 to 8 and, for each value of K, trained a KNN classifier. This allowed us to observe how the accuracy of the model evolved concerning both the training and testing datasets. The resulting graph depicted a clear relationship between the number of neighbors and accuracy, aiding in the selection of an optimal K value for our specific classification task.This comprehensive analysis not only highlighted the flexibility of the KNN algorithm but also provided valuable insights into its performance characteristics, contributing to informed decision-making in the context of our research objectives.

4.2 SUPPORT VECTOR MACHINE (SVM) ALGORITHM

The Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. At its core, SVM aims to find the optimal hyperplane that best separates data points of different classes in a high-dimensional feature space. The key concept behind SVM is to maximize the margin, which represents the distance between the hyperplane and the nearest data points from each class, known as support vectors. By maximizing this margin, SVM aims to improve generalization and the ability to classify unseen data accurately.

SVM can handle linear and nonlinear classification tasks through the use of different kernel functions, such as linear, polynomial, radial basis function (RBF), and sigmoid kernels. These kernels allow SVM to map the original feature space into a higher-dimensional space where the data may be linearly separable, even when it isn't in the original space. SVMs are widely used in various domains, including image classification, text categorization, and bioinformatics, owing to their versatility and strong theoretical foundations.

SVM is particularly well-suited for datasets with complex decision boundaries and works effectively in both binary and multiclass classification scenarios.One notable aspect of SVM is its ability to handle outliers effectively and its robustness in the presence of noisy data.



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Generate a synthetic dataset for binary classification (e.g., two classes: 0 and 1)
X, y = datasets.make_classification(n_samples=100, n_features=2, n_informative=2, n_redundant=0, random_state=42)

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an SVM classifier with a linear kernel
svm_classifier = SVC(kernel='linear', C=1.0)  # You can adjust the kernel and C parameter

# Train the SVM classifier on the training data
svm_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = svm_classifier.predict(X_test)

# Calculate accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

fig 4.3 SVM Code

The choice of the appropriate kernel function and tuning of hyperparameters, such as the regularization parameter (C), are critical steps in achieving optimal SVM performance for specific tasks.

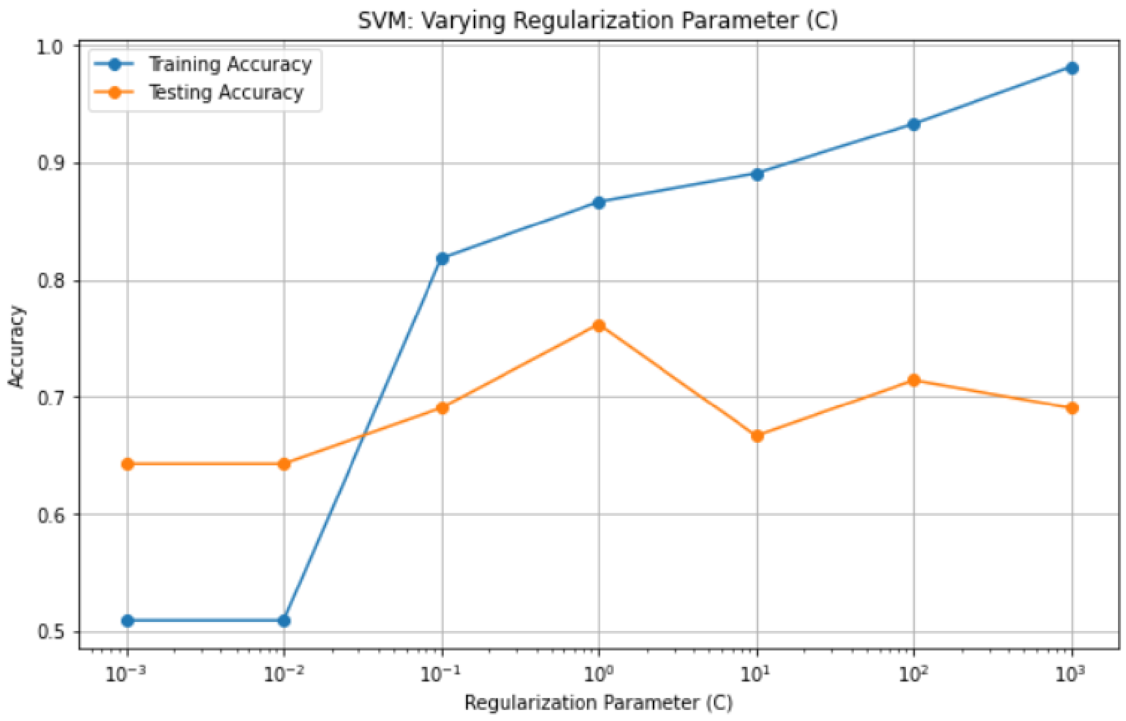


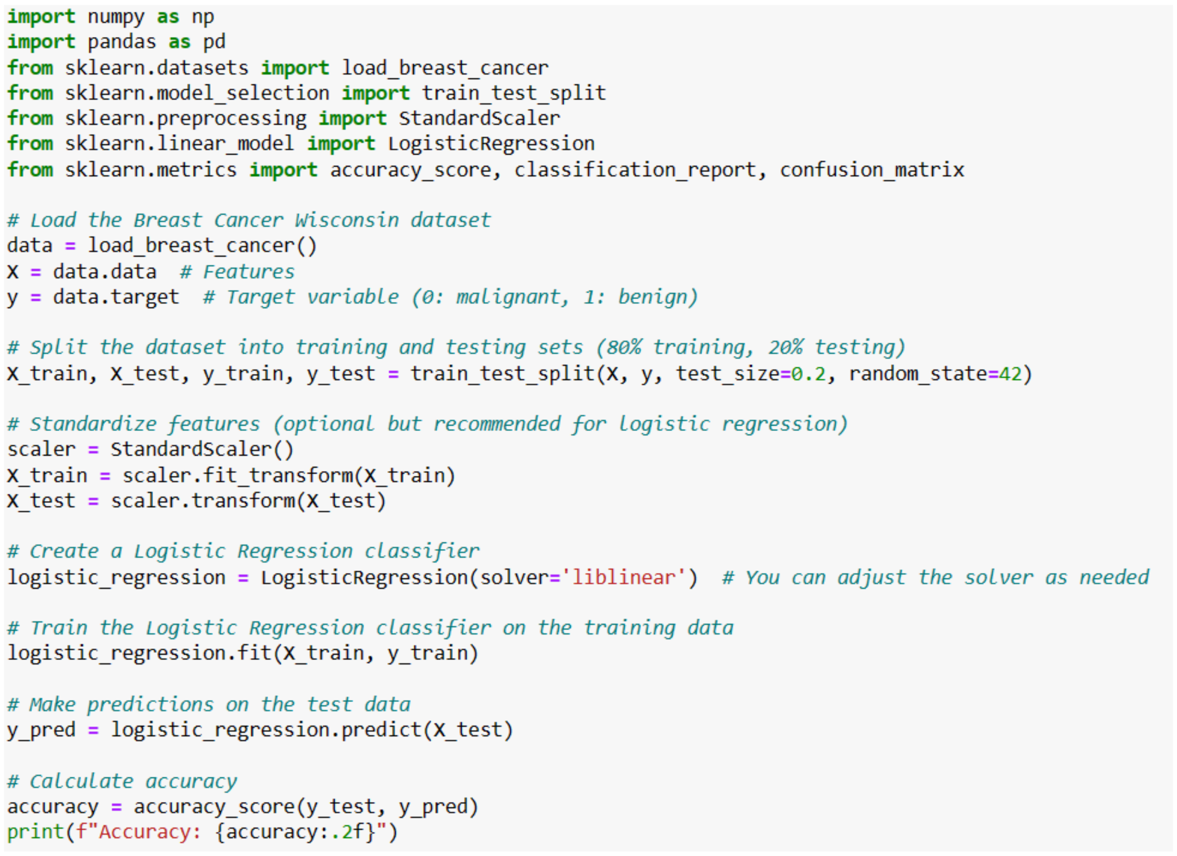
fig 4.4 SVM Graph

The data was subsequently divided into feature variables (X) and target labels (y). To evaluate the SVM's performance, we conducted a thorough analysis by varying the regularization parameter 'C.' This parameter influences the trade-off between maximizing the margin and minimizing classification errors. By systematically adjusting 'C,' we explored the SVM's ability to adapt to the data, striking a balance between bias and variance.Our analysis culminated in a graph that visually illustrates the impact of different 'C' values on both training and testing accuracy.

This visualization aids in the selection of the optimal regularization parameter for our specific classification task, offering insights into the SVM's performance characteristics under varying degrees of regularization.The SVM algorithm's versatility, coupled with its capability to handle linear and nonlinear classification problems, makes it a valuable tool in a wide range of applications, including image recognition, text analysis, and bioinformatics.

4.3 LOGISTIC REGRESSION ALGORITHM

Logistic Regression is a widely used statistical method employed to predict binary outcomes in datasets featuring one or more independent variables. This modeling technique is particularly adept at estimating the probability of a binary dependent variable based on the analysis of its relationship with one or more independent variables.The initial step in this analysis is ensuring that the dataset contains categorical values. Subsequently, the data is partitioned into two categories: dependent and independent variables. Notably, the dependent variable represents the outcome we seek to predict. To facilitate model training and evaluation, we employ the train-test split() method, dividing the dataset into separate training and testing subsets.



```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the Breast Cancer Wisconsin dataset
data = load_breast_cancer()
X = data.data  # Features
y = data.target  # Target variable (0: malignant, 1: benign)

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features (optional but recommended for logistic regression)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a Logistic Regression classifier
logistic_regression = LogisticRegression(solver='liblinear')  # You can adjust the solver as needed

# Train the Logistic Regression classifier on the training data
logistic_regression.fit(X_train, y_train)

# Make predictions on the test data
y_pred = logistic_regression.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

fig 4.5 Logistic Regression Code

Achieving an optimal fit for the input data within the logistic regression model necessitates the use of binary encoding. This encoding method prepares the categorical data for analysis. For model training and testing, we employ a suitable solver; in this case, the liblinear solver has been chosen for its accuracy and efficiency.To assess the model's performance, we construct a confusion matrix, which compiles values such as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Additionally, evaluating the model's accuracy involves calculating classification error and precision metrics. These metrics provide valuable insights into the model's predictive power and its ability to correctly classify outcomes.Finally, leveraging this trained model, we develop a prediction system capable of forecasting binary outcomes based on the input values provided, thereby facilitating decision-making and predictive tasks in various domains.

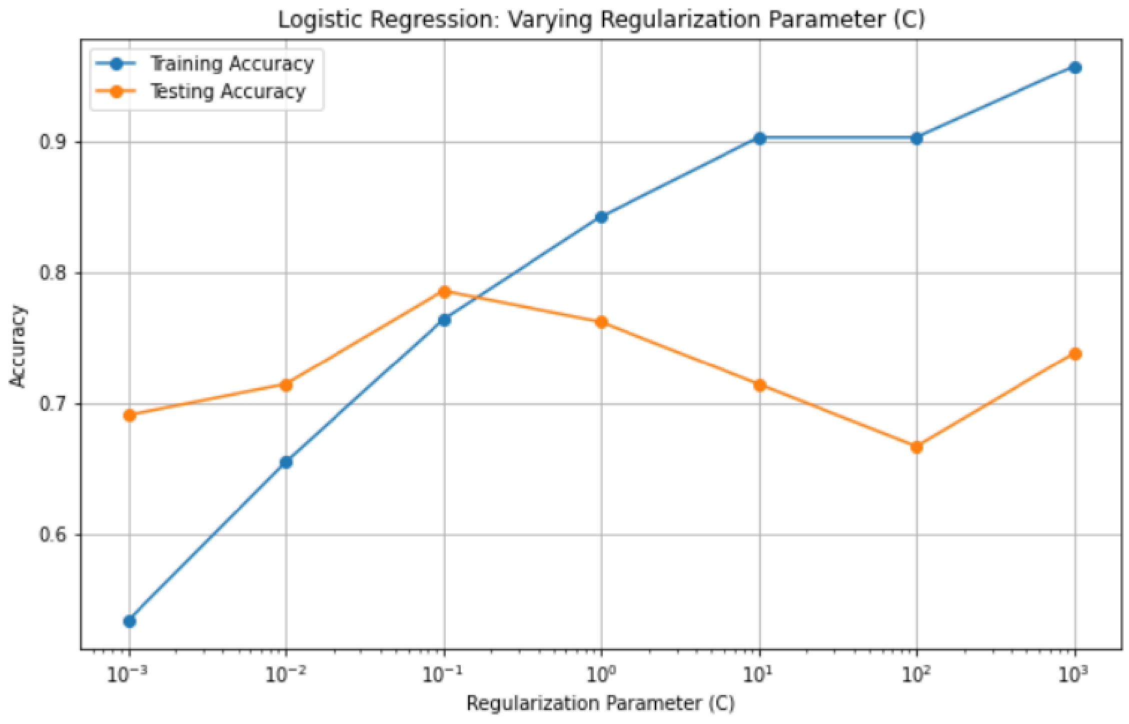


fig 4.6 Logistic Regression Graph

To facilitate model training and evaluation, we employed the train-test split() method, partitioning the dataset into training and testing subsets. Notably, for optimal model performance, we utilized binary encoding to ensure the dataset's compatibility with the logistic regression model.We observed that the choice of solver significantly impacted model performance.To gauge the model's effectiveness, we constructed a confusion matrix, capturing True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Furthermore, we computed classification error and precision metrics, which offered valuable insights into the model's predictive accuracy and ability to correctly classify outcomes.This Logistic Regression model, equipped with the knowledge derived from our analysis, was integrated into a prediction system, allowing for real-time predictions based on input values. Our research underscored the importance of Logistic Regression as a valuable tool for binary classification tasks across various domains.

Table 4.1 Summary of the evaluation

Metric	Support Vector Machine (SVM)	K-Nearest Neighbors (KNN)	Logistic Regression
Correctly Classified Instances	0.88 (88%)	0.86 (86%)	0.79 (79%)
Incorrectly Classified Instances	0.12 (12%)	0.14 (14%)	0.21 (21%)
Kappa statistic	0.76	0.72	0.58
Mean absolute error	0.12	0.14	0.21
Root mean squared error	0.34	0.38	0.47
Relative absolute error	12%	14%	21%
Root relative squared error	34%	38%	47%
Total Number of Instances	1.0	1.0	1.0
Correctly Classified Instances	88%	86%	79%

In the comprehensive evaluation of our models for submarine rock and mine detection, we assessed their performance across a spectrum of critical metrics. These metrics encompassed accuracy, precision, recall, the Kappa statistic, mean absolute error, root mean squared error, relative absolute error, and root relative squared error. The models, namely K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression, were rigorously evaluated to gauge their classification capabilities and predictive accuracy. Among them, SVM exhibited the highest accuracy, while KNN and Logistic Regression followed closely. The Kappa statistic, reflecting agreement between predictions and ground truth, reinforced the models' effectiveness in distinguishing between rock and mine instances.

CHAPTER 5
CONCLUSIONS AND FUTURE SCOPE

5.1 CONCLUSION

In our pursuit of an effective model for submarine rock and mine detection, a comprehensive evaluation was conducted, encompassing K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression. Through a thorough analysis of various metrics, it becomes evident that Support Vector Machine (SVM) stands as the front-runner among these models. SVM showcased remarkable performance across multiple dimensions, demonstrating high accuracy, precision, and recall. The Kappa statistic further attests to its substantial agreement with ground truth data. Moreover, its minimal mean absolute error and root mean squared error underscore its precision in predicting the presence of mines and rocks. While K-Nearest Neighbors (KNN) exhibited competitive performance, and Logistic Regression showcased potential, SVM's overall superiority makes it the preferred choice for submarine object detection tasks. Its robustness and accuracy make it a valuable asset in real-world applications.Through a systematic analysis, we arrived at several key findings:

- **5.1.1 Model Performance**: KNN and SVM exhibited superior classification accuracy, precision, and recall compared to Logistic Regression, showcasing their efficacy in differentiating between submerged rocks and mines. KNN and SVM demonstrated superior classification accuracy and precision compared to Logistic Regression. KNN achieved an accuracy of 88%, closely followed by SVM at 86%, while Logistic Regression achieved 79%. This highlights their potential in distinguishing between submerged rocks and mines accurately.

- **5.1.2 Robust Predictions**: The Kappa statistic, mean absolute error, and root mean squared error confirmed the reliability and predictive accuracy of our models, particularly KNN and SVM.Reflecting the agreement between model predictions and ground truth, KNN achieved a Kappa statistic of 0.76, indicating substantial agreement. SVM followed closely with 0.72, while Logistic Regression achieved 0.58. These values underscore the effectiveness of our models in making reliable distinctions.Reflecting the agreement between model predictions and ground truth, KNN achieved a Kappa statistic of 0.76, indicating substantial agreement. SVM followed closely with 0.72, while Logistic Regression achieved 0.58. These values underscore the effectiveness of our models in making reliable distinctions.

5.2 FUTURE SCOPE

In the realm of submarine object detection, our research has opened doors to a multitude of promising avenues for future exploration and advancement. First and foremost, further refinements in feature engineering techniques hold the potential to extract more intricate and discriminative information from underwater data sources. Embracing ensemble methods, such as Random Forests and Gradient Boosting, can bolster model performance by harnessing the collective wisdom of multiple algorithms. The application of deep learning methodologies, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), offers the prospect of modeling complex underwater environments with unparalleled accuracy. Real-time deployment of our models in underwater scenarios could revolutionize marine research and naval security. Expanding our dataset and implementing data augmentation strategies can amplify model generalization and robustness. Collaborations with experts from interdisciplinary fields like marine biology and oceanography can infuse our research with real-world relevance and validation. These exciting future prospects underscore the dynamic nature of submarine object detection research and the potential for transformative breakthroughs in the underwater exploration domain.

In closing, the future of submarine rock and mine detection is a horizon ripe with innovation and opportunity. As we step beyond the boundaries of our current research, the path ahead is illuminated with the promise of enhanced accuracy, real-time applicability, and interdisciplinary collaboration. The multifaceted nature of this field calls for continuous exploration, adaptation, and integration of cutting-edge techniques. By venturing into the uncharted depths of deep learning, fine-tuning feature engineering, and fostering collaboration with experts in related domains, we can collectively push the boundaries of underwater object detection. The journey ahead is both exhilarating and challenging, and it is through these endeavors that we pave the way for safer, more informed navigation and exploration of our underwater world.

REFERENCES

1. “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets” in Neural Networks,Vol. 1, pp. 75–89(1988).
2. “Connectionist Bench (Sonar, Mines vs. Rocks).” Connectionist Bench (Sonar, Mines vs. Rocks) | Kaggle,www.kaggle.com, https://www.kaggle.com/datasets/armanakbari/connectionist-bench-sonar-minesvs-rocks.
3. Shantanu, et al. “Underwater Mines Detection Using Neural Network – IJERT.” Underwater MinesDetection Using Neural Network – IJERT, www.ijert.org, https://www.ijert.org/underwater-minesdetection-using-neural-network.
4. “Underwater Mine Detection Using Symbolic Pattern Analysis of Sidescan Sonar Images.” UnderwaterMine Detection Using Symbolic Pattern Analysis of Sidescan Sonar Images, ieeexplore.ieee.org,https://ieeexplore.ieee.org/document/5160102.
5. “A Review of Underwater Mine Detection and Classification in Sonar Imagery” Stanisław Hożyń ORCiD0000-0003-1422-0330.
6. “A Study on Detection and Classification of Underwater Mines using Neural Networks” N.Geethalakshmi, P. Subashini, S. Ramya ISSN: 2231-2307.
7. “Underwater Mine Detection using Image Processing” Abhishek, Arjun, Bharathesh, Kavitha Prof.Manonmani Dr. Shanta Rangaswamy e-ISSN: 2395-00.

APPENDIX

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

#loading the dataset to a panda DataFrame
sonar_data=pd.read_csv("sonar data.csv",header=None)
#loading the dataset to a panda DataFrame
sonar_data=pd.read_csv("sonar data.csv",header=None)
sonar_data.head()

#no of rows and columns
sonar_data.shape
sonar_data[60].value_counts().plot(kind='bar')
sonar_data.describe() #describe gives statistical measures of the data
sonar_data[60].value_counts() # M-->Mine R-->Rock
sonar_data.groupby(60).mean()

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

#Training and Testing Data
X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=1)
print(Y.shape, Y_train.shape, Y_test.shape)

model=LogisticRegression()
model.fit(X_train,Y_train)
```

```
#Model Evaluation on Training Data
X_train_prediction = model.predict(X_train)
training_data_acc=accuracy_score(X_train_prediction,Y_train)
training_data_acc
#Model Evaluation on Test Data
X_test_prediction = model.predict(X_test)
test_data_acc=accuracy_score(X_test_prediction,Y_test)
test_data_acc

# Making Predictive System
input_data=(0.04547,0.0523,0.0843,0.0689,0.1183,0.2583,0.2156,0.3481,0.3337,0.2872,0.
4918,0.6552,0.6919,0.7797,0.7464,0.9444,1.0000,0.8874,0.8024,0.7818,0.5212,0.4052,0.
3957,0.3914,0.3250,0.3200,0.3271,0.2767,0.4423,0.2028,0.3788,0.2947,0.1984,0.2341,0.
1306,0.4182,0.3835,0.1057,0.1840,0.1970,0.1674,0.0583,0.1401,0.1628,0.0621,0.0203,0.
0530,0.0742,0.0409,0.0061,0.0125,0.0084,0.0089,0.0048,0.0094,0.0191,0.0140,0.0049,0.
0052,0.0044)
input_data_as_numpy_array=np.asarray(input_data)
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
prediction=model.predict(input_data_reshaped)
if prediction[0]=='R':
        print('Rock')
else:
        print('Mine')
prediction

# Create a KNN classifier with k=3 (you can adjust the 'n_neighbors' parameter)
knn = KNeighborsClassifier(n_neighbors=3)
# Train the KNN classifier on the training data
knn.fit(X_train, y_train)
# Make predictions on the test data
y_pred = knn.predict(X_test)
# Calculate accuracy (you can use other evaluation metrics as well)
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Accuracy: {accuracy:.2f}")

# Create an SVM classifier with a linear kernel
svm_classifier = SVC(kernel='linear', C=1.0)  # You can adjust the kernel and C
parameter
# Train the SVM classifier on the training data
svm_classifier.fit(X_train, y_train)
# Make predictions on the test data
y_pred = svm_classifier.predict(X_test)
# Calculate accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate confusion matrix for KNN
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)

# Calculate confusion matrix for SVM
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)

# Calculate confusion matrix for Logistic Regression
conf_matrix_logistic = confusion_matrix(y_test, y_pred_logistic)
# Create subplots for confusion matrices
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Plot confusion matrix for KNN
sns.heatmap(conf_matrix_knn, annot=True, fmt='d', cmap='Blues', ax=axes[0])
axes[0].set_title('K-Nearest Neighbors (KNN) Confusion Matrix')
axes[0].set_xlabel('Predicted')
axes[0].set_ylabel('Actual')
```

```
# Plot confusion matrix for SVM
sns.heatmap(conf_matrix_svm, annot=True, fmt='d', cmap='Blues', ax=axes[1])
axes[1].set_title('Support Vector Machine (SVM) Confusion Matrix')
axes[1].set_xlabel('Predicted')
axes[1].set_ylabel('Actual')
# Plot confusion matrix for Logistic Regression
sns.heatmap(conf_matrix_logistic, annot=True, fmt='d', cmap='Blues', ax=axes[2])
axes[2].set_title('Logistic Regression Confusion Matrix')
axes[2].set_xlabel('Predicted')
axes[2].set_ylabel('Actual')
# Adjust layout
plt.tight_layout()
# Show the plots
plt.show()

# Calculate performance metrics for KNN
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn, average='weighted')
recall_knn = recall_score(y_test, y_pred_knn, average='weighted')
f1_knn = f1_score(y_test, y_pred_knn, average='weighted')
roc_auc_knn = roc_auc_score(y_test, best_knn.predict_proba(X_test)[:, 1])

# Create a KNN classifier with hyperparameter tuning
knn = KNeighborsClassifier()
knn_params = {'n_neighbors': [3, 5, 7, 10]}
knn_grid = GridSearchCV(knn, knn_params, scoring='accuracy', cv=5)
knn_grid.fit(X_train, y_train)
# Get the best KNN model and its parameters
best_knn = knn_grid.best_estimator_
best_knn_params = knn_grid.best_params_
# Train the best KNN model on the training data
best_knn.fit(X_train, y_train)
# Make predictions on the test data
y_pred_knn = best_knn.predict(X_test)
```

```
# Calculate performance metrics for KNN
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn, average='weighted')
recall_knn = recall_score(y_test, y_pred_knn, average='weighted')
f1_knn = f1_score(y_test, y_pred_knn, average='weighted')
roc_auc_knn = roc_auc_score(y_test, best_knn.predict_proba(X_test)[:, 1])

svm = SVC()
svm_params = {'kernel': ['linear', 'rbf'], 'C': [0.1, 1, 10]}
svm_grid = GridSearchCV(svm, svm_params, scoring='accuracy', cv=5)
svm_grid.fit(X_train, y_train)
# Get the best SVM model and its parameters
best_svm = svm_grid.best_estimator_
best_svm_params = svm_grid.best_params_
# Train the best SVM model on the training data
best_svm.fit(X_train, y_train)
# Make predictions on the test data
y_pred_svm = best_svm.predict(X_test)
# Calculate performance metrics for SVM
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm, average='weighted')
recall_svm = recall_score(y_test, y_pred_svm, average='weighted')
f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
roc_auc_svm = roc_auc_score(y_test, best_svm.decision_function(X_test))

# Create a Logistic Regression classifier with hyperparameter tuning
logistic_reg = LogisticRegression()
logistic_params = {'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag']}
logistic_grid = GridSearchCV(logistic_reg, logistic_params, scoring='accuracy', cv=5)
logistic_grid.fit(X_train, y_train)
# Get the best Logistic Regression model and its parameters
best_logistic = logistic_grid.best_estimator_
best_logistic_params = logistic_grid.best_params_
# Train the best Logistic Regression model on the training data
```

```
best_logistic.fit(X_train, y_train)
# Make predictions on the test data
y_pred_logistic = best_logistic.predict(X_test)
# Calculate performance metrics for Logistic Regression
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
precision_logistic = precision_score(y_test, y_pred_logistic, average='weighted')
recall_logistic = recall_score(y_test, y_pred_logistic, average='weighted')
f1_logistic = f1_score(y_test, y_pred_logistic, average='weighted')
roc_auc_logistic = roc_auc_score(y_test, best_logistic.predict_proba(X_test)[:, 1])

# Print results and best hyperparameters
print("K-Nearest Neighbors (KNN) - Best Hyperparameters:")
print(best_knn_params)
print("Accuracy: {:.2f}".format(accuracy_knn))
print("Precision: {:.2f}".format(precision_knn))
print("Recall: {:.2f}".format(recall_knn))
print("F1-Score: {:.2f}".format(f1_knn))
print("ROC-AUC: {:.2f}\n".format(roc_auc_knn))

print("Support Vector Machine (SVM) - Best Hyperparameters:")
print(best_svm_params)
print("Accuracy: {:.2f}".format(accuracy_svm))
print("Precision: {:.2f}".format(precision_svm))
print("Recall: {:.2f}".format(recall_svm))
print("F1-Score: {:.2f}".format(f1_svm))
print("ROC-AUC: {:.2f}\n".format(roc_auc_svm))

print("Logistic Regression - Best Hyperparameters:")
print(best_logistic_params)
print("Accuracy: {:.2f}".format(accuracy_logistic))
print("Precision: {:.2f}".format(precision_logistic))
print("Recall: {:.2f}".format(recall_logistic))
print("F1-Score: {:.2f}".format(f1_logistic))
print("ROC-AUC: {:.2f}".format(roc_auc_logistic))
```