**An Industrial Oriented Project Report**

On

**“SONAR ROCK VS MINE PREDICTION”**

**Meenugu Srinivas [20E31A0556]**

###### ABSTRACT

Underwater Mine usage by the Naval Defense System provides great Security but also possesses a threat to the marine life and submarine vessels as the mines can be easily mistaken for rocks. So, we need a much more accurate system to predict the object as it is very dangerous if a mistake is made. To have a great accuracy we need more accurate data to generate accurate results. Our idea presents a method for prediction of underwater mines and rocks using Sonar Signals. Sonar Signals are used to record the various frequencies of underwater objects at 60 different angles. We constructed three binary classifier models according to their accuracy. Then, prediction models are used to predict the mine and rock categories. Python and Supervised Machine Learning Classification algorithms are used to construct these prediction models.

**TABLE OF CONTENTS**

#### Chapter Title Page No.

###### [Abstract IV](#_bookmark0)

###### List of Figures VII

###### Introduction 1

1. **System Analysis**

2.1 Existing System 3

2.2 Proposed System 4

2.3 Requirement Specifications 8

2.4 Modules 9

1. **Literature Survey 11**
2. **Overview of concepts**

4.1Python 13

4.2 Anaconda Navigator 19

4.3 Algorithm and Techniques 20

4.4 Python Packages 21

4.5 Logistic Regression 21

4.6 K-Nearest Neighbor 22

5. System Design

5.1 System Architecture 23

5.2 UML Diagrams 24

6. Implementation

6.1 Installation 29

6.2 Source Code 33

**7. Testing**

7.1Unit Testing 38

7.2 Functional Testing 38

7.3 System Testing 38

7.4 Performance Testing 39

7.4 Integration Testing 39

7.5 Programing Testing 40

7.6 Validation Testing 40

7.7 User Acceptance Testing 41

7.8 White Box Testing 41

7.9 Black Box Testing 41

**8. Output Screens 43**

**9. Conclusion 45**

**References**  **46**

|  |  |
| --- | --- |
|  |  |

**LIST OF FIGURES**

**Figure Name Page No.**

1. Figure.1 Data Flow diagram for Machine Learning 2
2. Figure.2 Work Process 4
3. Figure.3 Work Flow 23
4. Figure.4 Use case Diagram 24
5. Figure.5 Sequence Diagram 25
6. Figure.6 Class Diagram 26
7. Figure.7 Activity Diagram 27
8. Figure.8 State Diagram 28
9. Figure.9 Architecture for proposed system 28

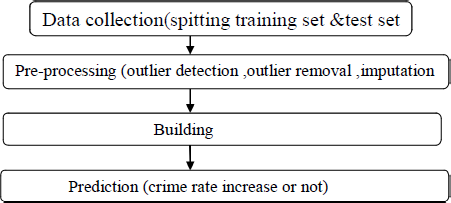
**CHAPTER 1**

**INTRODUCTION**

Underwater mines or naval mines are self-contained explosive devices placed in water to destroy enemies’ surface ships or submarines. Underwater mines are used since the mid-19th Century. Sea mines were introduced by David Bushner in 1977 during the American civil war. Previously mines were only activated by physical contact but the newly created mines can be activated by various methods. Modern mines can be activated by acoustic, pressure, and magnetic changes in the water which provoke them to explode. These are called influence mines. Generally, Underwater mines are classified as offensive or defensive warfare. Mines are strewn across hostile shipping lanes in order to damage merchant ships and military boats. Defensive mines are placed along coastlines to divert enemy submarines and ships away from critical locations and into more heavily guarded places. Usually, Mines are mistaken as rocks during their identification, as mines can have the same shape, length, and width as rocks

Underwater mines, also known as naval mines, have been used since the mid-19th century as self-contained explosive devices to deter enemy surface ships and submarines. Introduced by David Buchner during the American Civil War, these mines still pose a threat today, with approximately 5,000 remaining in the Adriatic Sea from both world wars. Unlike earlier versions that relied on physical contact for activation, modern mines can be triggered by acoustic, pressure, and magnetic changes in the water, known as influence mines. They are categorized as offensive or defensive warfare tools. Offensive mines are scattered across hostile shipping lanes to damage merchant ships and military vessels, while defensive mines are strategically placed along coastlines to divert enemy submarines and ships away from critical areas and into more heavily guarded zones. However, their resemblance to rocks in terms of shape, length, and width often leads to misidentification, necessitating precise input for accurate detection. One effective method for detecting mines is through the use of SONAR technology. SONAR, which stands for Sound Navigation and Ranging, utilizes sound waves to locate and detect objects underwater. Its applications extend beyond military purposes and include acoustic mine detection, fish finding, ocean floor mapping, and locating divers for non- military uses. The range and frequency of SONAR are limited due to sound wave attenuation

typically range from 0.1 to 1 MHz, with a range of 1 to 0.1 km. Ultrasonic waves are preferred over infrasonic waves in SONAR due to their inability to propagate underwater. SONAR is classified into two types: active and passive. Passive SONAR, also known as listening SONAR, detects sounds, while active SONAR employs a sound transmitter and receiver. When the transmitter emits a sound wave that hits an object, it reflects back and creates an echo. By analyzing the frequencies of the object's echo, the receiver determines its nature. In the case of detecting mines or rocks, the frequencies obtained by active SONAR at 60 different angles are used as input to discern between the two. Active SONAR typically operates in the frequency range of 20KHz. The process of mine countermeasures can be divided into four stages. Firstly, detection involves locating targets using various signals such as acoustic or magnetic cues. Secondly, classification is employed to differentiate potential mines from harmless objects. Thirdly, identification confirms the classification with the assistance of additional information from tools like underwater cameras. Lastly, disposal entails safely removing or destroying the detected mines.



###### Figure 1: Data flow diagram for Machine Learning

###### 

###### CHAPTER 2

###### SYSTEM ANALYSIS

**2.1 EXISTING SYSTEM**

The existing system for Sonar Rock vs Mine prediction using machine learning algorithms using traditional methods had several disadvantages. The detection of mines was done by explosive ordnance disposal divers, marine mammals, video cameras on mine neutralization vehicles, and laser systems, which can be time-consuming and costly. Additionally, these methods had a limited range and were not highly accurate, leading to the risk of undetected mines. Moreover, the use of such equipment can cause a risk to marine life, and the loss of life cannot be ruled out. These traditional methods had limitations and risks associated with them, such as harm to marine life, insufficient accuracy, and the potential loss of human life. As technology improved, SONAR became a primary tool for detecting mines in the underwater environment. SONAR uses sound waves to detect objects in water and has proved to be an effective tool for detecting mines in real-time. However, even the SONAR-based system has some limitations, such as the possibility of generating false positives, difficulty in detecting small-sized mines, and the need for constant calibration.

###### PROBLEM SYSTEM

* Predict Existing methods for Sonar Rock vs Mine prediction are time-consuming and

expensive, requiring specialized equipment and resources.

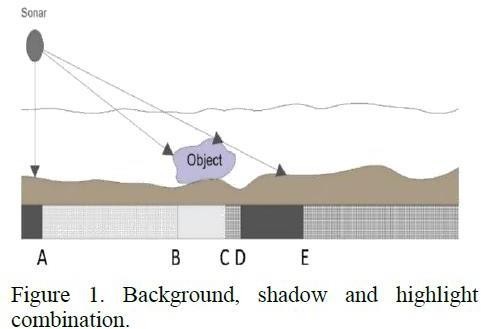
* Traditional methods have limited detection ranges and accuracy, risking undetected mines and compromising naval safety and security.

###### 2.2 PROPOSED SYSTEM

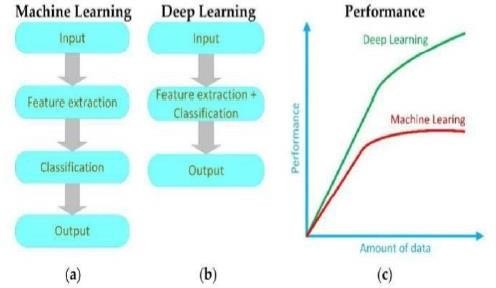
We developed a predictive system for underwater mine detection using machine learning techniques on a dataset obtained by striking metal cylinders with sonar signals from 60 angles. Our models accurately distinguish between rocks and mines based on their unique frequency characteristics. To enhance the safety and effectiveness of Mine Countermeasure (MCM) units in identifying mines and protecting naval vessels, we implemented computer-aided detection (CAD), computer-aided classification (CAC), and automated target recognition (ATR) algorithms. These algorithms analyze texture-based, geometrical, and spectral features in sonar images. While machine learning has limitations, deep learning overcomes them with its ability to work with vast amounts of data and perform automatic feature extraction, making it more reliable for mine detection and classification.

**Figure** **2**:

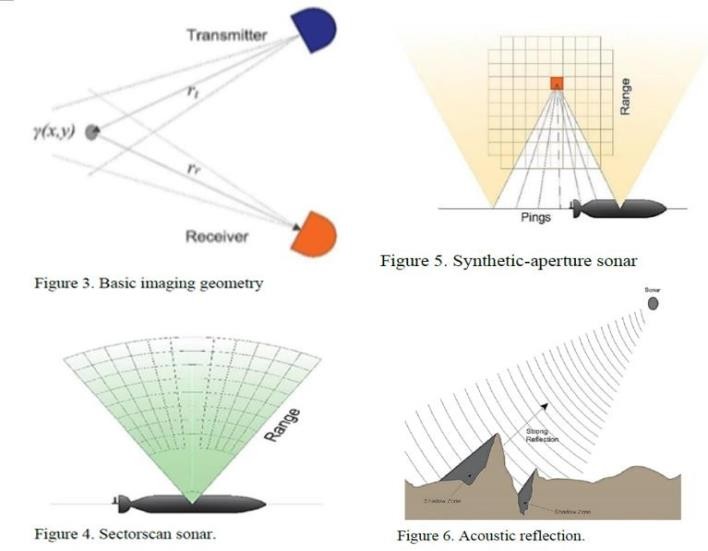
(a)Working process of ML;

 (b)working process of DL;

(c)performance of ML and DP.



Deep learning algorithms in mine detection face challenges due to limited high- quality data availability. Techniques like sonar data simulation and augmentation address this issue. Transfer learning and algorithm fusion improve reliability. Combining classical image processing with deep learning enhances performance and mitigates unbalanced data effects. A review of recent and past methods in mine detection and classification is presented. Sonar, utilizing acoustic waves, measures the time taken for waves to bounce back from objects, allowing distance calculation. Factors like water properties and frequency affect sonar performance, and hydrophones are used for transmission and reception.



**METHODOLOGIES:**

* + Data collection: Gather relevant data on geology, mining techniques, equipment, etc., manually or using sensors.
  + Data preprocessing: Clean, scale, split data for analysis, removing missing values and normalizing it.
  + Feature selection: Choose important features using domain knowledge or techniques like correlation analysis or PCA.
  + Model training: Train logistic regression and K-nearest neighbor models by adjusting parameters to minimize loss.
  + Model evaluation: Assess performance using metrics like accuracy, precision, recall, and F1 score.
  + Model tuning: Adjust parameters to optimize model performance, e.g., k in K- nearest neighbor.
  + Model deployment: Integrate optimized models with mining systems for real- world prediction scenarios.
  + Object Detection: Detect mines using highlight and shadow segments despite environmental obstacles.
  + Image enhancement: Apply techniques like histogram equalization and denoising to normalize images for detection and classification.
  + Image segmentation: Identify mine-related regions like highlights and shadows using thresholding, MRF, fuzzy functions, or deep learning.
  + MLO Detection: Use Gabor-based deep neural network to accurately detect MLOs at multiple scales for AUVs.

**ADVANTAGES:**

* + Accurate prediction: Reliability is improved by using machine learning on a dataset obtained through SONAR simulations.
  + Diverse dataset: Data from various repositories increases diversity, improving model generalization.
  + Simulation-based data collection: Controlled environment allows precise labeling and avoids risks associated with real mines.
  + Multiple angle recordings: Dataset with 60 different angles enhances models' ability to distinguish between rocks and mines.
  + Real-time detection potential: System can be deployed for timely mine detection, enhancing safety and effectiveness.
  + Scalability and applicability: System can adapt to different underwater environments and integrate with existing mine detection systems.

#### 

#### REQURIMENT SPECIFICATION

###### INTRODUCTION

Requirements are the basic constrains that are required to develop a system.

Requirements are collected while designing the system. The following are the requirements that are to be discussed.

* + - Functional requirements
    - Non-Functional requirements
    - Environment requirements
    - Hardware requirements
    - Software requirements

###### FUNCTIONAL REQUIREMENTS

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, NumPy, mat plot lib and sea born.

###### NON-FUNCTIONAL REQUIREMENTS

* Problem definition
* Preparing data
* Evaluating algorithm
* Improving results
* Prediction the result

###### 

###### HARDWARE AND SOFTWARE SPECIFICATIONS

**SOFTWARE REQUIREMENTS**

* + Windows 7or higher.
  + Colab – an online python interpreter

**HARDWARE REQUIREMENTS**

* + Processor – i3 or higher
  + Memory – 1GB RAM
  + Hard Disk - 1TB

**2.4 MODULES**

* DATA PRE-PROCESSING
* FEATURE SELECTION AND REDUCTION
* SEGMENTATION
* PREDICTION USING MACHINE LEARNING ALGORITHM

###### LIBRARIES USED:

* NumPy: performs a wide variety of mathematical operations on arrays .
* Pandas: used for creating data frames
* Sklearn.model\_selection import train\_test\_split : splits our original data into training and test data
* Sklearn.linear\_model import accuracy\_score : evaluate our model to check how well our model is performing.
* Sklearn.metrics import LogisticsRegression & S klearn.neighbors import KNeighborsClassifier: statistical method for predicting binary classes
* import matplotlib.pyplot as plt Matplotlib :is a python library used to create 2D graphs and plots by using python scripts

###### DATA PREPROCESSING:

Pandas: For data manipulation and preprocessing tasks such as loading the dataset, handling missing values, and data cleaning. NumPy: For numerical operations and array manipulations required during data preprocessing

###### FEATURE SELECTION AND REDUCTION:

Scikit-learn: Offers various feature selection techniques such as correlation analysis, recursive feature elimination (RFE), and principal component analysis (PCA) for reducing the dimensionality of the dataset and selecting relevant features.

###### SEGMENTATION

Scikit-image: Provides image processing functions for segmenting the Sonar signals or images into highlight and shadow areas related to mines. OpenCV: Offers tools and algorithms for image segmentation, contour detection, and region-based segmentation.

###### PREDICTION USING MACHINE LEARNING:

Scikit-learn: Implements the Logistic Regression and K-Nearest Neighbor algorithms for classification tasks. TensorFlow or PyTorch: Deep learning frameworks that provide advanced neural network models for classification tasks, such as convolutional neural networks (CNNs).

**CHAPTER 3**

#### LITERATURE SURVEY

###### REFERENCE 1:

**TITLE:** Application of machine learning algorithms for classification of mines and rocks using sonar data.

**AUTHORS:** Abhishek Kumar andSanjay Kumar Soni.

**DESCRIPTION**: The study uses a dataset of sonar readings collected from underwater mines and rocks. The authors applied several machine learning algorithms, including decision trees, random forests, support vector machines (SVM), and k-nearest neighbors(KNN), to classify the sonar data. They evaluated the performance of each algorithm based on several performance metrics, such as accuracy, precision, recall, and F1-score.

###### DRAWBACKS:

The choice of algorithm used for classification can significantly affect the accuracy of the results. The selection of the best algorithm for a specific task requires expertise in machine learning.

###### REFERENCE 2:

**TITLE:** Mine Detection and Classification Using Support Vector Machines and Principal Component Analysis

**AUTHORS:** Eun Jin Kim, Tae Seong Kim, and Soo Hyung Kim

**DESCRIPTION:** The study used a dataset of GPR signals collected from different types of landmines buried in the ground. The authors applied PCA to reduce the dimensionality of the dataset and SVM to classify the GPR signals into mine or non mine categories. They evaluated the performance of the algorithm based on several performance metrics, such as accuracy, sensitivity, and specificity.

###### DRAWBACKS:

Computational requirements: SVM can be computationally expensive, especially when working with large datasets. This can limit the scalability of the algorithm for some application

###### REFERENCE 3:

**TITLE:** Mine Detection and Classification Using Neural Networks

**AUTHORS:** R. Todd Moon and W. Dale Blair.

**DESCRIPTION:** The study used a dataset of GPR signals collected from different types of landmines buried in the ground. The authors used a feedforward neural network with backpropagation to classify the GPR signals into mine or non mine categories. They evaluated the performance of the algorithm based on several performance metrics, such as accuracy, false alarm rate, and receiver operating characteristic (ROC) curve

###### DRAWBACKS:

Model complexity: Neural networks can be complex models that require significant computational resources to train and deploy. This can limit the scalability of the algorithm for some applications

**CHAPTER 4**

**OVER VIEW OF CONCEPTS**

###### 4.1 PYTHON

Python could be an interpreted, high-level and general programing language. Python is a style philosophy emphasizes code readability with its notable use of great indentation. Its language constructs and object-oriented approach aim to assist programmers write clear, logical code for tiny and large- scale projects. Python is commonly represented as A battery enclosed language because of its comprehensive commonplace library. Python is a multi-paradigm programming language, Object-oriented programming and structured programming language are absolutely supported and plenty of its options support practical programming and aspect-oriented programming metaprogramming and metaobjects (magic methods)). several different paradigms are supported via extensions, together with design by contract and logic programming. Python uses dynamic writing and a mix of reference numeration and a cycle-detecting dustman for memory management. It also options dynamic name resolution (late binding), that binds technique and variable names throughout program execution.

Python is an interpreter, object-oriented, high-level programing language with dynamic semantics. Its high-level inbuilt information structures, combined with dynamic typing and dynamic binding build it terribly engaging for fast Application Development, still as to be used as a scripting or glue language to attach existing parts together. Python' simple, straightforward to find out syntax emphasizes readability and so reduces the cost of program maintenance. Python supports modules and packages, which inspires program modularity and code reuse. The Python interpreter and also the intensive commonplace library are on the market in supply or binary type at no cost for all major platforms and may be freely distributed. Often, programmers fall soft on with Python thanks to the magnified productivity it provides. Since there's no compilation step, the edit-test- debug cycle is implausibly fast. Debugging Python programs is easy, a bug or dangerous input can ne'er cause a segmentation fault. Instead, once the interpreter discovers an error, it raises an exception. once the program doesn't catch the exception, the interpreter prints a stack trace. A supply level

computer programmerpermits examination of native and world variables, analysis of discretional expressions, setting breakpoints, stepping through the code a line at a time, and then on. The writer is written in Python itself, testifying to Python' introverted power. On the opposite hand, usually the fastest thanks to debug a program is to feature a number of print statements to the source, the quick edit-test-debug cycle makes this easy approach terribly effective Python is a dynamic programming language which supports several different programming

paradigms as follows:

* Procedural programming
* Object oriented programming
* Functional programming

Standard: Python byte code is executed in the Python interpreter (similar to Java) platform independent code

* Extremely versatile language Website development, data analysis, server maintenance, numerical analysis,
* Syntax is clear, easy to read and learn (almost pseudo code)
* Common language
* Intuitive object-oriented programming
* Full modularity, hierarchical packages
* Comprehensive standard library for many tasks
* Big community
* Simply extendable via C/C++, wrapping of C/C++ libraries
* Focus: Programming speed

**INTRODUCTION TO PYTHON**

Guido van Rossum began functioning on Python within the late 1980s,as a successor to the fundamental principle programming language, and initial discharged it in 1991 as Python 0.9.0. Python two.0 was released in 2000 and introduced new features, akin to list comprehensions and a trash collection system victimization reference count and was discontinued with version 2.7.18 in 2020. Python 3.0 was released in 2008 and was a serious revision of the language that's not fully backward-compatible and far Python 2 code doesn't run unqualified on Python 3. Python 2.0 was released on sixteen Gregorian calendar month 2000, with several major new options, together with a cycle-detecting refuse collector and support for Unicode.

Python three.0 was discharged on 3 Gregorian calendar month two008. it had been a serious revision of the language that's not fully backward-compatible. several of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 embody the 2to3 utility, that automates (at least partially) the interpretation of Python 2 code to Python 3. Python 2.7’s end-of- life date was at the start set at 2015 then delayed to 2020 out of concern that an outsized body of existing code couldn't simply be forward-ported to Python 3. No a lot of security patches or different enhancements are going to be discharged for it. With Python 2’s end-of-life, solely Python 3.6.x and later are supported. Python 3.9.2 and 3.8.8 were speeded up as all versions of Python (including 2.7) had security issues, resulting in potential remote code execution and internet cache poisoning.

###### DESIGN PHILOSOPHY AND FEATURES

The standard library has 2 modules (itertools and functools) that implement practical tools borrowed from Haskell and normal ML. instead of having all of its practicality designed into its core, Python was designed to be extremely extensile (with modules). This compact modularity has created it significantly standard as a method of adding programmable interfaces to existing applications. Python attempts for a simpler, less-cluttered syntax and synchronic linguistics whereas giving developers a alternative in their writing methodology.

Python developers strive to avoid premature optimization, and reject patches to non-critical elements of the Python reference implementation that may provide marginal will increase in speed at the price of clarity. once speed is important, a Python technologist will move time-critical functions to extension modules written in languages reminiscent of C, or use PyPy, a just-in-time compiler. Python is additionally available, that interprets a Python script into C and makes direct C-level API calls into the Python interpreter.

###### SYNTAX AND SEMANTICS, INDENTATION

Python is supposed to be a simply clear language. Its format is visually uncluttered, and it usually uses English keywords wherever different languages use punctuation. not like several other languages, it doesn't use crisp brackets to delimit blocks, and semicolons once statements are allowed however are rarely, if ever, used. it's fewer syntactical exceptions and special cases than C or Pascal. Python uses whitespace indentation, instead of curly brackets or keywords, to delimit blocks.

###### TYPING

Python makes use of duck typing and has typed gadgets however untyped variable names. Type constraints aren't checked at assemble time; rather, operations on an item may also fail, signifying that the given item isn't always of a appropriate type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that aren't well-defined (for example, including a range of to a string) in preference to silently trying to make feel of them. Python permits programmers to outline their very own sorts the use of training, which can be most usually used for item-orientated programming. New times of training are built through calling the class (for example, Spam Class () or EggsClass ()), and the training are times of the meta class type (itself an example of itself), permitting metaprogramming and reflection.

###### LIBRARIES

Python’s giant standard library, usually cited united of its greatest strengths, provides tools suited to several tasks. For Internet-facing applications, many standard formats and protocols corresponding to MIME and hypertext transfer protocol are supported. It includes modules for making graphical user interfaces, connecting to relative databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing. Some components of the quality library are lined by specifications (for example, the net Server entranceway Interface (WSGI) implementation wsgiref follows life 333), however most modules are not. they’re given by their code, internal documentation, and check suites. However, as a result of most of the quality library is cross-platform Python code, solely a number of modules want sterilization or redaction for variant implementations. As of March 2021, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 290,000 packages with a large vary of functionality, including: Automation

* Data analytics
* Databases
* Documentation
* Graphical user interfaces
* Image processing
* Machine learning
* Mobile App
* Multimedia
* Computer Networking
* Scientific computing
* System administration
* Test frameworks
* Text processing
* Web frameworks
* Web scraping

###### DEVELOPMENT ENVIRONMENTS

Most Python implementations (including CPython) embody a read–eval– print loop (REPL), allowing them to perform as a instruction interpreter that the user enters statements consecutive and receives results immediately. different shells, as well as IDLE and IPython, add further skills like improved auto- completion, session state retention and syntax highlighting. still as normal desktop integrated development environments, there are internet browser-based IDEs; Sage science (intended for developing science and math-related Python programs); PythonAnywhere, a browser-based IDE and hosting environment; and cover IDE, an advertisement Python IDE action scientific computing.

###### 4.2 ANACONDA NAVIGATOR - JUPYTER NOTEBOOK

Anaconda may be a distribution of the Python associate degreed R programming languages for scientific computing (data science, machine learning applications, large-scale knowledge processing, prophetical analytics, etc.), that aims to change package management and deployment. The distribution includes data-science packages appropriate for Windows, Linux, and macOS. it's developed and maintained by boa, Inc., that was based by Peter Wang and Travis Oliphant in 2012. As a boa, Inc. product, it is additionally referred to as boa Distribution or boa Individual Edition, whereas different merchandise from the corporate are boa Team Edition and Anaconda Enterprise Edition, each of which aren't free. The subsequent applications are on the market by default in Navigator:

* + - * + JupyterLab
        + Jupyter Notebook
        + QtConsole
        + Spyder
        + Glue
        + Orange
        + RStudio
        + Visual Studio Code

Jupyter Notebook (formerly IPython Notebooks) may be a net-based interactive machine setting for making Jupyter notebook documents. The "notebook" term will informally build respect to many alternative entities, chiefly the Jupyter web application, Jupyter Python web server, or Jupyter document format counting on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which might contain code, text (using Markdown),

mathematics, plots and made media, sometimes ending with the ".ipynb" extension. A Jupyter Notebook will be born-again to variety of open commonplace output formats (HTML, presentation slides, LaTeX, PDF, Restructured Text, Markdown, Python) through "Download As" within the net interface, via the nbconvert library or "jupyter nbconvert" command interface during a shell. To change visualization of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly on the market notebook document, convert it to markup language on the fly and show it to the user.

Jupyter Notebook provides a browser-based REPL engineered upon variety of common ASCII text file libraries:

* + - * + IPython
        + ØMQ (ZeroMQ)
        + Tornado (web server)
        + jQuery
        + Bootstrap (front-end framework)
        + MathJax

Jupyter Notebook absolutely was forty-nine Jupyter-compatible kernels for several programming languages, together with Python, R, Julia and Haskell. The Notebook interface was additional to IPython within the 0.12 unleash (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0–Jupyter1.0).

###### 4.3 ALGORITHM AND TECHNIQUES

**ALGORITHM USED:**

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, biometric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After

understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

###### 4.4 PYTHON PACKAGES:

**sklearn:** In python, sklearn is a machine learning package which include a lot of ML algorithms.Here, we are using some of its modules like train test split, Decision Tree Classifier or Logistic Regression and accuracy score.

###### NumPy: It is a numeric python module which provides fast math functions for

###### calculations.It is used to read data in NumPy arrays and for manipulation purpose.

###### Pandas: Used to read and write different files. Data manipulation can be done easily with data frames

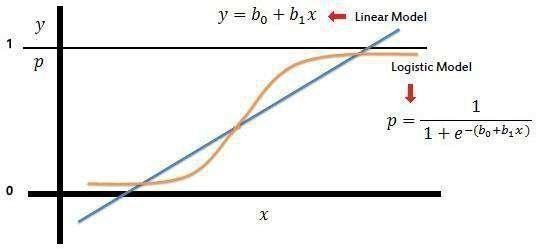
###### 4.5 LOGISTIC REGRESSION:

It is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1(yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts P(Y=1) as a function of X.

###### Logistic Regression Assumptions:

* Binary logistic regression requires the dependent variable to be binary. For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome .Only the meaningful variables should be included.
* The independent variables should be independent of each other. That is, the mode should have little.
  + The independent variables are linearly related to the log odds.
  + Logistic regression requires quite large sample sizes.



###### 4.6 K-NEAREST NEIGHBOR:

To implement K-Nearest Neighbors (KNN) for prediction, load the dataset into a pandas dataframe, split it into input features (X) and target variable (y), split the data into training and testing sets, create a KNN model using KNeighbors Classifier, fit the model to the training data, make predictions on the testing data, and evaluate the model's performance using a confusion matrix.

###### Assumptions of K-Nearest Neighbors:

Instance Similarity: KNN assumes that similar instances have similar labels, calculated based on their feature values.Local Smoothness: KNN assumes neighboring instances have similar labels due to similar data distributions.

* Optimal Value of K: KNN assumes an optimal value of K, impacting bias-variance trade- off and prediction accuracy.
* Feature Scaling: KNN is sensitive to feature scale and assumes equal contribution, requiring appropriate scaling.
* Noise and Outliers: KNN is affected by noise and outliers, potentially leading to inaccurate predictions

#### 

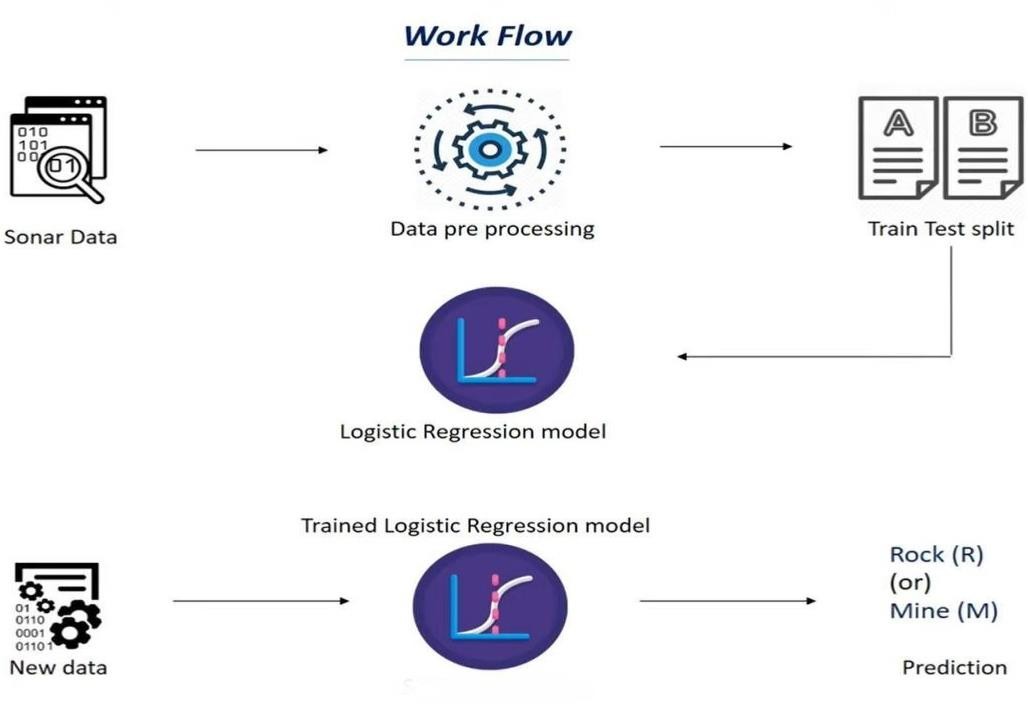
#### 

#### CHAPTER 5

#### SYSTEM DESIGN

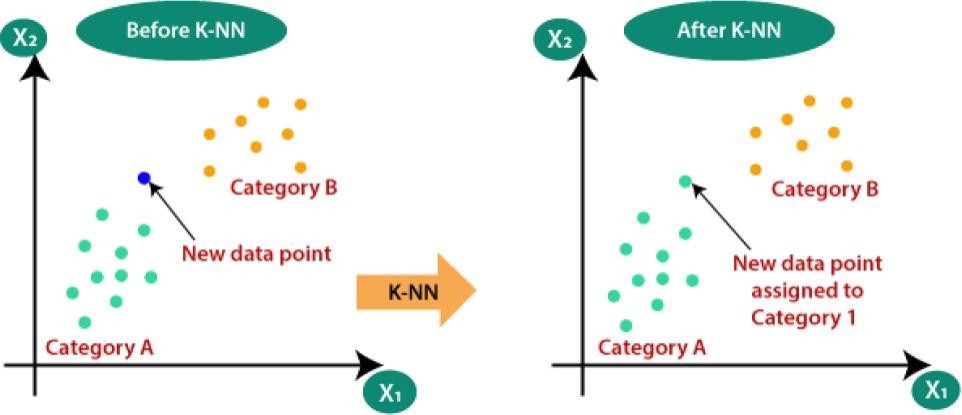
###### 5.1 SYSTEM ARCHITECTURE

###### LOGISTIC REGRESSION



**Figure 3:** Work flow

**K-NEAREST NEIGHBOR**



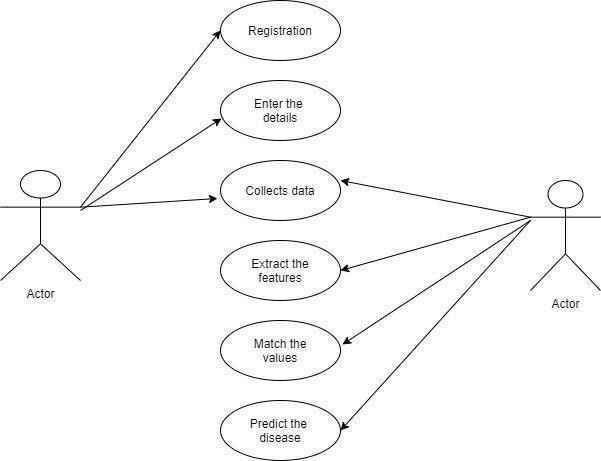
###### 

###### 

###### 5.2 UML DIAGRAMS

###### USECASE DIAGRAMS

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. A Use Case describes a sequence of actions that provided something of unmeasurable value to an actor and is drawn as a horizontal ellipse. An actor is a person, organization or external system that plays a role in one or more interaction with the system.

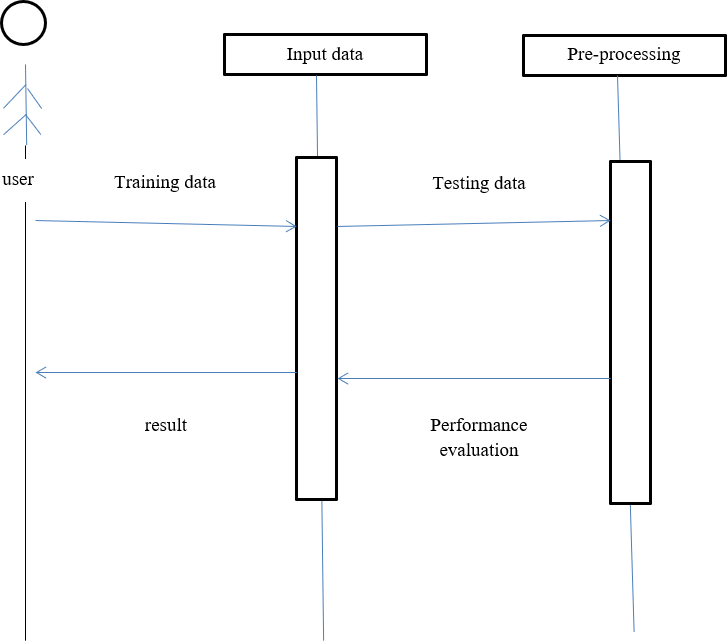


###### 

###### Figure 4: Use case Diagram

**SEQUENCE DIAGRAM**

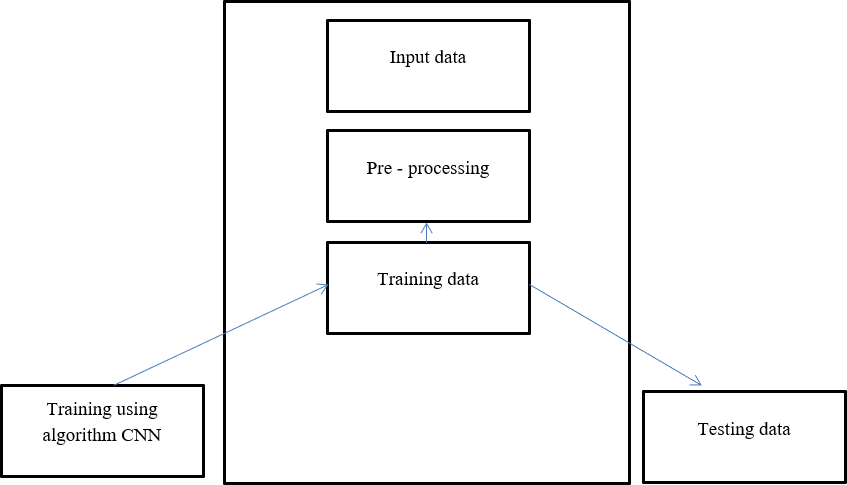
A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.



###### Figure 5: Sequence Diagram

**CLASS DIAGRAM**

A Class diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

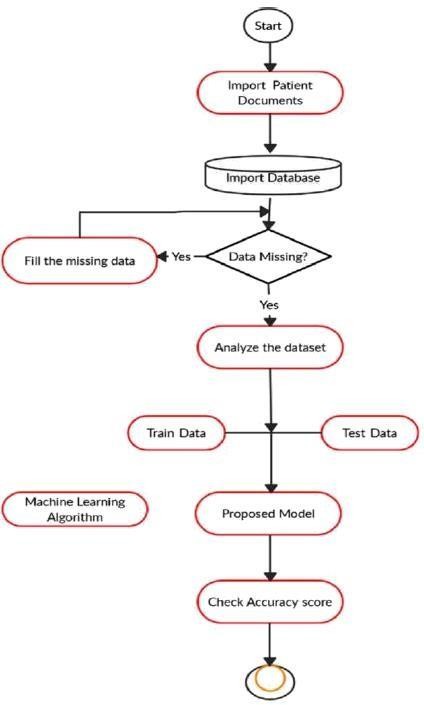


###### Figure 6: Class Diagram

**ACTIVITY DIAGRAM**

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

* Rounded rectangles represent activities.
* Diamonds represent decisions.
* Bars represent the start or end of concurrent activities.
* An encircled circle represents the end of the workflow.
* A black circle represents the start of the workflow

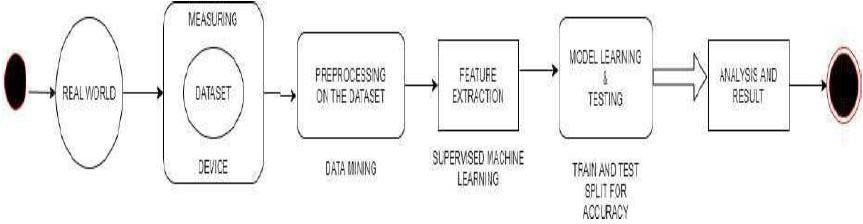


###### 

###### Figure 7: Activity Diagram

**STATE DIAGRAM**

A state diagram is a graphical representation that models the behavior of a single object, specifying the sequence of events that object goes through during its lifetime in response event. It is a preliminary step used to create an overview of the system.



###### Figure 8: State Diagram

###### ARCHITECTURE DIAGRAM FOR PROPOSED SYSTEM

###### 

###### 

###### Figure 9: Architecture diagram for proposed system

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 INSTALLATION OF GOOGLE COLAB**

Google Colab is a free cloud-based platform that allows data scientists and developers to run and share their code in a Jupyter Notebook environment. It is a popular tool for machine learning and data analysis because it provides access to powerful computing resources and a wide range of libraries and tools.

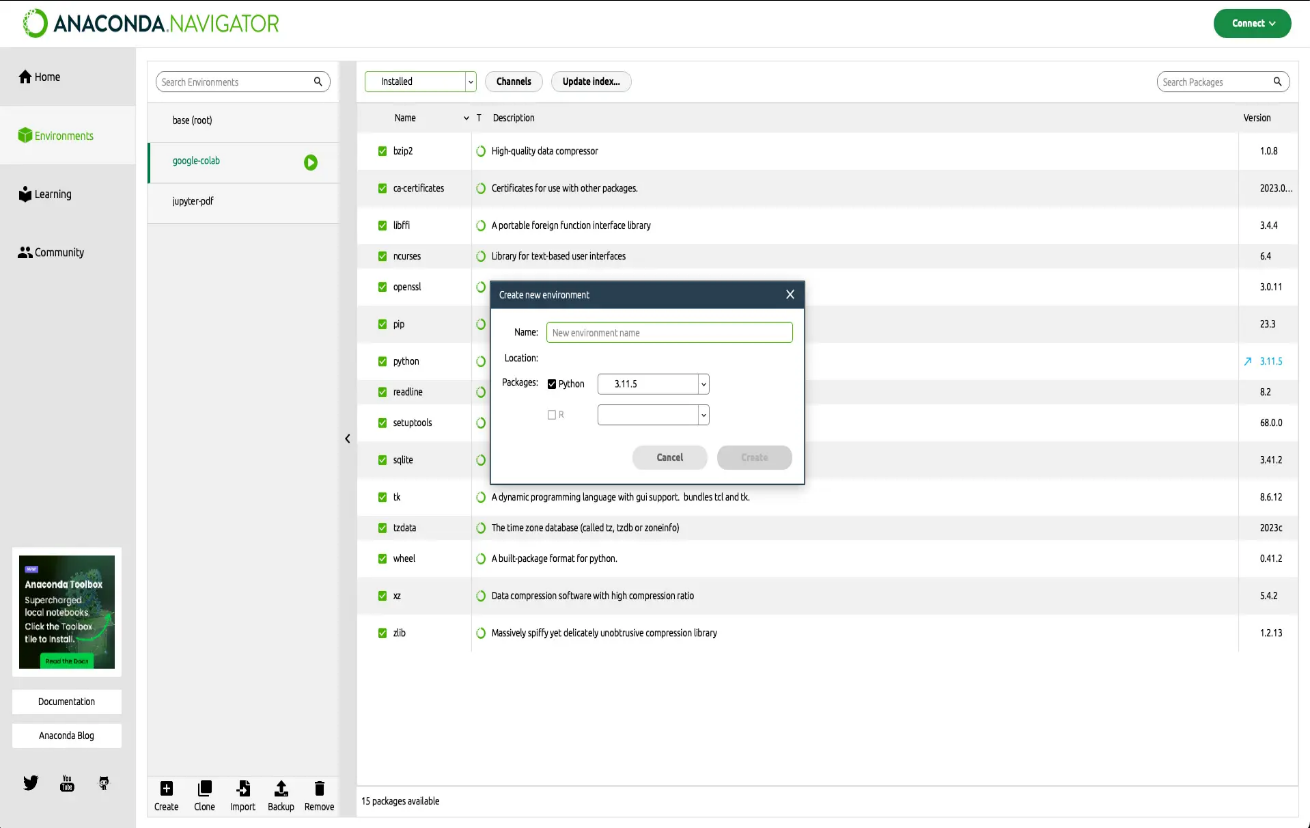
However, there may be times when you want to run Google Colab locally on your own machine. This could be because you want to work offline, have more control over your environment, or need to work with sensitive data that cannot be shared in the cloud.

## Step 1: Install Anaconda

## Anaconda is a popular data science platform that provides a Python distribution, package manager, and environment manager. It is a great tool for managing your Python dependencies and creating isolated environments for your projects.To install Anaconda, go to the Anaconda website and download the appropriate version for your operating system. Follow the installation instructions provided by the installer.

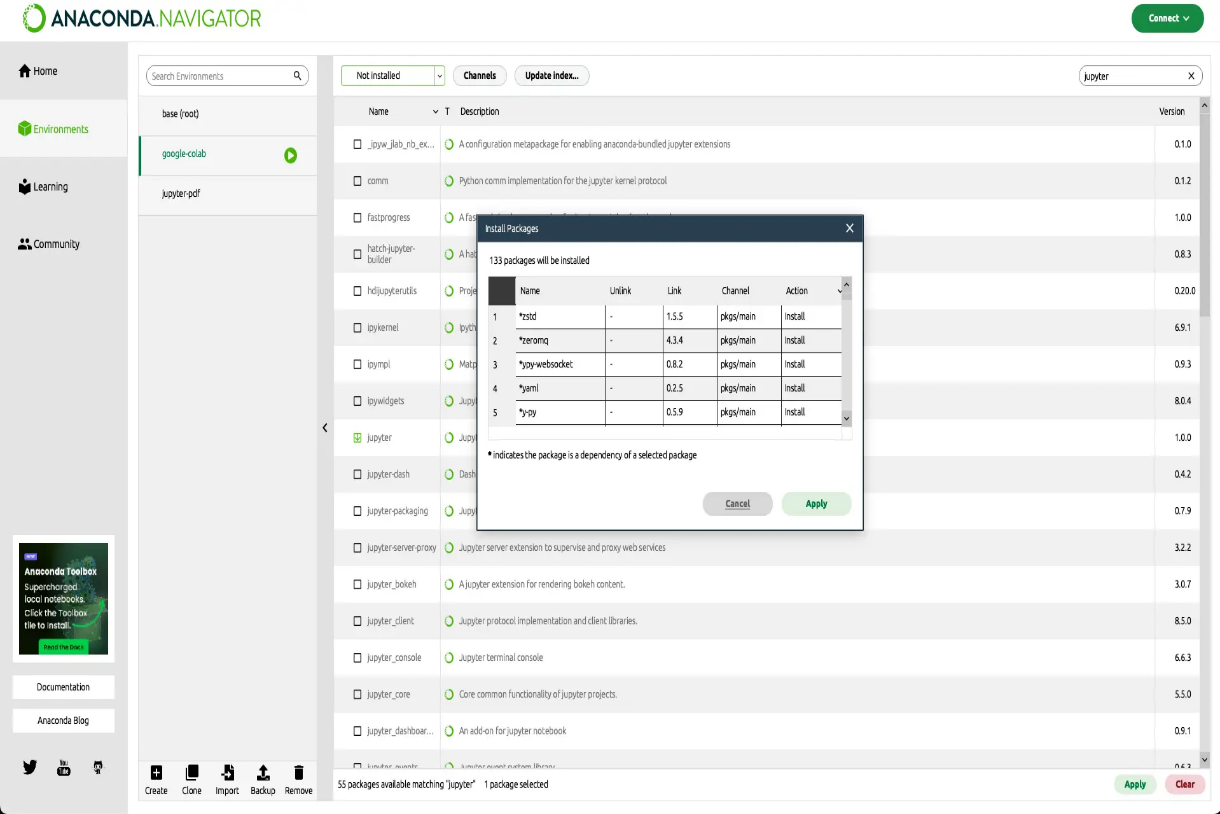
## Step 2: Create a New Environment

## Once you have installed Anaconda, you can create a new environment for running Google Colab locally. This will ensure that you have all the necessary packages and dependencies installed without affecting your system-wide Python installation.To create a new environment, open the Anaconda Navigator and click on the “Environments”tab. Click the “Create” button to create a new environment.Give your new environment a name and choose the Python version you want to use. We recommend using Python 3.6 or later, as this is the version used by Google Colab. Next, click the “Create” button to create your new environment. This may take a few minutes as Anaconda installs all the necessary packages and dependencies.Conda navigator creating environment

****

**Step 3: Install Jupyter Notebook**

Jupyter Notebook is a web-based interactive computing environment that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.To install Jupyter Notebook in your new environment, open the Anaconda Navigator and select your new environment from the drop-down menu. Then utilize the search bar to locate the Jupyter Notebook package, select the Jupyter Notebook package, and click the “Apply” button and confirm the installation by clicking “Apply” once more.

****

**Step 4: Install Google Colab**

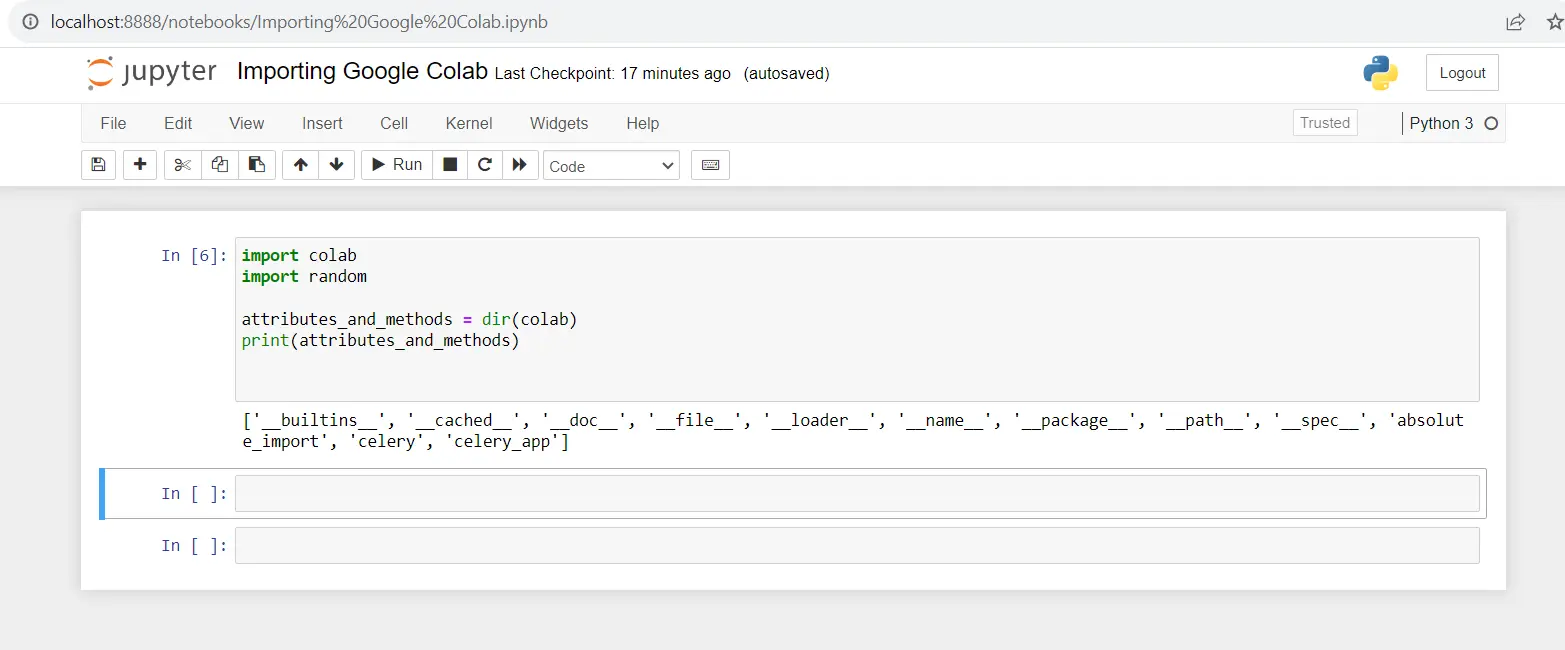
To install Google Colab, you will need to install the colab package using the pip package manager.Open a terminal or command prompt and activate your new environment by running:conda activate <env\_name>Replace <env\_name> with the name of your new environment.Before proceeding with the installation, please check the compatibility of the colab package with your Python version. Some packages may not be compatible with all Python versions. You can typically find this information on the package’s documentation or repository.Then run the following command to install the colab package:pip install colab

**Step 5: Run Google Colab Locally**

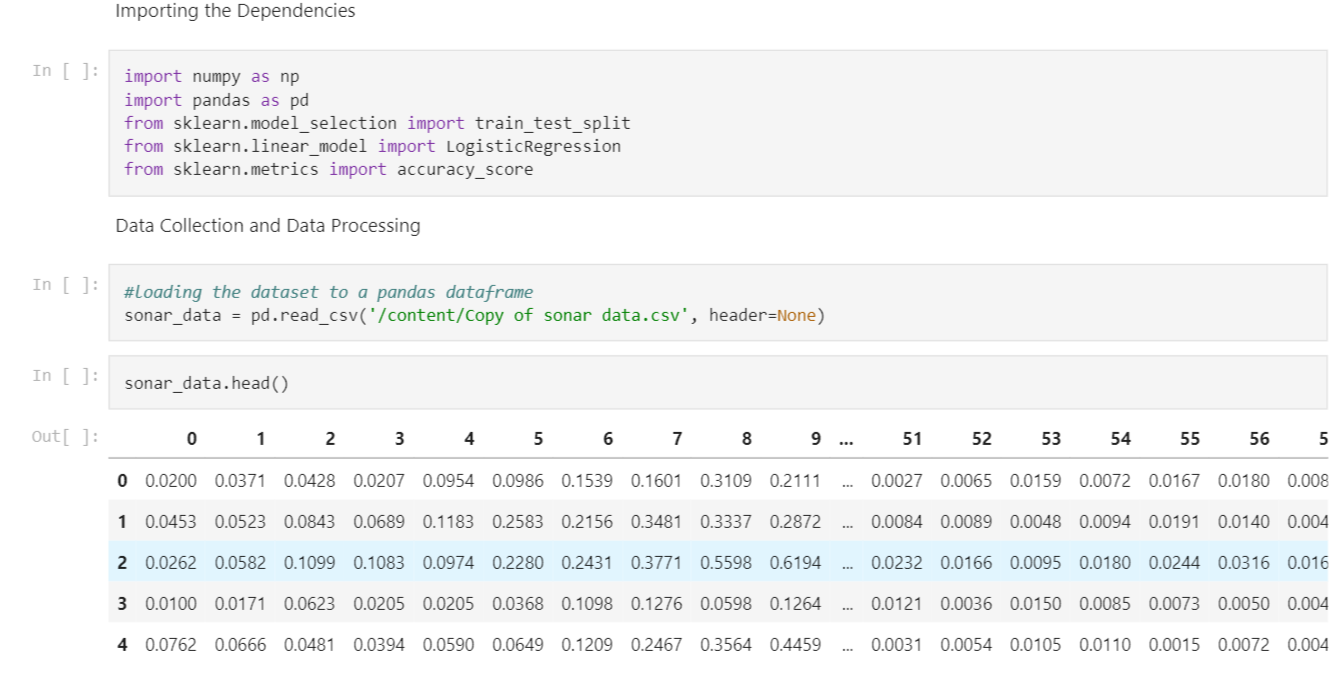
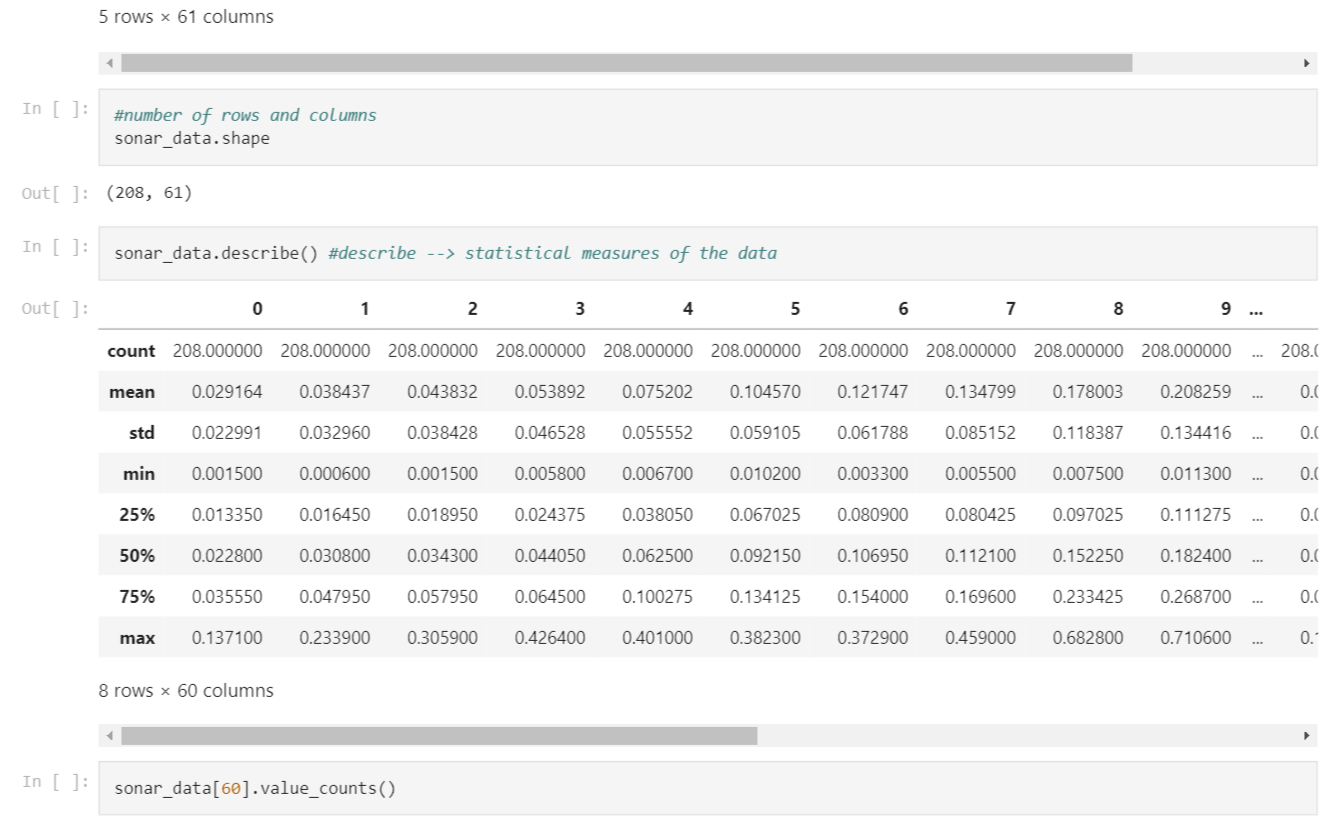
Run the following command to start the Jupyter Notebook server:jupyter notebook

This will open a new tab in your web browser with the Jupyter Notebook interface. You can now create a new notebook and import the colab package to start using Google Colab locally.

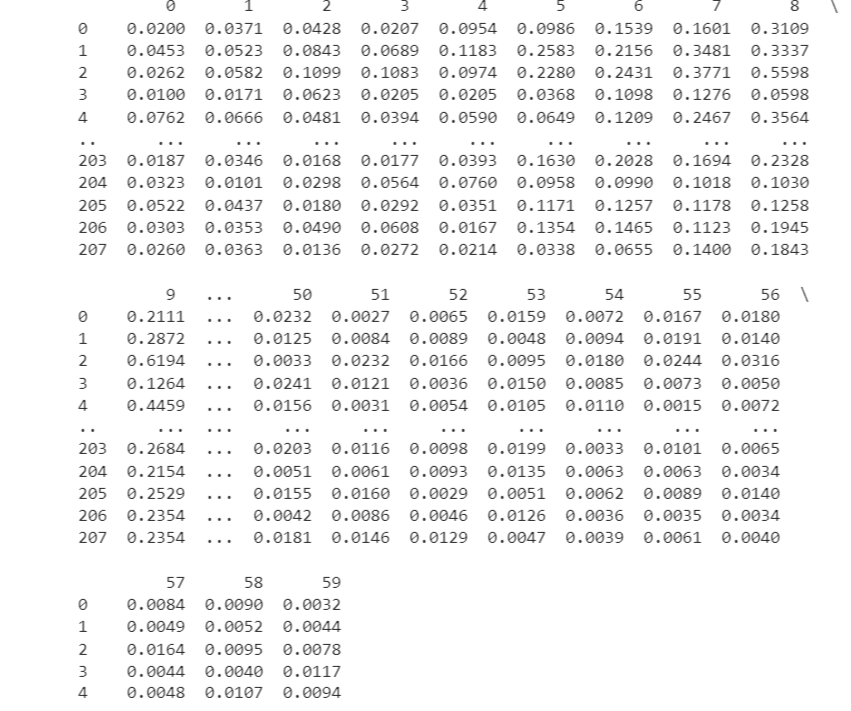
Google Colab test result

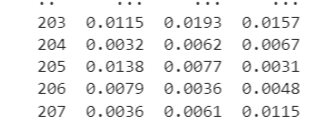


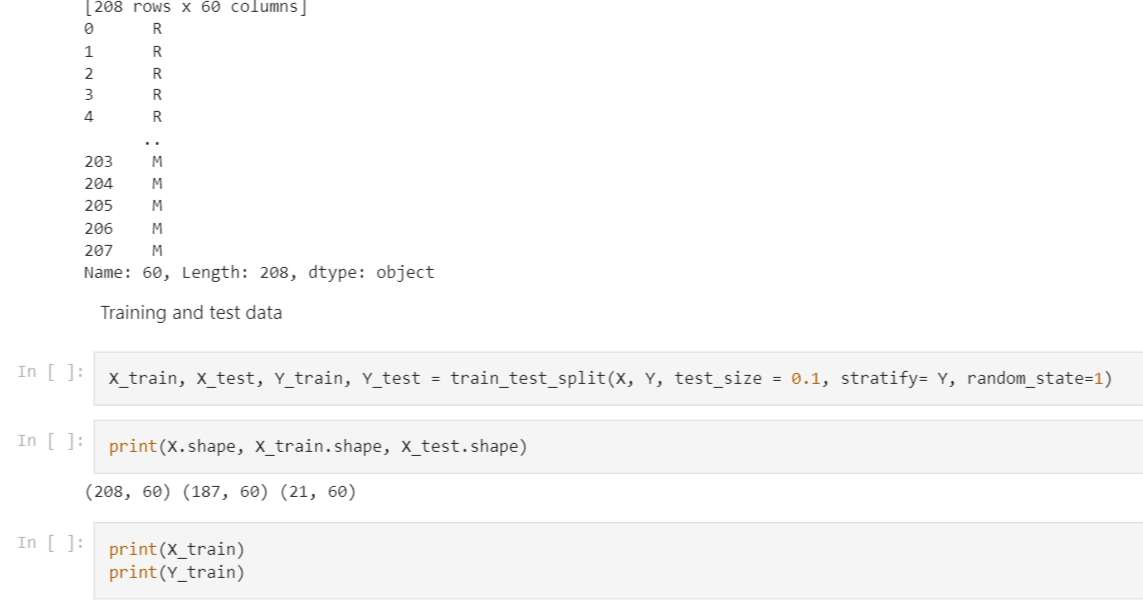
**6.2 SOURCE CODE**

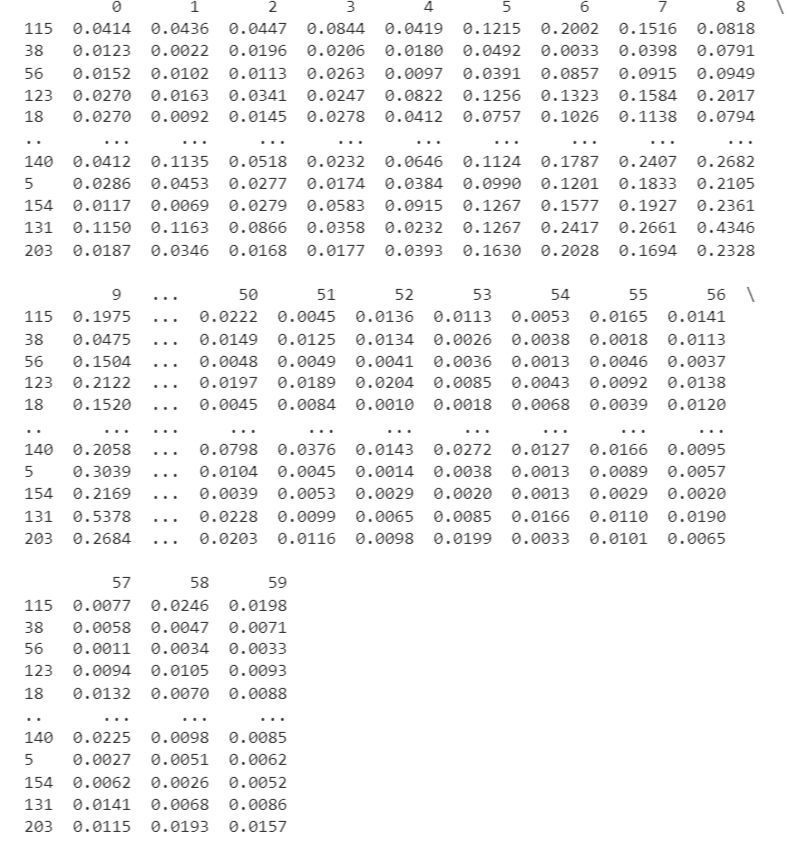
 

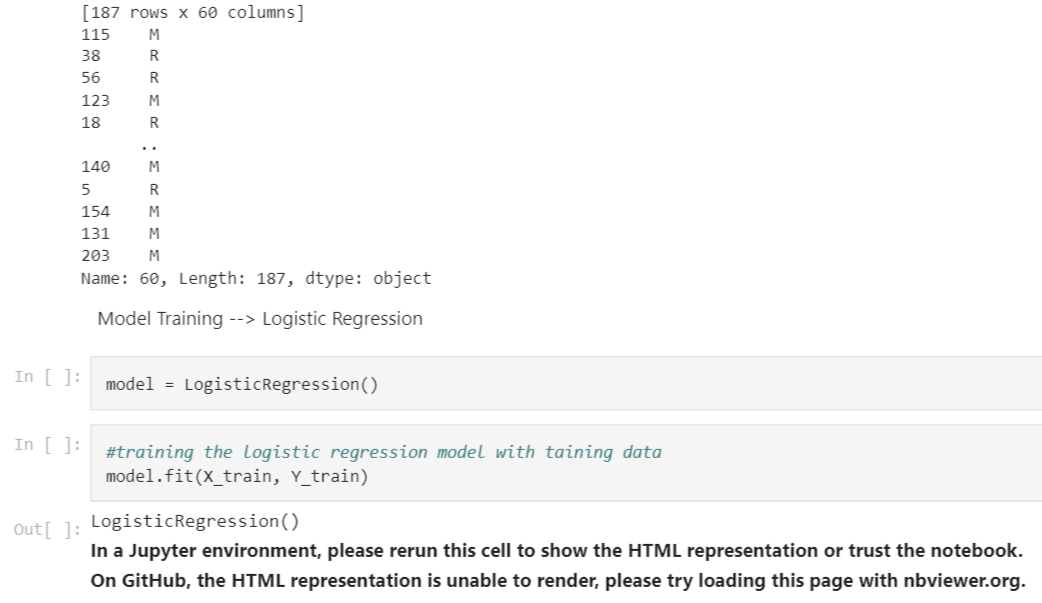


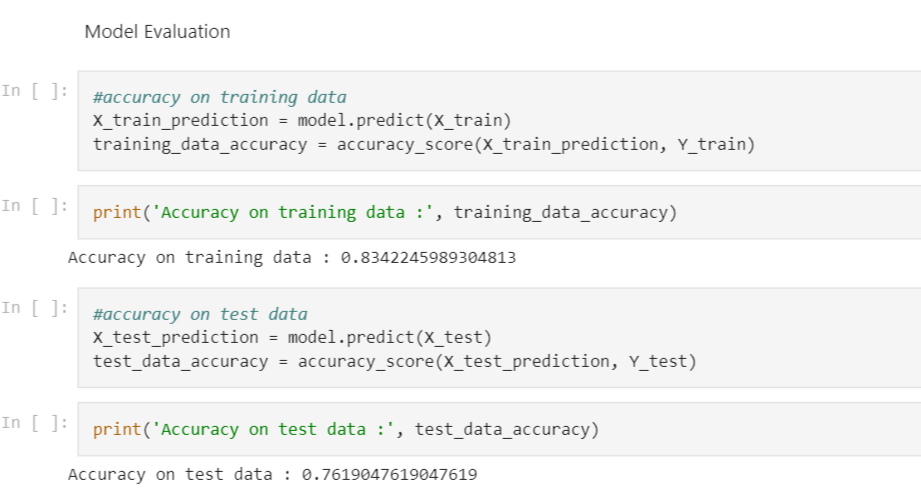














#### 

#### CHAPTER 7

#### TESTING

###### 7.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

###### 7.2 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must be invoked.

###### 7.3 SYSTEM TESTING

System testing ensures that the entire integrated software system meets

requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration point.

###### 7.4 PERFORMANCE TESTING

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

###### 7.5 INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure, while at the same time conducting tests to uncover error associated with interfacing. The following are the types of Integration Testing: -

* Top-down Integration
* Bottom-up Integration

**Top-down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breathe first manner.

###### Bottom-up integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

###### 7.6 PROGRAM TESTING

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

###### 7.7 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

* The function or performance characteristics conform to specifications and are accepted.
* A validation from specification is uncovered and a deficiency created.

###### 

###### 

###### 7.8 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

###### 7.9 WHITE BOXTESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* Flow graph notation
* Cyclometric complexity
* Deriving testcases
* Graph matrices Control

###### 7.10 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

###### SOFTWARE TESTING STRATEGIES

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer-based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any t

#### CHAPTER 8

#### OUTPUT SCREEN

#### 

#### SCREEN SHOT 1: PREDECTING THE OBJECT ROCK

#### 

#### SCREEN SHOT 2 : PREDECTING THE OBJECT MINE

#### 

#### CHAPTER 10

#### CONCLUSION

After analyzing the given dataset, it was found that the k Nearest Neighbors (KNN) algorithm outperformed the Logistic Regression algorithm in accurately distinguishing between rocks and mines. This conclusion was based on evaluating performance metrics including accuracy, precision, recall, and F1 score. However, it is important to note that the superiority of KNN does not imply it is always the optimal choice for every classification problem. The effectiveness of a model depends on factors such as data quality, preprocessing techniques, hyperparameter selection, and evaluation metrics employed. Therefore, it is essential to consider the specific characteristics of the dataset and choose the most suitable algorithm accordingly.

**FUTURE SCOPE**

For future endeavors, In this project we made a model which gives us the best accuracy among all the machine learning algorithms and by using that model we have made a web service Using stream lit which will connect the back end of ml model to the front end that we designed it, The best-performing model was identified, and a web service was created using Stream lit to connect the ML model to a user-friendly front-end interface. By deploying this system online, users can assess their health conditions and take appropriate measures. Future work involves enhancing the deployed model by incorporating additional algorithms and displaying the ROC curve for improved prediction. This web-based platform can also serve as a valuable resource for medical professionals as a second reference for health assessments.

#### REFERENCES

1. Stanislaw Ho˙zy´n. “A Review of Underwater Mine Detection and Classification in Sonar Imagery.” (2021).
2. Sri Ramya Yaramasu, Uppada Sai Gayatri, Vadlamani Manjusha, Vaishna C Bhanu, Koda Indu. “UNDERWATER MINE & ROCK PREDICTION BY EVALUATION OF,” (2016)
3. H. Jia, J. Wu, Y. Sun, and J. Zhang. "A mine ventilation network model based on KNN algorithm and logistic regression," (2017).
4. Wang, J. Yang, and Y. Li. "Research on prediction of mine water inrush based on K-nearest neighbor algorithm and logistic regression," (2016).
5. S. Du, G. Zhang, and F. Chen. "Research on mine roof monitoring and prediction model based on K-nearest neighbor algorithm and logistic regression," (2016).
6. Y. Guo, S. Hu, and X. Zhang. "Research on the prediction model of mine water inrush based on logistic regression and KNN algorithm," (2020)
7. S. Das, P. Dutta, S. Bhattacharjee, and P. Dey, "Sonar-based mine classification using logistic regression and k-nearest neighbor," (2018).
8. L. Wang, Z. Li, X. Zhang, and Y. Chen, "A comparative study of logistic regression and k-nearest neighbor for sonar rock vs. mine classification," (2019).
9. M. Huang, C. Liu, Y. Wang, and X. Li, "Sonar data analysis for rock vs. mine classification using logistic regression and k-nearest neighbor," (2017).
10. R. Sharma, S. Singh, and R. Kumar, "Performance analysis of logistic regression and k-nearest neighbor algorithms for sonar rock vs. mine prediction," (2020).
11. A. Patel, S. Patel, and S. Shah, "Comparative study of logistic regression and k-nearest neighbor for underwater mine detection using sonar signals," (2019).

H. Gupta, P. Chauhan, and S. Singh, "Sonar rock vs. mine prediction using logistic regression and k-nearest neighbor: A case study," (2018).

1. K. Roy, S. Sarkar, and S. Das, "Evaluation of logistic regression and k-nearest neighbor algorithms for sonar-based mine classification," (2017).
2. A. Verma, R. Kumar, and A. Pandey, "Comparative analysis of logistic regression and k-nearest neighbor for sonar rock vs. mine discrimination," (2020).
3. N. Sharma, S. Gupta, and S. Singh, "Performance evaluation of logistic regression and k-nearest neighbor in sonar rock vs. mine classification," (2019).
4. V. Singh, M. Mishra, and S. Kumar, "A study on sonar rock vs. mine prediction using logistic regression and k-nearest neighbor algorithms," (2016)