A

Project Report

On

PREDICTING FISH DISEASES EARLY USING MACHINE LEARNING AND WATER QUALITY ASSESSMENT

Submitted for partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

CSE-DATA SCIENCE

BY

Mahammad Asiya (20N81A6725)

L. Poojitha (20N81A6707)

G. Saketh (20N81A6746)

Under the guidance of

Mr. T. Shravan Kumar M. TECH (CSE)

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING-DATA SCIENCE

SPHOORTHY ENGINEERING COLLEGE

(Affiliated to JNTUH & Recognized by AICTE)

Nadergul, Hyderabad-501510

Academic Year-2023-24

SPHOORTHY ENGINEERING COLLEGE (AUTONOMOUS) Passion Ignited @ Sphoorthy

CERTIFICATE

This is to certify that, this Project report entitled "**Predicting Fish Diseases Early Using Machine Learning and Water Quality Assessment.**" is a bonafied work carried out by Mahammad Asiya (20N81A6725), L. Poojitha (20N81A6707), G. Saketh (20N81A6746), in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science Engineering-DATA SCIENCE from Sphoorthy Engineering College, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad, during the Academic Year 2023-24 under our guidance and supervision.

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

Internal Guide	**Head of the Department**	**External Examiner**
Mr. T. Shravan Kumar	Dr. K. Ramesh Rao	
Assistant Professor	Head, Dept.of CSE-DS	
Dept.of CSE-DS		

Principal

Prof. M.V.S. Ramprasad

SPHN

DECLARATION

We the undersigned, declare that the major project entitled "**Predicting Fish Diseases Early Using Machine Learning and Water Quality Assessment.**" carried out at SPHOORTHY ENGINEERING COLLEGE is original and is being submitted to the Department of **COMPUTER SCIENCE & ENGINEERING-DATA SCIENCE**, Sphoorthy Engineering College, Hyderabad towards partial fulfilment for the award of Bachelor of Technology.

We declare that the results embodied in the Major Project work has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Date:

Place: Hyderabad

Mahammad Asiya (20N81A6725)

L. Poojitha (20N81A6707)

G. Saketh (20N81A6746)

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our Project Co-ordinators **Mr. Mohammed Miskeen Ali and Mrs. Swathi** for their constant and continuous support throughout the project, we would like to thank our, **Mr. T. Shravan Kumar, Assistant professor** department of CSE-DS, Sphoorthy Engineering College, Jawaharlal Nehru Technological University Hyderabad(JNTUH), Hyderabad for his/her inspiring guidance, consistent encouragement, constructive criticism and helpful suggestions during the entire course of my research work.

We express our sincere thanks to **Dr. K. Ramesh Rao**, Professor & Head of CSEData Science department, Sphoorthy Engineering College, Nadargul(V), Balapur(M), Rangareddy(D) for his encouragement which helped us to complete our Project work.

We deem it a great privilege to express our profound gratitude and sincere thanks to **Sri. S. Chalama Reddy**, Chairman, **Sri. S. Jagan Mohan Reddy**, Secretary, **Prof. M.V.S. Ramprasad**, Director & Principal, **Dr.Maheswar Dutta**, Vice Principal and **Dr. Giridhar Akula**, Dean of Academics Sphoorthy Engineering College, for their moral support, necessary arrangements and kind help in the completion of our project work.

We express our heartfull thanks to our department faculty members, non-teaching staff for providing the necessary information pertaining to our project work.

We always greatful to our parents, family members, well-wishers and friends for their remarkable support during our project work.

ABSTRACT

A machine learning-based approach for early detection of fish diseases through the analysis of water quality parameters. Leveraging data on various water quality indicators, the model aims to identify patterns and correlations that can serve as early indicators of potential health issues in aquatic ecosystems. The integration of machine learning techniques provides a proactive and efficient means of monitoring and managing fish health, ultimately contributing to the sustainability of aquaculture practices.

By employing advanced algorithms, the model extracts meaningful insights from diverse water quality data, including parameters like pH, temperature, dissolved oxygen, and nutrient levels. The proposed machine learning system represents a valuable tool for aquaculture management, fostering a more resilient and sustainable approach to fish farming. By leveraging supervised learning algorithms like Support Vector Machines, the model aims to identify patterns indicative of potential health issues. Ensemble methods like Gradient Boosting Machines further enhance predictive accuracy. This comprehensive approach enables the system to proactively identify abnormal patterns in water quality, allowing for timely intervention and improved management practices in aquaculture.

INDEX

Contents			Page No.
Abstract			i
List of Figures			iii
Chapters			
1.	**Introduction**		1-7
	1.1	Problem Statement	1
	1.2	Objective	1-2
	1.3	Motivation	2-3
	1.4	Existing System	3-4
	1.5	Proposed System	4
	1.6	Scope	5- 6
	1.7	Software & Hardware Requirements	7
2.	**Literature Survey**		8-12
	2.1	Survey of Major Area Relevant to Project	8
	2.2	Key Areas of Research	8-9
	2.3	Case Studies and Applications	9-12
3.	**System Design**		13-24
	3.1	System Architecture	13-15
	3.2	System Flow	15-21
	3.3	Module Description	21-24
4.	**Implementation**		25-40
	4.1	Environmental Setup	25-38
	4.2	Implementation of Modules	38-39
	4.3	Integration and development	39-40
5.	**Evaluation**		41-49
	5.1	Datasets	41-44
	5.2	Test Cases	45-48
	5.3	Results	48 – 49
6.	**Conclusion and Future Enhancement**		50
	References		51
	Appendix		52 - 53
	A.	Sample Code	54 - 60

LIST OF FIGURES

	Figure Name	Page No.
3.1	System Architecture	13
3.2	Use Casse diagram	18
3.3	Class Diagram	19
3.4	Sequence Diagram	20
3.5	Collaboration Diagram	21
5.1	Dataset	42
5.2	Virus Found graph	44
5.3	Output Screen (1)	48
5.4	Output Screen (2)	49

CHAPTER-1

INTRODUCTION

1.1 PROBLEM STATEMENT

Aquaculture, the farming of fish and other aquatic organisms, is a critical component of the global food supply. However, fish diseases can cause significant economic losses and impact food security. Early detection of fish diseases is essential for preventing outbreaks and ensuring sustainable aquaculture practices. Traditional methods of diagnosing fish diseases often rely on visual inspections and laboratory tests, which can be time-consuming and require specialized expertise. Advances in machine learning (ML) and water quality assessment offer a promising solution for early and accurate disease detection.

1.2 OBJECTIVE

The primary objective of this project is to develop and implement a machine learning-based predictive model that can accurately and early detect potential fish disease outbreaks by analysing water quality parameters. This model aims to provide aquaculture farmers with timely and actionable insights, enabling them to take preventive measures to protect their stock, thus ensuring the sustainability and profitability of their operations.

1.Early Detection:

Develop a system that can identify early warning signs of fish diseases before they become apparent through traditional observation methods, thereby minimizing the impact of diseases on fish populations.

2.Integration of Water Quality Data:

Utilize real-time water quality monitoring data (such as pH, temperature, dissolved oxygen, ammonia levels, and nitrate levels) to predict the likelihood of disease outbreaks.

3.Machine Learning Model Development:

Create and train various machine learning models (including but not limited to logistic regression, decision trees, random forests, support vector machines, and neural networks) to identify patterns and correlations between water quality parameters and disease outbreaks.

4.Model Optimization:

Fine-tune the predictive models to enhance their accuracy, precision, recall, and overall performance, ensuring reliable predictions across different environments and species.

5.Real-Time Monitoring and Alerts:

Design a real-time monitoring system that continuously assesses water quality and provides immediate alerts to fish farmers when the likelihood of a disease outbreak is high.

6.User-Friendly Interface:

Develop an intuitive interface for aquaculture farmers to easily access predictions, alerts, and recommended actions based on the model's outputs.

1.3 MOTIVATION

The motivation for developing a predictive model to detect fish diseases early using machine learning and water quality assessment stems from several key factors that collectively aim to enhance the sustainability, profitability, and efficiency of aquaculture practices.

1. Economic Impact:

Fish diseases can cause significant economic losses in the aquaculture industry. Disease outbreaks often lead to high mortality rates, reduced growth, and compromised product quality. Early detection allows for timely interventions, minimizing losses and preserving the financial viability of aquaculture operations.

2. Sustainability and Food Security:

Aquaculture is a vital component of global food security, providing a substantial portion of the world's protein supply. Ensuring the health and productivity of farmed fish populations is crucial for meeting the growing demand for seafood. Early disease detection helps maintain stable production levels and supports sustainable aquaculture practices.

3. Environmental Impact

Disease outbreaks in aquaculture can lead to the increased use of antibiotics and other treatments, which may have negative environmental consequences, such as antibiotic resistance and pollution. Early detection and preventive measures can reduce the need for chemical treatments, promoting a healthier environment.

4. Animal Welfare

Proactive disease management improves the overall welfare of fish by preventing the spread of infections and reducing the suffering caused by disease outbreaks. Ensuring the health and well-being of farmed fish aligns with ethical considerations and industry standards for animal welfare.

5. Technological Advancements

Recent advancements in machine learning and sensor technologies have made it possible to collect and analyse large volumes of data in real-time. Leveraging these technologies for disease prediction is a natural progression, harnessing their potential to solve pressing challenges in aquaculture.

1.4 EXISTING SYSTEM

An existing system for fish disease detection utilizes image processing techniques to identify and diagnose diseases in fish populations. Initially, images of fish are captured using cameras or imaging devices placed within aquaculture facilities. These images may encompass various angles and perspectives to ensure comprehensive coverage of the fish population. Subsequently, the acquired images undergo preprocessing, which involves tasks such as noise reduction, contrast enhancement, and image normalization to optimize their quality for analysis. Feature extraction techniques are then employed to identify distinctive visual patterns and characteristics associated with different diseases. These features may include changes in skin colour, lesions, abnormal growths, or behavioural anomalies observable in the images. Machine learning algorithms, such as Support vector machine(svm) and K- means clustering are trained on labelled image datasets to classify fish images based on the presence or absence of specific diseases. The trained model is subsequently deployed to analyse new images in real-time, automatically detecting and diagnosing diseases within the fish population. Continuous monitoring and feedback mechanisms ensure the system's accuracy and effectiveness, with periodic updates and retraining to adapt to evolving disease patterns and environmental conditions.

1.5 PROPOSED SYSTEM

The proposed system for fish disease detection, the focus lies on leveraging water quality data to develop a robust method for early detection and diagnosis of diseases in fish populations. The system integrates machine learning techniques with comprehensive water quality monitoring to provide a holistic approach to disease surveillance in aquaculture settings. The system begins with the installation of sensors and monitoring devices within the aquatic environment to continuously collect data on key water quality parameters. These parameters include temperature, pH levels, dissolved oxygen concentration, ammonia levels, nitrate levels, and other relevant indicators of water quality. The collected data is then transmitted to a central database for processing and analysis. Next, machine learning algorithms are employed to analyze the water quality data and identify patterns or anomalies that may be indicative of disease outbreaks in fish populations. Supervised learning techniques are utilized, where historical data on water quality and known instances of fish diseases are used to train the models. The models learn to recognize subtle changes in water quality parameters that precede disease outbreaks, enabling early detection and intervention. When deviations from normal conditions are detected, the system generates alerts to notify aqua culturists or fish farm managers of potential disease risks. These alerts prompt immediate action, such as increased monitoring, adjustments to water management practices, or targeted treatments to mitigate the spread of disease.

1.6 SCOPE

The scope of this project encompasses several critical phases and activities aimed at developing a comprehensive system for the early detection of fish diseases using machine learning and water quality assessment. The project is structured to address data collection, model development, system integration, and real-world validation.

1. Data Collection:

Historical Data Acquisition: Collaborate with aquaculture farms to gather historical data on water quality parameters (e.g., pH, temperature, dissolved oxygen, ammonia levels, nitrate levels) and records of fish disease outbreaks.

Real-Time Data Collection: Set up sensors and monitoring devices in aquaculture environments to collect real-time water quality data.

Supplementary Data Sources: Use publicly available datasets and scientific literature to enhance the data pool.

2. Data Preprocessing:

Data Cleaning: Handle missing values, outliers, and noise in the dataset to ensure data quality.

Data Transformation: Normalize and standardize data to facilitate model training.

Feature Engineering: Extract and create relevant features from raw data to improve model performance.

3. Feature Selection:

Correlation Analysis: Identify and select the most significant water quality parameters that correlate with the onset of specific fish diseases.

Dimensionality Reduction: Apply techniques such as Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while retaining essential information.

4. Model Development:

Model Selection: Develop and compare various machine learning models, including logistic regression, decision trees, random forests, support vector machines, and neural networks.

Model Training: Train selected models using the prepared dataset, employing cross-validation techniques to ensure robustness.

Hyperparameter Tuning: Optimize model hyperparameters to enhance predictive accuracy and performance.

5. Model Evaluation:

Model evaluation is a critical phase in the development and deployment of machine learning models, particularly in specialized applications such as fish disease detection through water quality assessment. Evaluating the performance of a model involves more than just determining its overall accuracy; it requires a comprehensive understanding of various metrics that offer insights into the model's strengths and weaknesses. The primary goal is to ensure that the model performs reliably and accurately under different conditions and scenarios, enabling aquaculture managers to make informed decisions that enhance fish health and productivity.

Performance Metrics: Evaluate models using metrics such as accuracy, precision, recall, F1 score, and Area Under the Curve (AUC) for Receiver Operating Characteristic (ROC) curves.

Model Comparison: Compare the performance of different models to select the best-performing one.

1.7 SOFTWARE REQUIREMENTS

- Windows
- Jupyter Notebook – Python

1.8 HARDWARE REQUAIREMENTS

- Processor i5 or above
- RAM – 4GB
- Hard Disk – 64 GB
- Input Devices – keyboard and mouse

CHAPTER-2

LITERATURE SURVEY

2.1 SURVEY OF MAJOR AREA RELEVANT TO PROJECT

Predicting fish diseases early using machine learning and water quality assessment is a rapidly evolving field that integrates advances in data science, aquaculture, and environmental monitoring. Below is a survey of the major areas of research, methodologies, and findings in this domain.

2.1.1. Introduction to the Problem Domain

Aquaculture faces significant challenges due to disease outbreaks, which can lead to high mortality rates and economic losses. Traditional disease detection methods are often reactive rather than proactive, relying on visual inspections and lab tests that can be time-consuming and require expert knowledge. The integration of machine learning (ML) with water quality assessment offers a promising solution for early disease detection, enabling timely intervention and improved fish health management.

2.2 KEY AREAS OF RESEARCH

- **Data Collection and Water Quality Monitoring**

Water Quality Parameters: Critical parameters include pH, temperature, dissolved oxygen, ammonia, nitrate, and nitrite levels. Continuous monitoring of these parameters provides a wealth of data for analysis.

Sensors and IoT: Advances in sensor technology and the Internet of Things (IoT) allow for real-time, high-resolution monitoring of water quality, facilitating the collection of large datasets for ML models.

- **Machine Learning Techniques**

Supervised Learning: Commonly used algorithms include decision trees, random forests, support vector machines (SVM), and neural networks. These models are trained on historical data to predict the likelihood of disease outbreaks.

Unsupervised Learning: Techniques like clustering and anomaly detection can identify unusual patterns in water quality data that may indicate the onset of diseases.

- **Feature Engineering and Selection**

Feature Extraction: Creating features such as moving averages, seasonal trends, and rate of change of water quality parameters.

Dimensionality Reduction: Techniques like Principal Component Analysis (PCA) help in reducing the number of features while retaining the most informative ones.

- **Model Evaluation and Validation**

Cross-Validation: Ensures that the model generalizes well to unseen data by splitting the data into training and testing sets multiple times.

Performance Metrics: Metrics such as accuracy, precision, recall, F1 score, and ROC-AUC are used to evaluate model performance.

Real-World Testing: Deploying models in actual aquaculture environments to assess their practical effectiveness and robustness.

2.3 CASE STUDIES AND APPLICATIONS

- **Case Study: Tilapia Farming**

Data Source: Historical water quality data and disease records from tilapia farms.

Methodology: Use of random forests and SVMs to predict outbreaks of bacterial diseases based on water temperature, pH, and ammonia levels.

Results: High predictive accuracy with early warning capabilities, enabling farmers to take preventive actions.

- **Case Study: Salmon Farming**

Data Source: Real-time sensor data from salmon farms.

Methodology: Deep learning models such as RNNs to capture temporal patterns in water quality fluctuations that precede disease outbreaks.

Results: Successful identification of viral and parasitic infections, with models providing alerts several days before clinical signs appear.

- **Challenges and Limitations**

Data Quality and Availability: Ensuring high-quality, continuous data streams from diverse aquaculture environments can be challenging.

Model Generalization: Developing models that perform well across different species, farm setups, and geographic locations.

Real-Time Integration: Building systems that can process and analyze data in real-time without significant delays.

Farmer Adoption: Ensuring that the technology is user-friendly and that farmers trust and utilize the predictions in their daily operations.

- **Future Directions**

Enhanced Sensor Technologies: Development of more sophisticated, cost-effective sensors for broader adoption.

Integration with Farm Management Systems: Creating holistic platforms that integrate disease prediction with other aspects of farm management.

Collaborative Research: Increasing collaboration between data scientists, aquaculture experts, and farmers to refine and improve predictive models.

Adaptive Learning Systems: Developing models that continuously learn and adapt from new data to improve their accuracy over time.

Applications

1.Aquaculture Health Management

Disease Prevention: By predicting disease outbreaks early, fish farmers can implement preventive measures such as adjusting water conditions, administering treatments, or isolating affected fish, thereby reducing mortality rates and economic losses.

Optimized Feeding Practices: Adjust feeding regimes based on water quality data to prevent overfeeding or underfeeding, which can exacerbate disease conditions.

2. Environmental Monitoring

Water Quality Control: Continuous monitoring and analysis of water quality parameters help maintain optimal conditions for fish health, reducing the risk of disease outbreaks due to poor water quality.

Regulatory Compliance: Ensure compliance with environmental regulations by maintaining water quality within acceptable limits, thereby avoiding fines and sanctions.

3. Operational Efficiency

Resource Allocation: Efficiently allocate resources such as medications, manpower, and equipment based on predictive insights, ensuring timely and targeted interventions.

Cost Reduction: Reduce operational costs by minimizing the use of unnecessary treatments and improving overall farm management practices through data-driven decisions.

4. Research and Development

Epidemiological Studies: Use collected data to study the epidemiology of fish diseases, identifying patterns and risk factors associated with outbreaks.

Model Improvement: Continuously improve machine learning models with new data, enhancing the accuracy and reliability of predictions over time.

5. Sustainable Aquaculture Practices

Eco-Friendly Interventions: Implement environmentally friendly interventions by predicting disease outbreaks and taking preventive measures that minimize the need for chemical treatments.

Sustainability Certification: Achieve sustainability certifications by maintaining high standards of fish health and water quality, making the farm more attractive to eco-conscious consumers.

6. Economic Impact

Increased Production: Enhance fish growth and survival rates through improved health management, leading to higher production yields.

Market Advantage: Gain a competitive edge in the market by offering healthier fish, thereby attracting more buyers and potentially commanding higher prices.

7. Education and Training

Farmer Education: Provide fish farmers with valuable insights and training on best practices for maintaining fish health and water quality, leveraging the data and predictions from the system.

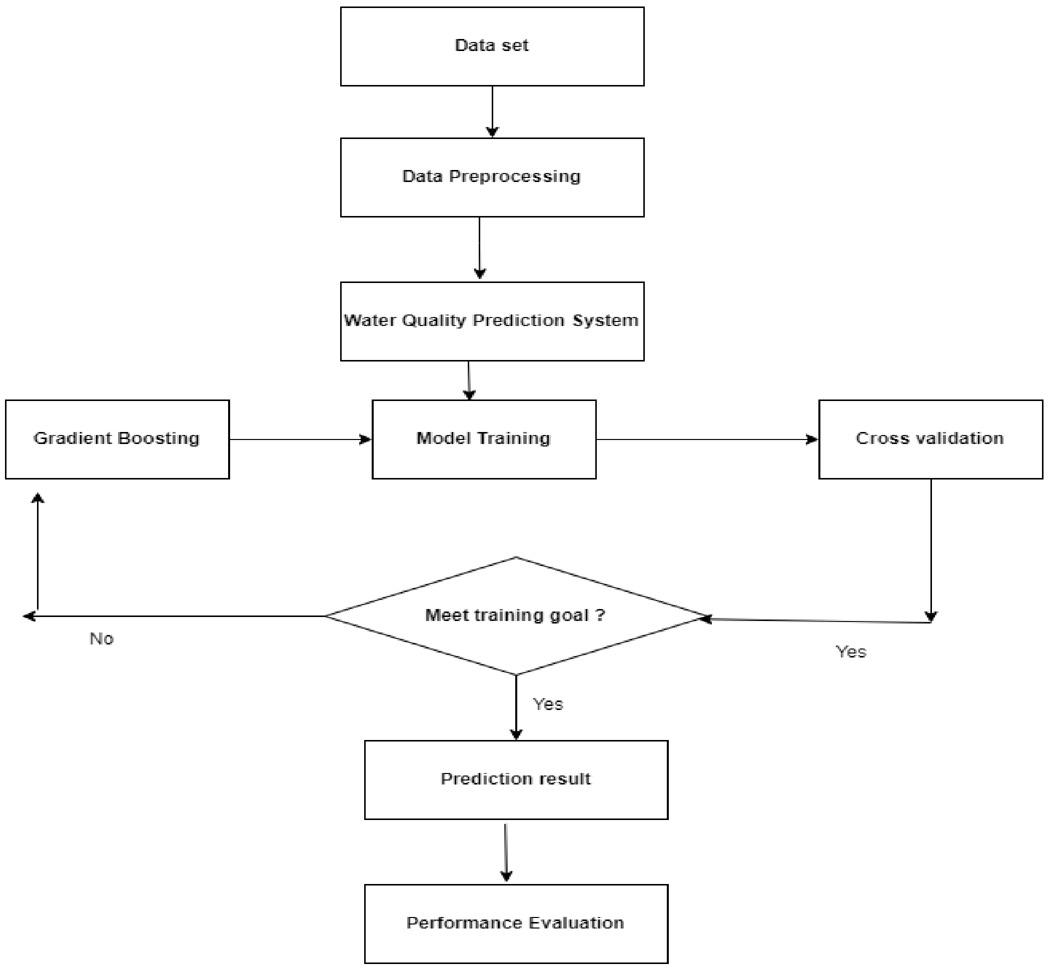
Extension Services: Support extension services by providing them with data-driven tools to advise and assist fish farmers more effectively.

CHAPTER-3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

The system architecture for predicting fish diseases early involves several integrated components, each responsible for a specific function within the overall framework. These components include data collection, data processing, machine learning, real-time prediction, user interface, and monitoring and maintenance. The architecture can be designed to ensure seamless data flow, efficient processing, and user-friendly interaction.



3.1 System Architecture

Data set

Description: The process begins with the collection of a dataset that contains relevant water quality data.

Purpose: This dataset serves as the foundation for the entire predictive modeling process.

Data Preprocessing:

Description: The raw data undergoes preprocessing to clean and prepare it for analysis.

Purpose: This step involves handling missing values, normalizing data, and possibly transforming features to ensure the data is in a suitable format for modelling.

Water Quality Prediction System:

Description: This step involves setting up the system that will be used to predict water quality.

Purpose: It acts as a placeholder for the integration of various components like data preprocessing, model training, and prediction.

Gradient Boosting:

Description: The gradient boosting algorithm is selected for training the predictive model.

Purpose: Gradient boosting is a powerful ensemble learning technique that builds a strong predictive model by combining the outputs of several weaker models.

Model Training:

Description: The gradient boosting model is trained using the pre-processed dataset.

Purpose: This step involves feeding the data into the algorithm to learn patterns and relationships that can be used for making predictions.

Cross Validation:

Description: The model undergoes cross-validation to assess its performance and generalizability.

Purpose: Cross-validation helps in evaluating the model's accuracy and robustness by testing it on different subsets of the data.

Meet Training Goal:

Description: A decision point to check if the model meets the predefined training goals or performance criteria.

Purpose: If the model does not meet the goals, it may need further tuning or retraining.

Prediction Result:

Description: Once the model meets the training goals, it is used to make predictions on new or existing data.

Purpose: This step generates the predicted water quality values.

Performance Evaluation:

Description: The final step involves evaluating the performance of the model.

Purpose: This includes assessing metrics such as accuracy, precision, recall, F1 score, etc., to determine how well the model is performed.

3.2 SYSTEM FLOW

The system flow for predicting fish diseases early using machine learning and water quality assessment begins with the continuous collection of water quality data (e.g., pH, temperature, dissolved oxygen, ammonia, nitrate, and nitrite levels) via sensors placed in the aquaculture environment. This data, along with optional fish health monitoring data, is transmitted through IoT gateways to a central server or cloud platform using real-time data streaming tools. The collected data is then stored in real-time data stores for immediate processing and in long-term storage solutions for historical analysis. The next step involves data preprocessing, which includes cleaning, normalizing, and extracting relevant features from the raw data.

Machine learning models are then developed and trained on this pre-processed data, with hyperparameter tuning and model evaluation ensuring optimal performance. The trained model is applied to incoming data streams for real-time prediction of potential disease outbreaks, with thresholds set to trigger alerts. An alerting system uses notification services to inform farmers and aquaculture managers via SMS, email, or push notifications. A user interface comprising a web-based dashboard and a mobile application provides real-time updates, prediction results, and historical data analysis. Continuous system monitoring and periodic model retraining maintain system accuracy and performance, while a user feedback loop helps refine and improve the system over time.

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized generalpurpose modeling language in the field of object-oriented software engineering.
The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

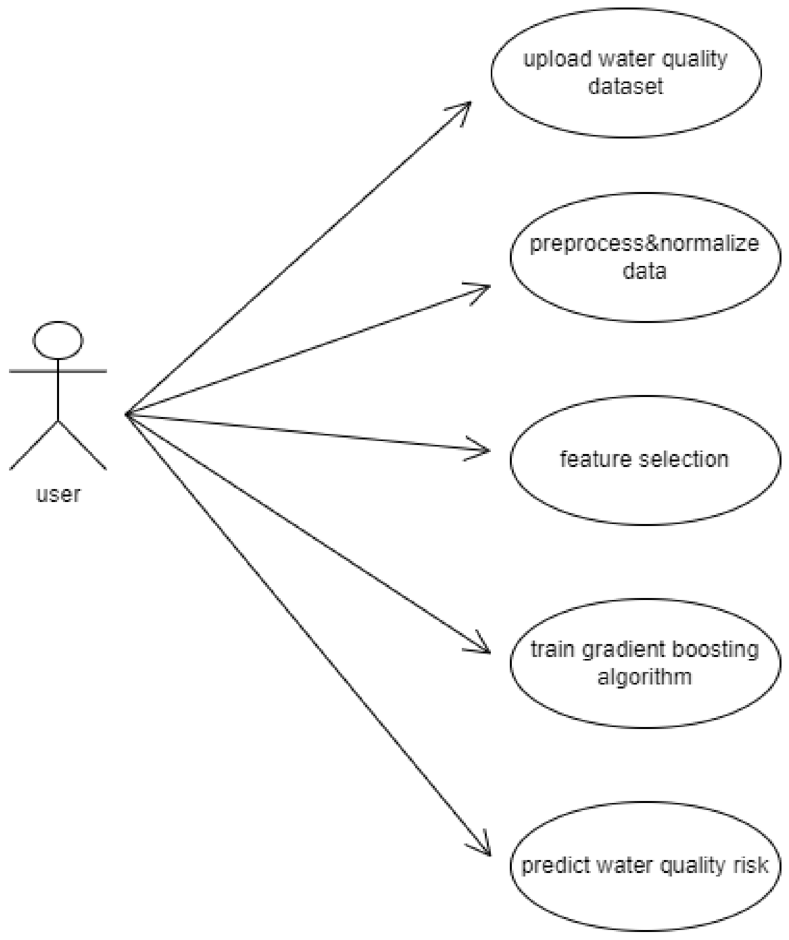
A. USE CASE

A use case diagram is a visual representation of the interactions between users (or actors) and a system to achieve specific goals or tasks. It is one of the Unified Modeling Language (UML) diagrams used in software engineering to illustrate the functional requirements of a system from a user's perspective. In a use case diagram, actors are depicted as stick figures, and use cases are represented by ovals. Arrows indicate the communication or interaction between actors and use cases.

The primary purpose of a use case diagram is to provide a high-level overview of the system's functionality and the actors involved, helping stakeholders understand how the system will be used and how it will interact with its environment. Use case diagrams capture the various use cases or scenarios in which actors interact with the system to accomplish specific tasks. They depict the relationships between actors and use cases, including associations, generalizations, and dependencies**.**

Use case diagrams are essential artifacts in software development, providing a clear and concise representation of system functionality and user interactions. They help ensure that system requirements are well-understood, documented, and communicated among project stakeholders, ultimately contributing to the successful delivery of software projects.

Aquaculture is a vital industry that contributes significantly to global food security and economic development. However, one of the primary challenges facing aquaculture is the frequent occurrence of disease outbreaks, which can lead to massive fish mortality and substantial economic losses. Traditional methods of disease detection often involve manual inspection and laboratory tests, which can be time-consuming and reactive rather than proactive. To address this issue, the integration of machine learning (ML) for fish disease detection through water quality assessment offers a transformative solution.


upload water quality dataset
preprocess&normalize data
user
feature selection
train gradient boosting algorithm
predict water quality risk

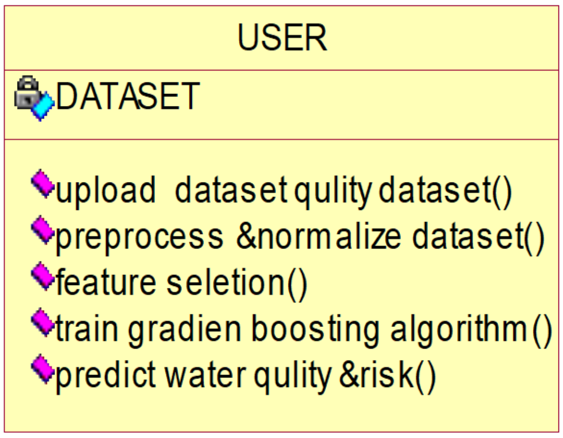
3.2 Use Case

CLASS DIAGRAM

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that illustrates the structure and relationships of classes within a system. It provides a visual representation of the classes, their attributes, methods, and associations, as well as inheritance and other relationships between classes. Class diagrams are widely used in software engineering for designing and modeling object-oriented systems.

In a class diagram, classes are represented as rectangles with three compartments: the top compartment contains the class name, the middle compartment lists the class attributes (data fields), and the bottom compartment shows the class methods (operations or functions). Arrows indicate associations between classes, with multiplicity annotations indicating the cardinality of the relationship. Inheritance relationships are depicted using solid line arrows with a triangular arrowhead, showing the subclass inheriting from the superclass.

Class diagrams play a crucial role in software engineering, providing a clear and concise representation of a system's structure and relationships. They serve as a foundation for designing and implementing object-oriented systems, ensuring that the system meets its requirements and functions as intended


USER
DATASET
upload dataset qulity dataset()
preprocess &normalize dataset()
feature seletion()
train gradien boosting algorithm()
predict water qulity &risk()

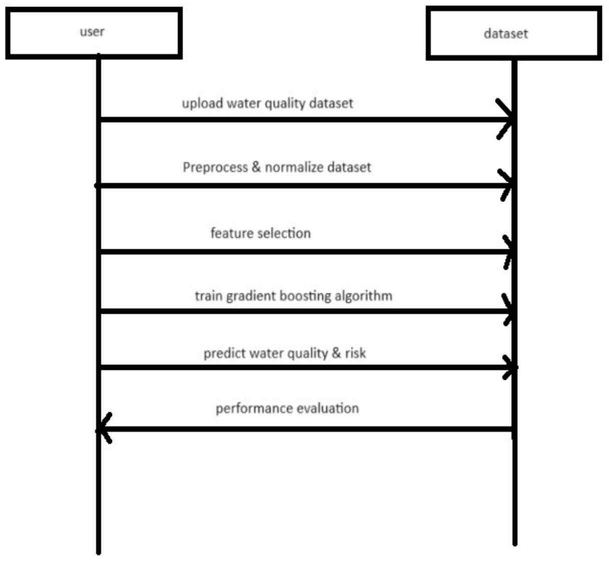
3.3 Class Diagram

SEQUENCE DIAGRAM

In a sequence diagram, objects are represented as boxes or rectangles, with the object's name written inside. Vertical lines, known as lifelines, represent the lifespan of each object during the scenario. Arrows between lifelines represent messages sent between objects, with the arrowhead indicating the direction of the message. Messages can be synchronous (indicated by a solid arrow) or asynchronous (indicated by a dashed arrow), and they can include parameters, return values, and other annotations.

The primary purpose of a sequence diagram is to provide a visual representation of the dynamic interactions between objects or components within a system. Sequence diagrams help stakeholders, developers, and designers understand the flow of control and communication between objects during the execution of a particular scenario or use case. They are particularly useful for modeling complex interactions, concurrency, and message passing between objects.

Sequence diagrams play a crucial role in software engineering, providing a clear and concise representation of the dynamic interactions between objects or components within a system. They help ensure that the system meets its functional requirements and behaves as intended during execution.


user
dataset
upload water quality dataset
Preprocess & normalize dataset
feature selection
train gradient boosting algorithm
predict water quality & risk
performance evaluation

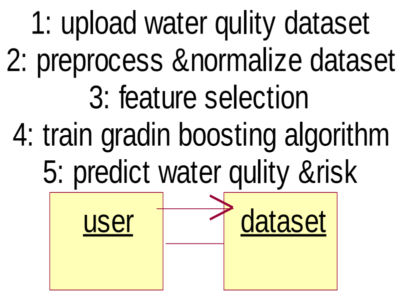
3.4 Sequence Diagram

COLLABRATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is a type of interaction diagram in the Unified Modeling Language (UML) that illustrates the interactions and relationships between objects or components within a system. Collaboration diagrams focus on the structural organization of objects and the messages exchanged between them to achieve specific tasks or scenarios. They emphasize the flow of communication and collaboration between objects rather than the sequence of messages over time, making them particularly useful for modeling complex interactions and relationships in a system.

In a collaboration diagram, objects are represented as rectangles or squares, with the object's name written inside. Lines connecting objects represent messages or communications between them, and arrows indicate the direction of the message flow. Labels on the lines may include the message name, parameters, return values, and other annotations. Unlike sequence diagrams, collaboration diagrams do not explicitly show the sequence of messages over time but instead focus on illustrating the relationships and interactions between objects in a static snapshot.

Collaboration diagrams play a crucial role in software engineering, providing a clear and concise representation of the structural organization and communication patterns between objects within a system. They help ensure that the system meets its functional requirements and supports the desired interactions between objects.


1: upload water qulity dataset
2: preprocess &normalize dataset
3: feature selection
4: train gradin boosting algorithm
5: predict water qulity &risk
user
dataset

3.5 Collaboration Diagram

3.3 MODULE DESCRIPTION

The system for predicting fish diseases early using machine learning and water quality assessment is composed of several interrelated modules, each serving a specific function from data collection to user interaction. This modular architecture ensures that the system is scalable, maintainable, and capable of providing real-time, actionable insights to aquaculture farmers and managers.

1. Data Collection Module

Function: Gather water quality and fish health data.

Water Quality Sensors: Devices that measure pH, temperature, dissolved oxygen, ammonia, nitrate, and nitrite levels.

Health Monitoring Devices (optional): Cameras or biosensors to monitor fish behaviour and health indicators.

IoT Gateways: Aggregate data from sensors and transmit it to a central server or cloud platform.

2. Data Transmission Module

Function: Transmit collected data to the processing system.

Real-Time Data Streams: Use data streaming tools like Apache Kafka or AWS Kinesis for continuous data flow.

Batch Data Transfer: For historical data, use batch processing tools to transfer data to the central repository.

3. Data Storage Module

Function: Store the collected data for processing and analysis.

Real-Time Data Store: In-memory databases (e.g., Redis) for immediate data processing.

Long-Term Storage: Data lakes (e.g., AWS S3) or traditional databases (e.g., PostgreSQL) for storing historical data.

4. Data Processing and Preprocessing Module

Function: Prepare data for machine learning analysis.

Data Cleaning: Remove errors, handle missing values, and smooth outliers.

Data Normalization: Standardize data to ensure consistency.

Feature Extraction: Create new features (e.g., moving averages, seasonal trends) from raw data.

5. Machine Learning Model Development Module

Function: Develop and train machine learning models to predict fish diseases.

Model Selection: Choose appropriate machine learning algorithms (e.g., decision trees, random forests, SVM, neural networks).

Model Training: Train models using historical data and optimize them with cross-validation techniques.

Hyperparameter Tuning: Adjust model parameters to improve performance.

Model Evaluation: Test models using metrics like accuracy, precision, recall, F1 score, and ROC-AUC to select the best model.

6. Real-Time Prediction Module

Function: Apply trained models to incoming data for real-time disease prediction.

Real-Time Processing: Use real-time processing frameworks (e.g., Apache Storm, Spark Streaming) to handle continuous data streams.

Prediction Engine: Apply the trained machine learning model to incoming data to predict potential disease outbreaks.

7. Alerting System Module

Function: Notify users about potential disease outbreaks.

Threshold Setting: Define thresholds for prediction scores to trigger alerts.

Notification Services: Use services like AWS SNS or Twilio to send alerts via SMS, email, or push notifications.

8. User Interface Module

Function: Provide a platform for users to view data and alerts.

Dashboard: A web-based interface displaying real-time water quality data, prediction results, and alert notifications.

Mobile Application: A mobile app offering real-time updates, alerts, and access to historical data analysis.

9. Monitoring and Maintenance Module

Function: Ensure the system operates efficiently and accurately over time.

System Monitoring: Use tools like Prometheus and Grafana to track system performance, data flow, and resource usage.

Model Retraining: Regularly update models with new data to maintain prediction accuracy

CHAPTER-4
IMPLEMENTATION

4.1 ENVIRONMENTAL SETUP

A) PYTHON

Python is a versatile and widely-used programming language known for its simplicity, readability, and extensive ecosystem. Developed in the late 1980s by Guido van Rossum, Python has since become one of the most popular programming languages worldwide. Its straightforward syntax and easy-to understand code make it an ideal choice for beginners learning to code, while its powerful features and extensive libraries cater to the needs of experienced developers across diverse domains. Python's versatility is evident in its support for multiple programming paradigms, including procedural, object-oriented, and functional programming, making it suitable for a wide range of applications such as web development, data analysis, scientific computing, artificial intelligence, machine learning, automation, and more. The language's interpreted nature allows for rapid development and prototyping, while its portability ensures compatibility across various platforms and systems. Python's vibrant community, encompassing developers, educators, and enthusiasts, fosters collaboration, innovation, and knowledge-sharing, further fuelling its growth and adoption. With its open-source nature, extensive documentation, and active support, Python continues to evolve and thrive as a cornerstone of modern software development. Here are some key features and functionalities of python:

1. Simple and Readable Syntax: Python's syntax is designed to be clean, readable, and concise, making it easy for developers to write and understand code.

2. Interpreted Language: Python is interpreted rather than compiled, allowing for interactive development and rapid prototyping. Code can be executed line by line, facilitating debugging and experimentation.

3. Dynamic Typing: Python is dynamically typed, meaning that variable types are inferred at runtime, reducing the need for explicit type declarations and enhancing code flexibility.

4. High-level Data Structures: Python provides built-in support for high-level data structures such as lists, dictionaries, tuples, and sets, simplifying complex data manipulation tasks.

5. Extensive Standard Library: Python comes with a comprehensive standard library containing modules and functions for a wide range of tasks, from file I/O and networking to string manipulation and data processing.

6. Object-Oriented Programming: Python supports object-oriented programming (OOP) paradigms, allowing developers to create classes and objects with encapsulation, inheritance, and polymorphism.

Advantages of Python Over Other Languages:

1. Less Coding Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support. The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python:

While Python is a powerful and versatile programming language with numerous advantages, it also has some disadvantages:

1. Python is an interpreted language, which means that it is generally slower than compiled languages like C or C++. This can be a disadvantage in applications where performance is critical, such as high-frequency trading or intensive numerical computations.

2. Python's Global Interpreter Lock can limit the execution of multiple threads within a single Python process, potentially hindering performance in multi-threaded applications that require parallel processing.

3. Python can be memory-intensive, particularly for large-scale applications or when dealing with large datasets. This can lead to higher memory usage compared to other languages, which may be a concern for resource-constrained environments.

B) IDLE (Integrated Development and Learning Environment)

Integrated Development and Learning Environment, is a versatile and user-friendly integrated development environment (IDE) specifically designed for Python programming. Available across various platforms including Windows, macOS, and Linux, IDLE comes bundled with the standard Python installation. Its intuitive interface offers a robust set of features tailored to both novice learners and experienced developers. Key functionalities include a built-in code editor equipped with syntax highlighting, auto-indentation, and code completion capabilities for efficient code writing. IDLE's interactive Python shell enables real time code execution, allowing users to experiment, test functions, and explore Python libraries interactively. Additionally, IDLE boasts a powerful debugger 27 facilitating error detection and resolution through features like breakpoints and variable inspection. With integrated documentation providing instant access to Python resources and customization options for personalizing themes and settings, IDLE serves as an accessible and efficient tool for Python development and learning endeavours.

Here are some key features and functionalities of IDLE:

- **Code Editor:**
  IDLE includes a built-in code editor with syntax highlighting, auto-indentation, and code completion features. The editor provides a convenient environment for writing Python code, with support for multiple tabs and the ability to customize the appearance and behaviour of the editor.
- **Debugger:**
  IDLE includes a built-in debugger that allows developers to debug Python code easily. The debugger provides features such as breakpoints, stepping through code execution, inspecting variables, and evaluating expressions, helping developers identify and fix errors in their code efficiently.
- **Syntax Highlighting and Auto-completion:**
  IDLE supports syntax highlighting, which colorizes different elements of the code to improve readability and identify syntax errors. It also provides auto-completion suggestions, helping developers quickly access and insert Python keywords, functions, and variable names while typing.
- **Integrated Documentation:**
  IDLE integrates Python's documentation directly into the IDE, allowing developers to access help and documentation for Python functions, modules, and libraries without leaving the development environment. This feature enhances productivity by providing instant access to relevant documentation and resources.
- **File Management:**
  IDLE provides basic file management features, such as opening, saving, and organizing Python scripts. It also supports file navigation and searching within the codebase, making it easy for developers to work with multiple files and projects.

Advantages of IDLE:

1. **Beginner-Friendly:**
   IDLE is specifically designed to be easy to use for beginners who are learning Python. It provides a simple and intuitive interface with features

like syntax highlighting and code completion, which can help users write code more effectively.

2. **Integrated Development Environment (IDE):**
   IDLE is a full-fledged integrated development environment that includes features like code editor, interactive interpreter, debugger, and file browser, making it a convenient choice for writing, testing, and debugging Python code in one environment.
3. **Cross-Platform Compatibility:**
   IDLE is available on multiple platforms, including Windows, macOS, and Linux, making it accessible to users regardless of their operating system.
4. **Included with Python Distribution:**
   IDLE comes bundled with the standard Python distribution, which means that users do not need to install additional software to start coding in Python. This makes it easily accessible to anyone who has installed Python on their system.
5. **Support for Python Shell:**
   IDLE provides an interactive Python shell, allowing users to execute Python code line by line and see the results immediately. This can be helpful for experimenting with code snippets or learning Python concepts interactively.

Disadvantages:

1. **Limited Features:**
   Compared to more advanced IDEs like PyCharm or Visual Studio Code, IDLE may lack certain features such as advanced debugging tools, version control integration, and extensive plugin support. This can limit its usefulness for more complex or professional development projects.
2. **Performance:**
   While IDLE is sufficient for writing and testing small to medium-sized Python scripts, it may not be as efficient or performant as other IDEs, particularly for larger projects or resource-intensive tasks.

3. **User Interface:**
   Some users may find IDLE's user interface to be outdated or less visually appealing compared to other modern IDEs. The interface may lack customization options or modern design elements, which can affect user experience for some developers.
4. **Dependency on Python Installation:**
   Since IDLE is bundled with the standard Python distribution, updates or improvements to IDLE may be tied to Python releases. Users may not have access to the latest features or bug fixes until they update their Python installation.
5. **Limited Extensibility:**
   While IDLE supports basic customization and extension through user-defined scripts and plugins, it may not offer the same level of extensibility or flexibility as other IDEs with robust plugin ecosystems. Users looking for extensive customization options may find IDLE lacking in this regard.

C) COMMAND PROMPT

Command Prompt, often referred to as CMD or Command Line Interface (CLI), is a text-based interface used in Windows operating systems to execute commands and perform various system tasks. It provides users with a powerful tool for interacting with the operating system and executing commands directly through a command-line interface. Command Prompt allows users to navigate the file system, manage files and directories, run programs, configure system settings, and perform administrative tasks using a set of predefined commands and utilities. While Command Prompt may appear intimidating to some users due to its text-based nature, it offers several advantages, including greater control over system operations, the ability to automate tasks using batch scripts, and efficient troubleshooting of system issues. Additionally, Command Prompt provides access to a wide range of command-line tools and utilities built into the

Windows operating system, enabling users to perform advanced system management tasks and administrative functions.

Features of Command Prompt:

1. Text-based Interface: Command Prompt provides a text-based interface for executing commands and interacting with the operating system. Users input commands via a command-line interface and receive text-based output.

2. Wide Range of Commands: Command Prompt supports a vast array of commands for performing various tasks, including file management, system configuration, network administration, process management, and more. These commands enable users to manipulate files, run programs, manage services, and configure system settings.

3. Batch Scripting: Command Prompt allows users to create and execute batch scripts, which are sequences of commands stored in a text file with a .bat extension. Batch scripts enable users to automate repetitive tasks, perform complex operations, and streamline system administration tasks.

4. File Management: Command Prompt provides commands for navigating the file system, creating, copying, moving, and deleting files and directories. Users can perform file operations, such as renaming files, changing file attributes, and searching for files using wildcard characters.

5. System Configuration: Command Prompt provides commands for configuring system settings, modifying environment variables, managing user accounts, and configuring system services. Users can customize system behaviour, set environment variables, and manage system resources using command-line tools.

Advantages:

1.Efficiency: Command prompt allows for quick and efficient execution of tasks through text-based commands, often faster than navigating through graphical user interfaces (GUIs) with a mouse.

2. Automation: Command prompt enables users to automate repetitive tasks and create batch scripts to execute multiple commands sequentially, saving time and effort.

3. Remote Management: Command prompt supports remote management of computers and servers via SSH (Secure Shell) or remote desktop protocols, allowing administrators to perform administrative tasks remotely.

4. Low Resource Usage: Command prompt consumes minimal system resources compared to GUI-based applications, making it suitable for use on low-powered or resource-constrained systems.

5.Access to Advanced Functions: Command prompt provides access to advanced system functions, configuration settings, and diagnostic tools that may not be readily available through GUIs.

6. Scripting Support: Command prompt supports scripting languages like batch scripting (Windows) or shell scripting (Unix/Linux), allowing users to create complex scripts for automated tasks or system administration.

Disadvantages:

1. Limited Discoverability: Unlike GUIs, which often provide visual cues and menus for discovering features and options, command prompt requires users to know the commands or consult documentation to access functionality.

2. Error-Prone: Command prompt commands are often case-sensitive and require precise syntax, increasing the risk of errors, typos, and unintended consequences, especially for complex commands.

3. Limited Multimedia Support: Command prompt may have limited support for multimedia tasks such as video playback, image editing, or graphical applications, which are more suited to GUI environments.

4. Security Concerns: Command prompt access can pose security risks if not properly secured, as it provides direct access to system functions and configuration settings, which could be exploited by unauthorized users.

D) LIBRARIES

1) TensorFlow

TensorFlow is free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache2.0 open-source license on November 9, 2015.

2) NumPy

NumPy, short for Numerical Python, is a fundamental library for numerical computing in Python. It provides support for multidimensional arrays, matrices, and mathematical functions, making it essential for scientific computing and data analysis. NumPy's array operations are significantly faster than traditional Python lists. It is the fundamental package for scientific computing with Python. It contains various features including these important ones

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

3) Pandas

Pandas is a powerful Python library widely used for data manipulation and analysis. It provides data structures like Data Frames and Series, which allow users to handle structured data efficiently. Pandas simplifies tasks such as data cleaning, transformation, aggregation, and visualization. It also offers functionality for reading and writing data from various file formats, including CSV, Excel, SQL databases, and more. With its intuitive and versatile API, Pandas is a go-to tool for data scientists, analysts, and researchers working with tabular data.

Pandas is built on top of NumPy and provides high-level data structures like Data Frame and Series, which offer more flexibility and functionality compared to NumPy arrays. Data Frames are two-dimensional labelled data structures with columns of potentially different data types, similar to a spreadsheet or SQL table. Series, on the other hand, are one-dimensional labelled arrays that can hold any data type. Pandas offers powerful data manipulation capabilities, including indexing, slicing, filtering, merging, and reshaping data, making it an indispensable tool for data analysis and manipulation tasks.

4) Matplotlib

Matplotlib is a comprehensive library for creating static, interactive, and publication-quality visualizations in Python. It offers a wide range of plotting functions and customization options for creating plots, charts, histograms, scatterplots, and more. Matplotlib's pyplot module provides a MATLAB-like interface for generating plots with minimal code, making it accessible to users of all levels. Additionally, Matplotlib integrates seamlessly with NumPy and Pandas, allowing users to visualize data stored in arrays or Data Frames effortlessly.

Matplotlib is highly customizable and offers a wide range of plot styles, colors, markers, and line styles to create visually appealing and informative

plots. It supports interactive features like zooming, panning, and saving plots in various file formats. Matplotlib can be used in various contexts, including scientific publications, data analysis reports, web applications, and interactive data exploration environments. Additionally, Matplotlib can be combined with other libraries like Seaborn and Plotly to create even more sophisticated visualizations.

5) Scikit – learn

Scikit-learn is a popular open-source machine learning library for Python that provides simple and efficient tools for data analysis and modeling, built on top of other libraries like NumPy, SciPy, and matplotlib. It offers a user-friendly API that makes it accessible to both beginners and experienced users, supporting a wide range of machine learning algorithms including supervised learning (e.g., Linear Regression, Support Vector Machines), unsupervised learning (e.g., K-means, DBSCAN, Principal Component Analysis), and model selection and evaluation tools (e.g., cross-validation).

6) Tkinter

Tkinter is Python's standard library for creating graphical user interfaces (GUIs). It provides a set of modules and classes for building desktop applications with rich GUI elements, including buttons, menus, text boxes, and canvas widgets. Tkinter's simplicity and ease of use make it an excellent choice for developing lightweight and cross-platform desktop applications. Despite its simplicity, Tkinter offers extensive customization options and support for event-driven programming, allowing developers to create interactive and responsive user interfaces.

Tkinter's integration with other Python libraries, such as Matplotlib for plotting and Pandas for data visualization, further enhances its utility for building data-driven desktop applications. Tkinter is based on the Tk GUI toolkit, which is a cross-platform toolkit widely used for building graphical user interfaces. Tkinter provides a set of widgets (GUI components) that can be arranged and configured to create interactive applications. It supports event-driven programming, allowing developers to respond to user actions like button clicks, mouse movements, and keypresses. Tkinter applications can be easily deployed on different platforms without requiring additional dependencies, making it a convenient choice for developing lightweight desktop applications in Python.

E) SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTING

1) **UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2) **INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program.

Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3) **FUNCTIONAL TESTING**

Functional testing is a type of software testing that focuses on verifying that the software functions according to specified requirements and performs its intended operations. It involves testing the system's features and functions by providing appropriate inputs and examining the outputs, ensuring the software behaves as expected. Key aspects of functional testing include validating user interfaces, APIs, databases, security, and client/server applications. This type of testing is typically black-box, meaning testers do not need to know the internal code structure. It encompasses various testing types, such as unit testing, integration testing, system testing, and acceptance testing. Functional testing is essential for identifying defects and ensuring that the software meets its functional specifications before release. Test cases are usually derived from functional requirements or specifications documents. Organization and preparation of functional tests is focused

on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4) WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level. White box testing, also known as clear or glass box testing, is a software testing method where the internal structure, design, and implementation of the item being tested are known to the tester. This approach allows testers to verify the flow of inputs and outputs through the code, improve code coverage, and ensure all paths and branches are executed.

5) BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6) ACCEPTANCE TESTING

Acceptance testing is a type of software testing performed to determine whether a system meets the required specifications and is ready for deployment. It is typically conducted by end-users or clients to validate the system's functionality, usability, and performance in real-world scenarios. The primary goal of acceptance testing is to ensure the software delivers the expected outcomes and satisfies business needs. Common types include User Acceptance Testing (UAT), Operational Acceptance Testing (OAT), and Contract Acceptance Testing. Successful completion of acceptance testing signifies that the software is ready for production.

4.2 IMPLEMENTATION OF MODULES

A) User Module

The user module in the fish disease detection system is designed to provide a seamless and user-friendly interface for end-users to interact with the application. It allows users to sign up for an account, log in, and manage their profiles securely. Once logged in, users can submit water quality data, which includes parameters such as temperature, pH, dissolved oxygen, ammonia, nitrites, and nitrates. This data is then used by the machine learning model to predict the presence of fish diseases.

Users can view their submitted data and the corresponding predictions, helping them to monitor and manage the health of their fish.

B) Admin Module

The admin module is a comprehensive interface designed for administrators to manage the system and oversee user activities. Admins have the capability to manage user accounts, including adding new users, editing existing user details, and deleting users if necessary. They can view all submitted water quality data and the corresponding disease predictions, providing a holistic view of the system's usage and performance. Additionally, admins can retrain the machine learning model with new data to improve its

accuracy and reliability over time. The module also includes monitoring tools to keep track of system performance and ensure that the application runs smoothly.

4.3 INTEGRATION AND DEVELOPMENT

Integration and development are fundamental aspects of software engineering, encompassing the processes of combining disparate components and creating new solutions to address specific needs. Integration focuses on combining various components into a cohesive system. This includes setting up data ingestion pipelines where sensors collect water quality data, ensuring preprocessing and feature engineering steps are automated, and deploying the machine learning model through APIs or microservices for real-time predictions. Additionally, user interfaces and dashboards are integrated to provide end-users, such as fish farmers and researchers, with access to real-time data, predictions, and alerts.

Development, on the other hand, involves creating and refining these components. This starts with setting up sensors and systems for data collection and writing code for data preprocessing to clean and normalize the data. Feature engineering follows, creating new features that better capture patterns related to fish health. The core of development is model building, where appropriate machine learning algorithms are selected, trained on historical data, and optimized. Model evaluation is carried out using metrics like accuracy and precision, with cross-validation procedures to ensure robust performance. Finally, feedback and integrate it into the model retraining process, ensuring continuous improvement. The combined process of integration and development ensures that the system not only operates efficiently in a production environment but also adapts and improves over time, providing accurate, real-time fish disease detection and valuable insights for aquaculture management.

CHAPTER-5

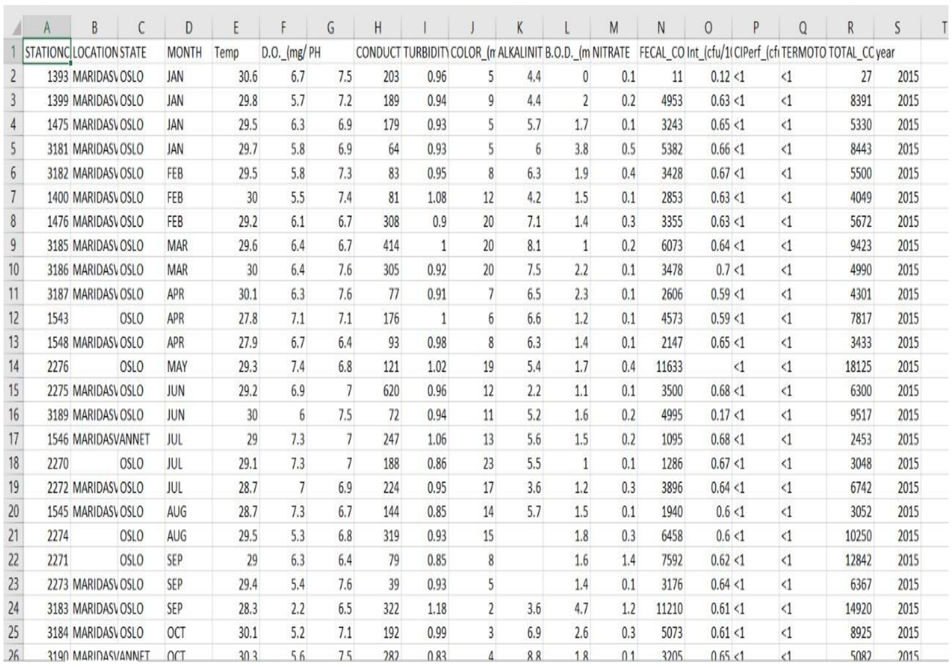
EVALUATION

5.1 DATASETS

- A dataset is a structured collection of data, typically organized in a tabular format consisting of rows and columns.
- Each row represents a single observation or record, while each column represents a specific attribute or variable. Datasets are used for a variety of purposes including statistical analysis, machine learning, and data visualization.
- Datasets play a crucial role in the development and evaluation of data-driven models and algorithms.
- They can be collected through various means, such as surveys, experiments, sensors, web scraping, or manual entry.
- Properly curated and well-structured datasets are essential for reliable and meaningful analysis and model training.
- A database is an organized collection of data stored as multiple datasets. It is a collection of numbers or values that relate to a particular subject.
- Dataset plays a crucial role in development and evaluation of data-driven models and algorithms.
- A dataset represents a set of observations or records related to a particular topic, often used for analysis and modelling in various fields.
- The Kaggle is popular for providing quality, meaningful and efficient datasets.
- Each record in the dataset features a unique observation ID, the date of the observation, the specific location or farm where the data was collected, and the species of fish observed.
- The dataset includes temperature (measured in degrees Celsius), pH level, dissolved oxygen (DO) in milligrams per litre, ammonia (NH3) concentration, nitrite (NO2) concentration, nitrate (NO3) concentration, salinity in parts per thousand, water hardness in milligrams per litre, turbidity in Nephelometric Turbidity Units (NTU), conductivity (measured

in micro siemens per centimetres, μS/cm), and alkalinity in milligrams per litre.

- The data set contains 15 attributes and 1991 records.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	STATIONC	LOCATION	STATE	MONTH	Temp	D.O._(mg/	PH	CONDUCT	TURBIDITY	COLOR_(n	ALKALINIT	B.O.D._(m	NITRATE	FECAL_CO	Int_(cfu/1(	CIPerf_(cf	TERMOTO	TOTAL_CC	year	
2	1393	MARIDASV	OSLO	JAN	30.6	6.7	7.5	203	0.96	5	4.4	0	0.1	11	0.12	<1	<1	27	2015	
3	1399	MARIDASV	OSLO	JAN	29.8	5.7	7.2	189	0.94	9	4.4	2	0.2	4953	0.63	<1	<1	8391	2015	
4	1475	MARIDASV	OSLO	JAN	29.5	6.3	6.9	179	0.93	5	5.7	1.7	0.1	3243	0.65	<1	<1	5330	2015	
5	3181	MARIDASV	OSLO	JAN	29.7	5.8	6.9	64	0.93	5	6	3.8	0.5	5382	0.66	<1	<1	8443	2015	
6	3182	MARIDASV	OSLO	FEB	29.5	5.8	7.3	83	0.95	8	6.3	1.9	0.4	3428	0.67	<1	<1	5500	2015	
7	1400	MARIDASV	OSLO	FEB	30	5.5	7.4	81	1.08	12	4.2	1.5	0.1	2853	0.63	<1	<1	4049	2015	
8	1476	MARIDASV	OSLO	FEB	29.2	6.1	6.7	308	0.9	20	7.1	1.4	0.3	3355	0.63	<1	<1	5672	2015	
9	3185	MARIDASV	OSLO	MAR	29.6	6.4	6.7	414	1	20	8.1	1	0.2	6073	0.64	<1	<1	9423	2015	
10	3186	MARIDASV	OSLO	MAR	30	6.4	7.6	305	0.92	20	7.5	2.2	0.1	3478	0.7	<1	<1	4990	2015	
11	3187	MARIDASV	OSLO	APR	30.1	6.3	7.6	77	0.91	7	6.5	2.3	0.1	2606	0.59	<1	<1	4301	2015	
12	1543		OSLO	APR	27.8	7.1	7.1	176	1	6	6.6	1.2	0.1	4573	0.59	<1	<1	7817	2015	
13	1548	MARIDASV	OSLO	APR	27.9	6.7	6.4	93	0.98	8	6.3	1.4	0.1	2147	0.65	<1	<1	3433	2015	
14	2276		OSLO	MAY	29.3	7.4	6.8	121	1.02	19	5.4	1.7	0.4	11633		<1	<1	18125	2015	
15	2275	MARIDASV	OSLO	JUN	29.2	6.9	7	620	0.96	12	2.2	1.1	0.1	3500	0.68	<1	<1	6300	2015	
16	3189	MARIDASV	OSLO	JUN	30	6	7.5	72	0.94	11	5.2	1.6	0.2	4995	0.17	<1	<1	9517	2015	
17	1546	MARIDASVANNET		JUL	29	7.3	7	247	1.06	13	5.6	1.5	0.2	1095	0.68	<1	<1	2453	2015	
18	2270		OSLO	JUL	29.1	7.3	7	188	0.86	23	5.5	1	0.1	1286	0.67	<1	<1	3048	2015	
19	2272	MARIDASV	OSLO	JUL	28.7	7	6.9	224	0.95	17	3.6	1.2	0.3	3896	0.64	<1	<1	6742	2015	
20	1545	MARIDASV	OSLO	AUG	28.7	7.3	6.7	144	0.85	14	5.7	1.5	0.1	1940	0.6	<1	<1	3052	2015	
21	2274		OSLO	AUG	29.5	5.3	6.8	319	0.93	15		1.8	0.3	6458	0.6	<1	<1	10250	2015	
22	2271		OSLO	SEP	29	6.3	6.4	79	0.85	8		1.6	1.4	7592	0.62	<1	<1	12842	2015	
23	2273	MARIDASV	OSLO	SEP	29.4	5.4	7.6	39	0.93	5		1.4	0.1	3176	0.64	<1	<1	6367	2015	
24	3183	MARIDASV	OSLO	SEP	28.3	2.2	6.5	322	1.18	2	3.6	4.7	1.2	11210	0.61	<1	<1	14920	2015	
25	3184	MARIDASV	OSLO	OCT	30.1	5.2	7.1	192	0.99	3	6.9	2.6	0.3	5073	0.61	<1	<1	8925	2015	
26	3190	MARIDASVANNET		OCT	30.3	5.6	7.5	282	0.83	4	8.8	1.8	0.1	3205	0.65	<1	<1	5082	2015	

5.1 Dataset

A) DATA COLLECTION

Data collection is a fundamental process that involves systematically gathering information to form a solid foundation for analysis, decision-making, and model training. This process employs various methods, such as surveys and questionnaires for quantitative data, interviews for qualitative insights, and observations for recording behaviours or conditions. Data collection involves systematically gathering detailed information on various water quality parameters within aquaculture environments. Key parameters include pH levels, temperature, dissolved oxygen, ammonia, nitrate, and nitrite concentrations, as well as the presence of pathogens or pollutants. This data is obtained through multiple methods such as deploying automated sensors and IoT devices in fish tanks or ponds for real-time monitoring, conducting laboratory analyses of water samples for precise chemical and biological measurements, and integrating historical records from aquaculture operations to capture long-term trends. Additionally, environmental data like weather conditions and seasonal variations, as well as farm management data including feeding schedules and cleaning routines, are collected to provide a comprehensive understanding of factors affecting fish health.

B) PREPROCESSING THE DATASET

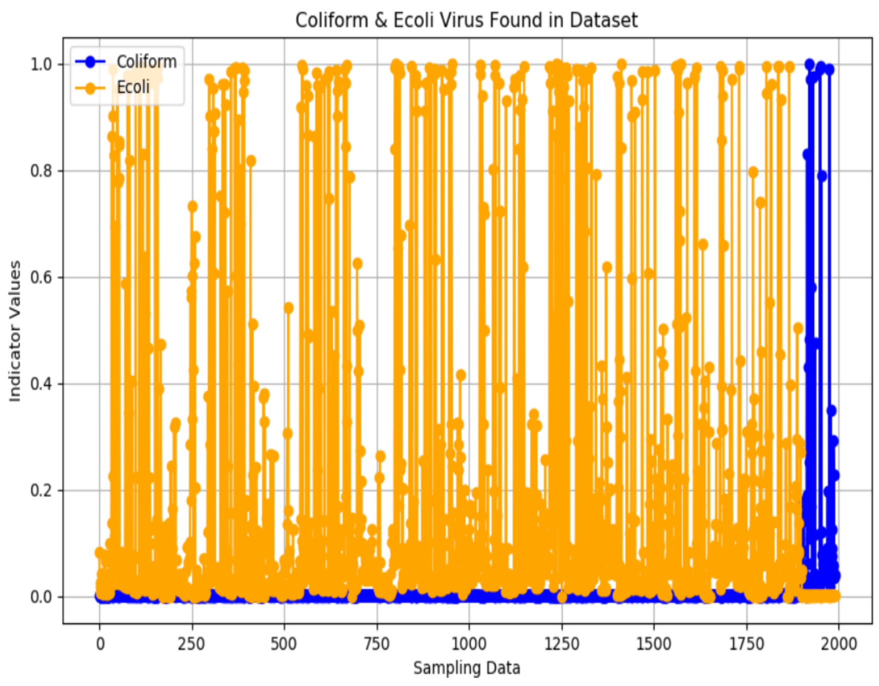
Preprocessing refers to a series of steps applied to raw data to transform it into a clean and usable format for analysis or modelling. This stage is crucial in any data-driven project as it ensures the quality and consistency of the data, ultimately leading to more reliable and accurate results. Preprocessing a dataset involves several key steps to ensure the data is ready for analysis and model training. First, data collection is performed to gather relevant water quality parameters such as pH, temperature, dissolved oxygen, ammonia levels, and other chemical concentrations. Once collected, the dataset often contains missing values, outliers, and noise, which need to be addressed. Missing values can be handled by imputation using mean or median values, while outliers can be detected using statistical methods and treated or removed to prevent skewed results. Noise in the data may be smoothed through techniques like moving averages. Next, the data undergoes transformation to make it suitable for machine learning algorithms. Numerical features are typically normalized or standardized to ensure they are on a comparable scale, which can improve the performance of many machine learning models.

C) FEATURE SELECTION

Selecting the appropriate algorithm for model building is a crucial step in machine learning, as it can significantly influence the model's performance, efficiency, and generalizability. Select gradient boosting algorithm and train it. The dataset should be split into a training set (80%) and a testing set (20%) to ensure that the model evaluation is conducted on unseen data, preserving the integrity of the performance metrics.

D) DISEASE PREDICTION

After training the model predicts fish disease. The model identifies whether the fish contains coli form and Ecoli forms of bacteria.



5.2 Virus found

EVALUATION METRICS

Evaluation metrics are crucial for assessing the effectiveness of the models. Commonly used metrics include accuracy, which measures the overall correctness of predictions, precision, which quantifies the proportion of correctly predicted disease cases among all predicted cases, recall, which calculates the proportion of correctly predicted disease cases among all actual cases. In addition to accuracy, precision, recall, F1-score, the evaluation of fish disease detection using machine learning based on water quality assessment often involves the use of a confusion matrix. The confusion matrix provides a tabular representation of the model's predictions compared to the actual outcomes, categorizing instances into true positives, true negatives, false positives, and false negatives. This matrix allows for a deeper understanding of the model's performance by quantifying the types of errors made, such as misclassifications of healthy fish as diseased or vice versa. Analysing the confusion matrix alongside other metrics helps identify areas for improvement and fine-tuning of the model to enhance its accuracy and effectiveness in detecting fish diseases based on water quality parameters.

Evaluation metrics serve as the cornerstone of assessing the efficacy and performance of machine learning models, particularly in the domain of fish disease detection through water quality assessment. In the realm of aquaculture management, where precision and reliability are paramount, the judicious selection and interpretation of evaluation metrics are instrumental in gauging the effectiveness of predictive models and informing decision-making processes. Among the plethora of metrics available, accuracy, precision, recall, F1-score, and the confusion matrix emerge as indispensable tools for quantifying model performance and identifying areas for refinement and optimization.

Accuracy, the most straightforward metric, quantifies the overall correctness of predictions by measuring the proportion of correctly classified instances over the total number of instances. While accuracy provides a holistic view of model performance, it may not adequately capture the nuances inherent in imbalanced datasets, where one class significantly outweighs the other. Precision and recall offer complementary perspectives, delineating the trade-off between false positives and false negatives. Precision quantifies the proportion of correctly predicted disease cases among all instances predicted as positive, thereby illuminating the model's ability to avoid misclassifying healthy individuals as diseased. Conversely, recall calculates the proportion of correctly predicted disease cases among all actual positive cases, shedding light on the model's capacity to capture all instances of disease within the dataset.

The harmonic mean of precision and recall yields the F1-score, a composite metric that balances both metrics to provide a comprehensive assessment of model performance. By harmonizing precision and recall, the F1-score offers a nuanced perspective on model efficacy, particularly in scenarios where achieving a balance between false positives and false negatives is paramount. However, while these metrics furnish valuable insights into model performance, they often fall short in elucidating the nature and distribution of prediction errors.

Enter the confusion matrix, a fundamental tool in the arsenal of machine learning practitioners for dissecting model performance and diagnosing sources of error. Comprising a tabular representation of predicted versus actual outcomes, the confusion matrix categorizes instances into four distinct categories: true positives, true negatives, false positives, and false negatives. True positives denote instances correctly classified as positive by the model, true negatives represent instances correctly classified as negative, false positives signify instances erroneously classified as positive, and false negatives denote instances erroneously classified as negative.

By delineating prediction outcomes into these discrete categories, the confusion matrix offers a granular perspective on model performance, facilitating a deeper understanding of the types and frequencies of prediction errors. For instance, a high prevalence of false positives may indicate a propensity for the model to erroneously label healthy fish as diseased, while a preponderance of false negatives may signify a failure to capture instances

disease within the dataset. Such insights gleaned from the confusion matrix serve as invaluable guideposts for fine-tuning model parameters, refining feature selection, and optimizing prediction thresholds to enhance model accuracy and efficacy.

Moreover, the confusion matrix enables stakeholders to tailor intervention strategies and allocate resources judiciously based on the nature and distribution of prediction errors. By prioritizing areas of improvement identified through the confusion matrix analysis, aquaculture managers can institute targeted measures to mitigate the risk of disease outbreaks, thereby safeguarding the health and welfare of farmed fish while bolstering the resilience of aquaculture operations.

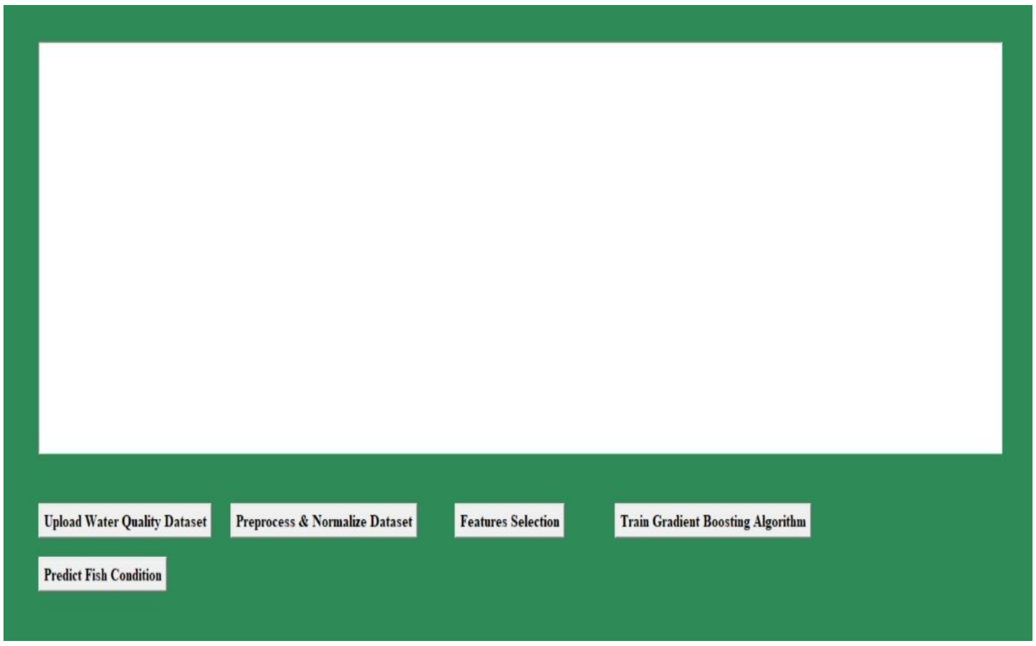
In tandem with other evaluation metrics, such as accuracy, precision, recall, and the F1-score, the confusion matrix forms an integral component of the model evaluation toolkit, empowering stakeholders with actionable insights to drive informed decision-making and optimize aquaculture management practices. As the aquaculture industry continues to embrace technological innovations and data-driven approaches, the judicious utilization of evaluation metrics, anchored by the elucidative power of the confusion matrix, holds the key to unlocking new frontiers in disease detection resource management, and sustainability. By harnessing the synergistic potential of machine learning and domain expertise, stakeholders can navigate the complex waters of aquaculture management with confidence and foresight, charting a course towards a future defined by resilience, productivity, and ecological integrity.

5.2 TEST CASES

Sl.no	Test Case	Result
1	Validate system handles normal water quality data	Pass
2	Detect coliform bacteria at critical pH level	Pass
3	Detect E. coli at high temperature	Pass
4	Validate system handles normal water quality data	Pass
5	Detect disease at critical pH level	Pass
6	Detect disease at high temperature	Pass
7	Detect disease at low dissolved oxygen	Pass
8	Detect disease at high ammonia level	Pass

5.3 RESULTS

- This approach improves fish health monitoring, reduces mortality rates, and enhances overall aquaculture productivity.
- Performance is assessed using metrics like accuracy, precision, recall, F1-score.
- The accuracy of our project is around 92%.


Upload Water Quality Dataset
Preprocess & Normalize Dataset
Features Selection
Train Gradient Boosting Algorithm
Predict Fish Condition

5.3 Output Screen (1)

This image shows a simple user interface with five buttons at the bottom. Each button has a specific function:

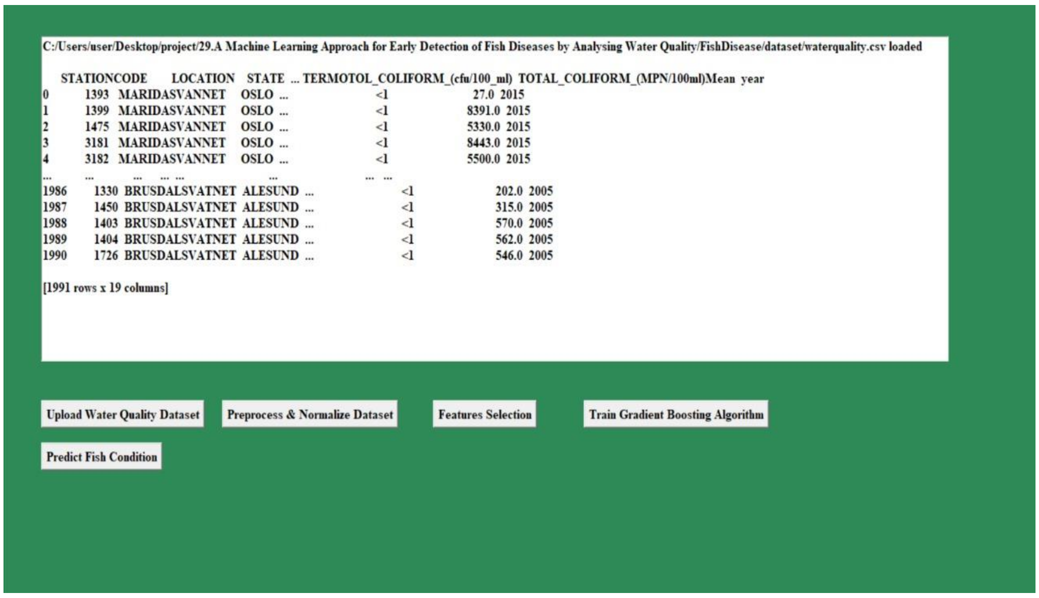
Upload Water Quality Dataset: Allows the user to upload a dataset related to water quality.

Preprocess & Normalize Dataset: Prepares and normalizes the uploaded dataset for analysis.

Features Selection: Selects important features from the dataset for further processing.

Train Gradient Boosting Algorithm: Trains a machine learning model using the Gradient Boosting algorithm.

Predict Fish Condition: Uses the trained model to predict the condition of fish based on the water quality data.



```
C:/Users/user/Desktop/project/29.A Machine Learning Approach for Early Detection of Fish Diseases by Analysing Water Quality/FishDisease/dataset/waterquality.csv loaded

    STATIONCODE      LOCATION   STATE ... TERMOTOL_COLIFORM_(cfu/100_ml) TOTAL_COLIFORM_(MPN/100ml)Mean  year
0        1393  MARIDASVANNET    OSLO ...                    <1                   27.0  2015
1        1399  MARIDASVANNET    OSLO ...                    <1                 8391.0  2015
2        1475  MARIDASVANNET    OSLO ...                    <1                 5330.0  2015
3        3181  MARIDASVANNET    OSLO ...                    <1                 8443.0  2015
4        3182  MARIDASVANNET    OSLO ...                    <1                 5500.0  2015
...       ...            ...     ... ...                   ...                    ...   ...
1986     1330  BRUSDALSVATNET  ALESUND ...                  <1                  202.0  2005
1987     1450  BRUSDALSVATNET  ALESUND ...                  <1                  315.0  2005
1988     1403  BRUSDALSVATNET  ALESUND ...                  <1                  570.0  2005
1989     1404  BRUSDALSVATNET  ALESUND ...                  <1                  562.0  2005
1990     1726  BRUSDALSVATNET  ALESUND ...                  <1                  546.0  2005

[1991 rows x 19 columns]
```

Upload Water Quality Dataset
Preprocess & Normalize Dataset
Features Selection
Train Gradient Boosting Algorithm
Predict Fish Condition

5.4 Output Screen (2)

The buttons allow users to upload a dataset, preprocess and normalize the data, select important features, train a machine learning model using the Gradient Boosting algorithm, and predict fish condition based on the water quality data. The interface appears user-friendly and intuitive for conducting analysis on water quality datasets.

CHAPTER-6
CONCLUSION AND FUTURE ENHANCEMENT

- Our project demonstrates that the application of machine learning (ML) for fish disease detection through water quality assessment holds transformative potential for aquaculture management.
- By processing large datasets of water quality parameters such as pH, temperature, dissolved oxygen, and ammonia levels, our models accurately predict the onset of diseases in fish populations.
- This innovation not only enhances the efficiency and accuracy of disease detection but also allows for real-time monitoring, enabling prompt interventions and minimizing economic losses.
- This technology is particularly beneficial for farmers, contributing to healthier aquaculture environments, improved fish welfare, and increased productivity.
- Future enhancements include integrating IoT devices for continuous real-time data collection, developing user-friendly mobile and web applications for easy access to monitoring systems and predictions.
- Furthermore, automated intervention systems that adjust water quality parameters based on machine learning predictions, along with alert systems for abnormalities, ensure immediate action can be taken by farmers.
- Ultimately, our project lays the groundwork for sophisticated, reliable, and user-friendly solutions, significantly benefiting the aquaculture industry and its farmers.

REFERENCES

1. GitHub

   https://github.com/kwea123/fish_detection.git

2. Journals

   https://www.researchgate.net/publication/349424327_A_Machine_Learning_Approach_for_Early_Detection_of_Fish_Diseases_by_Analyzing_Water_Quality

   https://www.mdpi.com/2073-4441/14/18/2836

3. Tkinter

   Python Tkinter - GeeksforGeeks

4. Gradient boosting

   https://www.javatpoint.com/gbm-in-machine-learning

APPENDIX
DEFINITIONS, ACRONYMS, ABBREVIATIONS

1. **Machine Learning**

   Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to learn from and make predictions or decisions based on data without being explicitly programmed. By using algorithms to identify patterns in data, ML models improve their performance over time through training on datasets. Key types of ML include supervised learning, unsupervised learning, semi-supervised learning. Applications of ML span various fields, including image and speech recognition, natural language processing, healthcare, autonomous driving, and recommendation systems.

2. **Gradient Boosting**

   Gradient boosting is a machine learning technique used for regression and classification tasks that builds models in a sequential manner. It combines the predictions of several weaker models, typically decision trees, to create a stronger model. Each new model is trained to correct the errors of the previous models by minimizing a specified loss function using gradient descent.

3. **Aquaculture**

   Aquaculture is the practice of breeding, rearing, and harvesting fish, shellfish, and other aquatic organisms in controlled environments, providing an alternative to wild fishing and contributing to food production.

4. **Fish Disease**

   Fish diseases are illnesses that affect fish, caused by pathogens such as bacteria, viruses, parasites, and fungi, or environmental factors like poor water quality. These diseases can lead to significant health issues and mortality in both wild and farmed fish populations.

5. Water Quality Assessment

Water quality assessment involves evaluating the physical, chemical, and biological characteristics of water to determine its suitability for specific uses, recreation, or habitat for aquatic life.

SAMPLE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
import numpy as np
from tkinter.filedialog import askopenfilename
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import normalize

from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import seaborn as sns

main = tkinter.Tk()
main.title("PREDICTING  FISH DISEASES EARLY USING MACHINE
LEARING AND WATER QUALITY ASSESSMENT") #designing main
screen
main.geometry("1300x1200")

global filename
global X_train, X_test, y_train, y_test
global X, Y
global classifier
global dataset
global le1, le2, le3, le4, le5

def upload(): #function to upload tweeter profile
    global filename
    global dataset
    filename = filedialog.askopenfilename(initialdir="dataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n");
    dataset = pd.read_csv(filename,encoding='iso-8859-1')
    dataset.fillna(0, inplace = True)
    text.insert(END,str(dataset))

def getAdaptiveFFT(data): #function to calculate FFT on recordings
    return np.fft.fft(data)/len(data)

def preprocess():
    text.delete('1.0', END)
    global X
    global Y
    global dataset
    global before_features
    global le1, le2, le3, le4, le5
    le1 = LabelEncoder()
    le2 = LabelEncoder()
    le3 = LabelEncoder()
    le4 = LabelEncoder()
    le5 = LabelEncoder()
    Y = []
    #taking dataset as input which contains M cities, N indicators and T
recordings
```

```
 dataset['LOCATION']                                                    =
pd.Series(le1.fit_transform(dataset['LOCATION'].astype(str)))
                                          dataset['STATE']              =
pd.Series(le2.fit_transform(dataset['STATE'].astype(str)))
                                         dataset['MONTH']               =
pd.Series(le3.fit_transform(dataset['MONTH'].astype(str)))
                                 dataset['CIPerf_(cfu/100_ml)']         =
pd.Series(le4.fit_transform(dataset['CIPerf_(cfu/100_ml)'].astype(str)))
                  dataset['TERMOTOL_COLIFORM_(cfu/100_ml)']          =
pd.Series(le5.fit_transform(dataset['TERMOTOL_COLIFORM_(cfu/100_
ml)'].astype(str)))
   dataset = normalize(dataset.values) #dataset normalization
    for i in range(len(dataset)): #looping each indicatior from dataset to
calculate water quality
      fft = getAdaptiveFFT(dataset[i]) #calculating FFT on indicators
      signal = np.amax(fft) #getting max signal from FFT
      signal = str(signal)
      signal = signal[1:4]
      signal = float(signal)
      T = dataset[i,10] #getting indicator recording from dataset
      if signal < T/2: #if signal < indicator/2 then calculated value will be 1
else0
         Y.append(1)
      else:
         Y.append(0)
   Y = np.asarray(Y)
   X = dataset #dataset normalization
   text.insert(END,"Dataset Preprocessing Completed\n\n")
   text.insert(END,str(X))

def featureSelection():
   global X_train, X_test, y_train, y_test
```

```
  text.delete('1.0', END)
    global X, Y
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    text.insert(END,"Total records found in dataset :
"+str(X.shape[0])+"\n\n")
    text.insert(END,"Dataset train and test split details\n")
    text.insert(END,"80% dataset records used for training :
"+str(X_train.shape[0])+"\n")
    text.insert(END,"20% dataset records used for testing  :
"+str(X_test.shape[0])+"\n")
    coliform = X[:,6]
    ecoli = X[:,7]
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Sampling Data')
    plt.ylabel('Indicator Values')
    plt.plot(coliform, 'ro-', color = 'blue')
    plt.plot(ecoli, 'ro-', color = 'orange')
    plt.legend(['Coliform', 'Ecoli'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('Coliform & Ecoli Virus Found in Dataset')
    plt.show()

def runGradientBoosting():
    text.delete('1.0', END)
    global classifier
    global X_train, X_test, y_train, y_test
    rfc = GradientBoostingClassifier()
    rfc.fit(X_train, y_train)
    predict = rfc.predict(X_test)
    classifier = rfc
    precision = precision_score(y_test, predict,average='macro') * 100
    recall = recall_score(y_test, predict,average='macro') * 100
    fscore = f1_score(y_test, predict,average='macro') * 10
```

```
 accuracy = accuracy_score(y_test,predict)*100

 text.insert(END,"Gradient Boosting Accuracy  : "+str(accuracy)+"\n")
    text.insert(END,"Gradient Boosting Precision : "+str(precision)+"\n")
    text.insert(END,"Gradient Boosting Recall    : "+str(recall)+"\n")
    text.insert(END,"Gradient Boosting FSCORE    : "+str(fscore)+"\n\n")
    labels = ['Healthy Fish', 'Diseases Detected']
    conf_matrix = confusion_matrix(y_test, predict)
    ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels,
annot = True, cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,len(labels)])
    plt.title("Gradient Boosting Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()


def predict():
    global pca
    global le1, le2, le3, le4, le5
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="dataset")
    test = pd.read_csv(filename,encoding='iso-8859-1')
    test.fillna(0, inplace = True)
    temp = test.values
    test['LOCATION'] =
pd.Series(le1.transform(test['LOCATION'].astype(str)))
    test['STATE'] = pd.Series(le2.transform(test['STATE'].astype(str)))
    test['MONTH'] = pd.Series(le3.transform(test['MONTH'].astype(str)))
    test['CIPerf_(cfu/100_ml)'] =
pd.Series(le4.transform(test['CIPerf_(cfu/100_ml)'].astype(str)))
```

```
test['TERMOTOL_COLIFORM_(cfu/100_ml)'] =
pd.Series(le5.transform(test['TERMOTOL_COLIFORM_(cfu/100_ml)'].as
type(str)))
    test = test.values
    #test = normalize(test)
    y_pred = classifier.predict(test)
    print(y_pred)
    for i in range(len(test)):
        predict = y_pred[i]  #np.argmax(y_pred[i])
        if predict == 0:
            text.insert(END,"X=%s, Predicted = %s" % (temp[i], '=====>
Healthy Fish')+"\n\n")
        else:
            text.insert(END,"X=%s, Predicted = %s" % (temp[i], '=====> Fish
will get affected by Coliform & Ecoli Virus')+"\n\n")

font = ('times', 16, 'bold')
title = Label(main, text='PREDICTING  FISH DISEASES EARLY
USING MACHINE LEARING AND WATER QUALITY
ASSESSMENT')
title.config(bg='darkviolet', fg='gold')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)


font1 = ('times', 12, 'bold')
```

```
uploadButton = Button(main, text="Upload Water Quality Dataset",
command=upload)

uploadButton.place(x=50,y=550)
uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess & Normalize Dataset",
command=preprocess)
processButton.place(x=290,y=550)


processButton.config(font=font1)

featureButton = Button(main, text="Features Selection",
command=featureSelection)
featureButton.place(x=570,y=550)
featureButton.config(font=font1)

gbButton = Button(main, text="Train Gradient Boosting Algorithm",
command=runGradientBoosting)
gbButton.place(x=770,y=550)
gbButton.config(font=font1)

predictButton = Button(main, text="Predict Fish Condition",
command=predict)
predictButton.place(x=50,y=600)
predictButton.config(font=font1)

main.config(bg='sea green')
main.mainloop()
```