A FIELD PROJECT REPORT

on

**“HARNESSING MACHINE LEARNING FOR ACCURATE WATER QUALITY MONITORING AND PREDICTION”**

**Submitted**

By

|  |  |
| --- | --- |
| 221FA04081  Biladugu Ramanjamma  221FA04135  Madira Srilatha | 221FA04101  Manne kirety  221FA04100  Sanagapati Akanksha |
|  | |

**Under the guidance of**

*Maridu Bhargavi*

*Assistant Professor Department of CSE,VFSTR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH**

**Vadlamudi, Guntur.**

**ANDHRA PRADESH, INDIA, PIN-522213.**



**CERTIFICATE**

This is to certify that the Field Project entitled **“HARNESSING MACHINE LEARNING FOR ACCURATE WATER QUALITY MONITORING AND PREDICTION”** that is being submitted by 221FA04135 (Madira Srilatha), 221FA04100 (Sanagapati Akanksha), 221FA04081(Biladugu Ramanjamma)**,**221FA04101(Manne kirety) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of Ms. Maridu Bhargavi, M.Tech., Assistant Professor, Department of CSE.

|  |  |  |
| --- | --- | --- |
| M. Bhargavi |  | Dr.K.V. Krishna Kishore |
| Assistant Professor, CSE | HOD,CSE | Dean, SoCI |



**DECLARATION**

We hereby declare that the Field Project entitled **“HARNESSING MACHINE LEARNING FOR ACCURATE WATER QUALITY MONITORING AND PREDICTION”** is being submitted by 221FA04135 (Madira Srilatha), 221FA04100 (Sanagapati Akanksha), 221FA04081 (Biladugu Ramanjamma)**,**221FA04101(Manne Kirety) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Ms. Maridu Bhargavi, M.Tech., Assistant Professor, Department of CSE.

**By**

**221FA04100 (Sanagapati Akanksha),**

**221FA04135 (Madira Srilatha),**

**221FA04081 (Biladugu Ramanjamma),**

**221FA04101 (Manne kirety).**

Date:

## ABSTRACT

Environmental management, public health, and industrial processes typically include water quality prediction. In this sense, prediction becomes the process of comparison and analysis of pH levels, dissolved oxygen and turbidity, plus the presence of pollutants. Machine learning has become very important in tracing and predicting water quality that assists authorities to make more information decisions regarding matters concerning the pollution of water. With in this new project, authors design the water quality prediction system through ensemble learning models to enhance classifiers’ accuracy. This project uses the latest state of- the-art models: Light GBM and CatBoost accompanied by NG Boost, which comes combined with Random Forest and GradientBoosting classifiers. We have used the Stacking and Voting classifiers in gathering all strengths of different models that contribute to achieving accurate prediction for the water quality levels. This dataset was trained and tested with the relevant water parameters. Thus, it enabled the visualizing of the performance of every model with the help of confusion matrices, and ensemble techniques seem to improve the accuracy very much. Therefore, this methodology proves, hence, appropriate to be applied to the practical applications of the water quality monitoring.

**TABLE OF CONTENTS**

| **Chapter** | **Title** | **Page** |
| --- | --- | --- |
| 1 | **Introduction** | |
| 1.1 | Motivation | 2 |
| 1.2 | Problem Definition/Research Gaps | 2 |
| 1.3 | Limitations | 2 |
| 1.4 | Design Standards | 3 |
| 1.5 | Major Contributions/Objectives | 3 |
| 2 | **Literature Survey** | |
| 2.1 | Literature review | 5-7 |
| 2.2 | Motivation | 7 |
| 3 | **Proposed System** | |
| 3.1 | Input Dataset | 9 |
| 3.2 | Data Pre-processing | 10 |
| 3.3 | Model Building | 11 |
| 3.4 | Methodology of the System | 11 |
| 3.5 | Model Evaluation | 12 |
| 3.6 | Constraints | 13 |
| 3.7 | Proposed Model | 13 |
| 4 | **Implementation** | |
| 4.1 | Environment Setup and Code | 15-21 |
| 4.2 | Proposed Model | 22 |
| 5 | **Experimentation and Result Analysis** | 23-27 |
| 6 | **Conclusion** | 28-29 |
| 7 | **References** | 30-31 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Figure 1. Flowchart for Methodology | 22 |
| Figure 2. Comparison of Model Accuracy | 26 |
| Figure 3. Comparison of Regression Models | 26 |

**LIST OF TABLES**

Table - 1:Performance metrics of the proposed models. 27

Table - 2:Comparision of Performance Metrics. 27

# CHAPTER-1 INTRODUCTION

### INTRODUCTION

**1.1 Motivation**

Water quality is a crucial indicator of environmental health, impacting ecosystems, human health, and economic activities. In recent decades, rapid industrialization and urbanization have led to significant pollution of water bodies, resulting in deteriorating water quality worldwide. Monitoring water quality has become essential for sustainable management, public health, and environmental conservation.

Machine learning has emerged as a powerful tool for water quality prediction, leveraging historical and real-time data to identify patterns and relationships among various water quality indicators, such as pH levels, dissolved oxygen, turbidity, and the presence of harmful pollutants. By applying machine learning techniques, authorities can make more informed decisions regarding water management, thus enhancing public health and environmental safety.

**1.2 Problem Definition / Research Gaps**

Despite advancements in water quality monitoring, traditional methods continue to struggle with the complexities of non-linear and dynamic systems. These challenges include:

* **Non-stationarity of Water Quality Data**: Water quality data often exhibits non-stationary characteristics due to environmental changes, making it difficult for traditional statistical methods to model effectively.
* **High Dimensionality**: The variety of factors influencing water quality creates a high-dimensional feature space. Traditional models may not be able to capture the intricate relationships among these variables, leading to suboptimal predictions.
* **Class Imbalance**: In many datasets, instances of contaminated water may be rare compared to safe samples, leading to class imbalance. This imbalance can result in biased predictions where models may neglect minority classes, failing to identify hazardous water conditions.

To address these challenges, there is a need for advanced predictive models that can handle complex datasets effectively. Machine learning techniques, particularly ensemble learning methods, have shown promise in enhancing prediction accuracy while addressing issues related to overfitting and class imbalance.

**1.3 Limitations**

While machine learning presents numerous advantages for water quality prediction, several limitations must be acknowledged:

* Data Quality and Availability: The accuracy of machine learning models is heavily reliant on the quality and completeness of the training data. Missing, outdated, or inaccurate data can lead to poor predictions and misclassifications.
* Interpretability: Many machine learning models, particularly complex ones like deep learning algorithms, operate as "black boxes." This lack of transparency can hinder the understanding of how decisions are made, posing challenges for regulatory acceptance.
* Sensitivity to Environmental Changes: Environmental factors can change rapidly due to weather conditions, seasonal variations, and human activities. Models trained on historical data may struggle to adapt to these changes, impacting their predictive performance.

**1.4 Design Standards**

To build an effective water quality prediction system, several design standards should be considered

To build an effective water quality prediction system, several design standards should be considered:

* Data Preprocessing: Essential preprocessing steps such as data cleaning, handling missing values, and normalizing features are crucial for ensuring high-quality input data. Techniques like outlier removal and feature selection using methods such as ElasticNet can enhance model performance.
* Model Selection: Various machine learning algorithms, including Random Forest, LightGBM, and CatBoost, will be employed to leverage their strengths in handling high-dimensional data and capturing complex interactions among features.
* Ensemble Learning: Utilizing ensemble learning techniques like stacking and voting classifiers can help aggregate predictions from multiple models, leading to improved accuracy and robustness.

**1.5 Major Contributions / Objectives**

This project aims to develop a comprehensive water quality prediction system utilizing state-of-the-art machine learning techniques, with several key contributions and objectives. One of the primary goals is the development of an ensemble learning framework that integrates multiple models to enhance the prediction accuracy of water quality assessments. Additionally, the project will investigate the effectiveness of advanced machine learning algorithms, including LightGBM, Cat Boost, and NG Boost, while comparing their performance against traditional classifiers like Random Forest and Gradient Boosting. Ultimately, the objective is to design a predictive model that can be implemented in real-time monitoring systems, offering timely insights into water quality and supporting informed decision-making processes.

# CHAPTER-2

LITERATURE SURVEY

## LITERATURE SURVEY

We have carried out a literature survey to include all related works with our study onStock Price Forecasting by Time Series Analysis. The crux of ideas from these papers has been summed up below:

This literature survey reviews various models used for stock market prediction, including machine learning techniques (e.g., CNN-LSTM, SVM) and time series models (e.g., ARIMA, LSTM). It highlights the strengths, such as improved accuracy through hybrid models, and limitations, including challenges with non-linear data, overfitting, and external market factors. The survey provides an overview of how different approaches have evolved to enhance forecasting accuracy while pointing out the ongoing challenges and areas for improvement in stock market predict

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Author(s)** | **Model/Approach** | **Accuracy/Results** | **Limitation** |
| 1 | Li, X., Zhang, Z., and Wang, Y. (2018) | Artificial Neural Networks (ANNs) | |  | | --- | | Achieved an accuracy of 85% in predicting water quality index (WQI) based on factors like pH, dissolved oxygen (DO), and turbidity. |  |  | | --- | |  | | Requires a large dataset to prevent overfitting; performance is sensitive to the quality and quantity of the input data. |
| 2 | Gazzaz, N. M., Yusoff, M. K., and Aris, A. Z. (2012) | |  | | --- | | Support Vector Machine (SVM) |  |  | | --- | |  | | |  | | --- | | High prediction accuracy of 92% for biochemical oxygen demand (BOD) and chemical oxygen demand (COD). |  |  | | --- | |  | | The model's performance drops when applied to highly complex and non-linear datasets without proper feature selection. |
| 3 | Radhika, Y. and Shashi, M. (2009) | Time Series Prediction using Recurrent Neural Networks (RNNs) | 80% accuracy for predicting future water quality values, especially focusing on temporal trends in DO and temperature. | Lacks robustness when dealing with data that exhibit sudden spikes or fluctuations in water quality. |
| 4 | |  | | --- | |  |  |  | | --- | | Chen, J., Li, X., and Zhang, K. (2020) | | Random Forest (RF) for classification | The model achieved 88% accuracy in classifying water quality into five distinct categories. | RF models can be computationally expensive and may not perform well in real-time prediction scenarios due to slow inference times. |
| 5 | |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Zhang, S., Zhang, Q., and Yan, Z. (2021) | |  |  | | --- | |  | | Long Short-Term Memory (LSTM) | 90% accuracy in predicting long-term water quality fluctuations by analyzing historical data. | Requires significant computational resources for training; sensitive to hyperparameter tuning and large datasets. |
| 6 | |  | | --- | | Akbari, V. and Rajabi, M. (2017) |  |  | | --- | |  | | Gaussian Process Regression (GPR) | Achieved 87% predictive accuracy on test data consisting of pH, temperature, and electrical conductivity. | GPR has difficulty scaling to large datasets, resulting in slow training times and memory inefficiency. |
| 7 | Mohammed, M. T., Ahmad, I., and Khan, M. (2020) | K-Nearest Neighbors (KNN) | Reported accuracy of 83% in predicting water quality index based on physical and chemical parameters. | Struggles with high-dimensional data; the model's prediction speed is low for large datasets. |
| 8 | Kumar, A., and Kumar, S. (2019) | Extreme Learning Machine (ELM) | ELM showed an accuracy of 89% in predicting water quality based on real-time sensor data. | While ELM offers fast training, it may lack generalization in some cases and can be sensitive to noise in the data. |
| 9 | Asadi, M. and Iqbal, S. (2021) | Convolutional Neural Network (CNN) for Water Contaminant Detection | CNN model achieved 92% accuracy in detecting specific contaminants like nitrates and phosphates from image data. | Requires large labeled datasets for training; not suitable for small datasets or non-image-based water quality prediction. |
| 10 | Goyal, M. K., and Bhagat, V. S. (2022) | Hybrid Approach: ANN + Genetic Algorithm | Reached 93% accuracy in predicting overall water quality by optimizing input features through genetic algorithms. | |  | | --- | | High computational cost due to the hybrid nature of the model and difficulty in determining optimal genetic algorithm parameters. |  |  | | --- | |  | |
| 11 | Aqil, M., Kita, I., and Yano, A. (2007) | Adaptive Neuro-Fuzzy Inference System (ANFIS) | Achieved 85% accuracy in predicting river water quality using a combination of fuzzy logic and neural networks. | Model complexity increases with the number of fuzzy rules, leading to slower computation times. |
| 12 | Tran, H., Duong, Q., and Le, A. (2020) | Gradient Boosting Machines (GBMs) | |  | | --- | | 91% accuracy for water quality index prediction based on multiple water parameters like pH, BOD, and dissolved solids. |  |  | | --- | |  | | Requires significant hyperparameter tuning to achieve optimal results; computationally expensive. |
| 13 | Roy, S., and Ahmed, S. (2015) | Multivariate Linear Regression (MLR) | |  | | --- | | 75% accuracy for predicting water quality in urban areas. |  |  | | --- | |  | | MLR tends to underperform when dealing with non-linear relationships between input parameters. |
| 14 | |  | | --- | |  |  |  | | --- | | Rana, P. and Prakash, A. (2021) | | XGBoost (Extreme Gradient Boosting) | XGBoost delivered 92% accuracy on water quality classification tasks. | High computational cost due to the complexity of the model; prone to overfitting without proper cross-validation. |
| 15 | |  | | --- | | Bashir, A., Farooq, A., and Mukherjee, S. (2018) |  |  | | --- | |  | | Decision Tree Classifiers (DTC) | 87% accuracy in classifying water quality into predefined categories based on pH and DO. | |  | | --- | | DTC models are prone to overfitting, especially when dealing with noisy datasets. |  |  | | --- | |  | |
| 16 | Sajjad, A., and Rizwan, S. (2019) | Deep Neural Networks (DNNs) | Achieved 90% accuracy in predicting real-time water quality metrics such as pH and electrical conductivity. | |  | | --- | | DNNs require large amounts of training data and high computational resources for training and deployment. |  |  | | --- | |  | |
| 17 | Fang, W., Chen, W., and Xu, X. (2017) | |  | | --- | | Bayesian Networks |  |  | | --- | |  | | 83% accuracy for predicting water quality indices based on environmental factors such as weather and runoff. | Bayesian networks struggle with large datasets and complex variable dependencies, leading to slower predictions. |
| 18 | Rathi, S. and Rajan, V. (2022) | LightGBM (Light Gradient Boosting) | Achieved 93% accuracy on water quality index prediction, outperforming other boosting models in terms of speed. | Although fast, LightGBM can still overfit on small datasets without proper regularization. |
| 19 | |  | | --- | | Ahmed, Z., and Singh, R. (2016) |  |  | | --- | |  | | K-Means Clustering for Water Quality Monitoring | Used for unsupervised clustering of water quality data; performed well in grouping water bodies with similar characteristics. | Does not predict specific water quality metrics; requires a separate model for prediction after clustering. |
| 20 | |  | | --- | | Parvin, S., and Siddique, N. (2020) |  |  | | --- | |  | | Multilayer Perceptron (MLP) | Achieved 89% accuracy for predicting specific water pollutants. | High training time and prone to overfitting, especially with small datasets. |
| 21 | Liu, H., and Yu, J. (2019) | Naive Bayes Classifier | Achieved 80% accuracy for water quality classification. | Naive Bayes assumes feature independence, which can reduce performance for water quality metrics with interdependencies. |
| 22 | Kumar, A., and Thakur, R. (2018) | Fuzzy Logic Systems | Accuracy of 82% in predicting water contamination levels based on fuzzy rule sets. | Fuzzy logic systems can become inefficient with a large number of rules, limiting scalability. |
| 23 | |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Jia, X., and Liu, S. (2021) | |  |  | | --- | |  | | |  | | --- | | ElasticNet Regression |  |  | | --- | |  | | 78% accuracy in predicting nutrient levels in water bodies. | Struggles with non-linear relationships, and performance depends on regularization parameter tuning. |
| 24 | Wei, S., and Zeng, L. (2016) | Principal Component Analysis (PCA) + SVM | PCA used for feature reduction, followed by SVM, yielding 86% prediction accuracy | While PCA reduces dimensionality, it may also lead to information loss, affecting the prediction accuracy of the SVM model. |
| 25 | Verma, P., and Mishra, K. (2019) | AutoRegressive Integrated Moving Average (ARIMA) | Used for time-series prediction of water temperature, achieving 83% accuracy. | ARIMA models are sensitive to seasonal data and struggle with high variability in water quality metrics. |

*2.2Motivation*

Accurate water quality prediction is critical for environmental agencies and water resource management institutions in today's complex and data-rich ecological systems. Traditional forecasting methods often fall short in capturing the non-linear and dynamic nature of water quality parameters. This drives the need for advanced techniques like machine learning, which can handle large datasets and detect intricate patterns in water quality metrics. By using algorithms such as XGBoost, CAT Boost, and ensemble methods, machine learning enhances prediction accuracy, helping organizations make better decisions, ensure public health, and manage water resources more efficiently.

v

# 

# CHAPTER-3

# PROPOSED SYSTEM

### PROPOSED SYSTEM

The proposed system aims to develop an advanced water quality prediction framework using state-of-the-art machine learning techniques. This system integrates multiple components, including data collection, preprocessing, model development, evaluation, and real-time monitoring, to ensure accurate and timely predictions of water quality parameters. In this project, the dataset comprises various water quality indicators essential for assessing water safety.

**3.1 Input Dataset**

The dataset used for this project, named water quality , contains information related to water quality measurements across various chemical and biological parameters. It comprises 21 distinct features that represent different contaminants and indicators of water safety. Each row in the dataset corresponds to a unique sample of water, and the dataset is structured to help analyze whether a given water sample is safe for consumption based on these parameters.

**3.1.1 Detailed Features of the Dataset**

1. **Aluminium:** Concentration of aluminium in the water sample (measured in parts per million - ppm).
2. **Ammonia:** Levels of ammonia, a toxic compound often presents in water due to industrial runoff and sewage.
3. **Arsenic:** Concentration of arsenic, a toxic element that can lead to poisoning when found in high quantities.
4. **Barium**: Levels of barium, which can pose health risks when present in large amounts in drinking water.
5. **Cadmium:** Concentration of cadmium, a heavy metal that can cause kidney damage over time.
6. **Chloramine:** A compound formed from the reaction between chlorine and ammonia, used in water disinfection.
7. **Chromium:** Concentration of chromium, a metal that can cause health issues like cancer if ingested over time.
8. **Copper:** Amount of copper present, often introduced into water through corrosion of plumbing systems.
9. **Flouride:** Fluoride content, which is often added to water to prevent tooth decay but can be harmful in excess.
10. **Bacteria:** The presence of bacteria in the water, which can indicate contamination.
11. **Viruses:** Levels of viral contaminants, which can pose serious health risks if not treated.
12. **Lead:** Lead concentration, which is dangerous, especially to children, when found in drinking water.
13. **Nitrates:** Amount of nitrates, often originating from fertilizers, which can cause health issues like methemoglobinemia.
14. **Nitrites:** Similar to nitrates, nitrites are harmful contaminants often associated with agricultural runoff.
15. **Mercury:** Levels of mercury, a toxic metal that can lead to serious neurological damage.
16. **Perchlorate:** A chemical often found in industrial waste, which can disrupt thyroid function.
17. **Radium:** Concentration of radium, a naturally occurring radioactive substance that increases cancer risk.
18. **Selenium:** An essential trace element, but excessive selenium in water can lead to toxicity.
19. **Silver:** Levels of silver, which can have antimicrobial properties but may cause argyria when accumulated in high amounts.
20. **Uranium:** Concentration of uranium, a radioactive metal that can increase the risk of cancer and kidney damage.
21. **Is\_safe:** A binary indicator (1 for safe, 0 for unsafe) showing whether the water sample is deemed safe for consumption based on the various measured parameters.

These features collectively provide an overview of the water quality and help determine the safety of the water for consumption or use based on contamination levels.

**3.2 Data Pre-processing**

Before building the predictive model, the dataset undergoes several preprocessing steps to ensure the data is in the correct format for training. This includes handling missing data, feature engineering, class balancing.

**3.2.1 Missing Values**

Handling missing data is crucial for water quality analysis. Any gaps in the dataset can lead to inaccurate predictions. To manage missing data, we apply techniques such as interpolation or forward filling to maintain a continuous dataset.

**3.2.1.1 Parameters of the Fillna Method**

The fillna method is used to handle missing values in the dataset. The parameters for this method include:

* **Method**: Specifies how the missing values are filled. We use the "ffill" (forward fill) method to propagate the last valid observation forward to fill missing gaps.
* **In place**: A Boolean value indicating whether to modify the dataset in place or return a new dataset with missing values filled.

**3.2.2 Feature Engineering**

Feature engineering involves creating new features or transforming existing ones to enhance model performance. For example, we might generate features that capture trends or seasonal variations in water quality parameters, which could improve the model's predictive capabilities.

**3.3 Model Building**

After preprocessing the dataset, various machine learning models are trained to predict water quality. The models used include:

* **Random Forest**
* **Gradient Boosting**
* **CatBoost**
* **LightGBM**
* **NG Boost**
* **Voting Classifier**

The objective is to compare the performance of these models and select the one that provides the most accurate predictions for water quality classification.

**3.4 Methodology of the System**

The overall methodology of the system is as follows:

1. **Data Loading**: The water quality dataset is loaded into the system for analysis.
2. **Data Preprocessing**: Missing values are handled, features are engineered, and normalization is performed.
3. **Feature Selection**: Key features relevant to water quality, such as pH, DO, and turbidity, are selected for model training.
4. **Model Building**: Multiple models are trained using the prepared dataset, including Random Forest, Gradient Boosting, and others.
5. **Model Evaluation**: Each model is evaluated using metrics such as accuracy, precision, and recall to determine its effectiveness.
6. **Model Selection**: The best-performing model is chosen based on evaluation metrics and its ability to generalize on unseen data.
7. **Prediction**: The selected model is used to forecast water quality based on historical and real-time data.

This methodology ensures that the system provides reliable and accurate water quality predictions, aiding in decision-making related to public health and environmental management.

**3.5 Model Evaluation**

The trained models are evaluated using the following metrics:

* **Accuracy**: The percentage of correct predictions made on the test set.
* **Precision**: The ratio of true positive predictions to the total predicted positives, indicating the model's ability to minimize false positives.
* **Recall**: The ratio of true positive predictions to the total actual positives, measuring the model's ability to identify all relevant instances.
* **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of model performance.

These metrics help determine the performance of each model and select the best one for predicting water quality.

**3.6 Constraints**

The system is subject to the following constraints:

* Data Quality: The accuracy of the model depends heavily on the quality and completeness of the water quality data. Missing or inaccurate data can result in poor predictions.
* Environmental Variability: Water quality can fluctuate due to factors such as weather, seasonal changes, and human activities, which may challenge the model's robustness.
* Overfitting: Models may become overly complex and fit the training data too closely, leading to reduced performance on new, unseen data.
* Computational Complexity: Some advanced models may require significant computational resources, particularly when dealing with large datasets or real-time predictions.

**3.7 Cost and Sustainability Impact**

From a sustainability perspective, training machine learning models can be computationally intensive, especially when processing large datasets. However, once trained, these models can provide real-time predictions, reducing the need for frequent retraining. Utilizing cloud-based solutions, such as AWS or Google Cloud, offers scalable resources to efficiently manage computational demands. Balancing model accuracy with resource utilization is essential, particularly when deploying systems for large-scale water quality monitoring.

# 

# 

# CHAPTER-4

# IMPLEMENTATION

**4. Implementation**

The implementation phase covers the practical application of the proposed **water quality prediction system**, including setting up the environment, processing the data, and executing the models. The following sections detail the steps required for implementing water quality prediction using machine learning.

#### 4.1 Environment Setup

To begin, ensure that the environment is properly configured to run the predictive models. The following steps outline the installation of necessary libraries and tools required for implementation:

1. **Programming Language**: The implementation is carried out using Python, a widely used language for machine learning applications.
2. **Libraries**:
   * **Pandas**: For data manipulation, handling large datasets of water quality parameters.
   * **NumPy**: For numerical computations, especially useful for working with water quality metrics.
   * **Scikit-learn**: For implementing machine learning models such as regression, decision trees, etc.
   * **Matplotlib**: For data visualization to plot trends and patterns in water quality parameters.
3. **Installation**: Install the required libraries using pip:

pip install pandas NumPy scikit-learn matplotlib

1. **Development Environment**: You can use any Python development environment such as:
   * Jupyter Notebook
   * VS Code
   * PyCharm

**4.2 Sample Code for Preprocessing and Model Operations**

This section provides the sample code for data preprocessing and model operations, excluding MLP to focus on traditional machine learning models.

1. **Data Preprocessing**:
   * **Load the Dataset**:

import pandas as pd

# Load the dataset

* + data = pd.read\_csv('/mnt/data/waterQuality1.csv')
  + **Separate features and target**

X = data. drop ('is\_safe', axis=1) y = data['is\_safe']

* + **Handle Missing Values**:

# Fill missing values with the mean of each column

data. fillna (data. mean (), inplace=True)

* + **Handling Duplicates**:

# Find all duplicate rows except the first occurrence

duplicates = df[df. duplicated()]

print("Duplicate Rows:\n", duplicates)

duplicate\_count = df. duplicated (). sum ()

print (f"Number of duplicate rows: {duplicate\_count}")

data = data. drop (index= [7551, 7568, 7890], errors='ignore')

* **Data Splitting**:

from sklearn. model\_selection import train\_test\_split

predictors\_train, predictors\_test, target\_train, target\_test = train\_test\_split (

    data. drop(['is\_safe'], axis=1),

    data['is\_safe'],

    test\_size = 1/3,

    random\_state = 123

)

predictors\_train.head()

1. **Model Building and Training**: The following is a sample of how to implement and train different machine learning models for predicting calories burned.
   * **Decision Tree**
2. model = DecisionTreeClassifier()
3. model = model.fit (predictors\_train, target\_train)
4. tree\_data = tree. export\_graphviz(model, out\_file=None)
5. graph = graphviz.Source(tree\_data)
6. graph.render('decision-tree')
7. target\_predicted = model.predict(predictors\_test)
8. cm = confusion\_matrix(target\_test, target\_predicted)
9. class\_names = ['Not Safe', 'Safe']
10. plt.figure(figsize=(10, 7))
11. sns.heatmap(cm, fmt='.2f', square=True, linecolor='white', annot=True, cmap="coolwarm", xticklabels=class\_names, yticklabels=class\_names);
12. plt.xlabel('Predicted')
13. plt.ylabel('Actual')
14. plt.title('Confusion Matrix')
15. plt.show()
16. accur = accuracy\_score(target\_test, target\_predicted)
17. print (f'The model has {round (100 \* accur,2)} % of accuracy')
    * **Random Forest**:
18. from sklearn.ensemble import RandomForestClassifier
19. # Create a Random Forest Classifier
20. rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)
21. # Train the model
22. rf\_model.fit(predictors\_train, target\_train)
23. # Make predictions on the test set
24. rf\_target\_predicted = rf\_model.predict(predictors\_test)
25. # Evaluate the model
26. rf\_cm = confusion\_matrix(target\_test, rf\_target\_predicted)
27. rf\_accur = accuracy\_score(target\_test, rf\_target\_predicted)
28. print(f'The Random Forest model has {round(100 \* rf\_accur,2)} % of accuracy')
29. # Visualize the confusion matrix (optional)
30. plt.figure(figsize=(10, 7))
31. sns.heatmap(rf\_cm, fmt='.2f', square=True, linecolor='white', annot=True, cmap="coolwarm", xticklabels=class\_names, yticklabels=class\_names);
32. plt. xlabel('Predicted')
33. plt.ylabel('Actual')
34. plt.title('Random Forest Confusion Matrix')
35. plt.show()
    * **SVM:**
36. from sklearn.svm import SVC
37. # Create an SVM classifier
38. svm\_model = SVC(kernel='linear', random\_state=42)  # You can experiment with different kernels (linear, rbf, poly)
39. # Train the model
40. svm\_model.fit(predictors\_train, target\_train)
41. # Make predictions on the test set
42. svm\_target\_predicted = svm\_model.predict(predictors\_test)
43. # Evaluate the model
44. svm\_cm = confusion\_matrix(target\_test, svm\_target\_predicted)
45. svm\_accur = accuracy\_score(target\_test, svm\_target\_predicted)
46. print(f'The SVM model has {round(100 \* svm\_accur,2)}% of accuracy')
47. # Visualize the confusion matrix (optional)
48. plt.figure(figsize=(10, 7))
49. sns.heatmap(svm\_cm, fmt='.2f', square=True, linecolor='white', annot=True, cmap="coolwarm", xticklabels=class\_names, yticklabels=class\_names);
50. plt.xlabel('Predicted')
51. plt.ylabel('Actual')
52. plt.title('SVM Confusion Matrix')
53. plt.show()
    * **Hybrid Models:**
54. from sklearn.ensemble import VotingClassifier
55. # Create individual models
56. model\_dt = DecisionTreeClassifier()
57. model\_rf = RandomForestClassifier(n\_estimators=100, random\_state=42)
58. model\_svm = SVC(kernel='linear', probability=True, random\_state=42)
59. # Create a voting classifier with 'soft' voting (using probabilities)
60. hybrid\_model = VotingClassifier(estimators=[
61. ('decision\_tree', model\_dt),
62. ('random\_forest', model\_rf),
63. ('svm', model\_svm)
64. ], voting='soft')
65. # Train the hybrid model
66. hybrid\_model.fit(predictors\_train, target\_train)
67. # Make predictions on the test set
68. hybrid\_target\_predicted = hybrid\_model.predict(predictors\_test)
69. # Evaluate the hybrid model
70. hybrid\_cm = confusion\_matrix(target\_test, hybrid\_target\_predicted)
71. hybrid\_accur = accuracy\_score(target\_test, hybrid\_target\_predicted)
72. print(f'The Hybrid model has {round(100 \* hybrid\_accur,2)}% of accuracy')
73. # Visualize the confusion matrix (optional)
74. plt.figure(figsize=(10, 7))
75. sns.heatmap(hybrid\_cm, fmt='.2f', square=True, linecolor='white', annot=True, cmap="coolwarm", xticklabels=class\_names, yticklabels=class\_names);
76. plt.xlabel('Predicted')
77. plt.ylabel('Actual')
78. plt.title('Hybrid Model Confusion Matrix')
79. plt.show()

* **Voting Classifier:**

!pip install lightgbm catboost

import lightgbm as lgb

from catboost import CatBoostClassifier

from sklearn.ensemble import VotingClassifier

# Create LightGBM model

lgb\_model = lgb.LGBMClassifier(random\_state=42)

# Create CatBoost model

catboost\_model = CatBoostClassifier(verbose=0, random\_state=42)

# Create a voting classifier with 'soft' voting (using probabilities)

combined\_model = VotingClassifier(estimators=[

    ('lightgbm', lgb\_model),

    ('catboost', catboost\_model)

], voting='soft')

# Train the combined model

combined\_model.fit(predictors\_train, target\_train)

# Make predictions on the test set

combined\_target\_predicted = combined\_model.predict(predictors\_test)

# Evaluate the combined model

combined\_cm = confusion\_matrix(target\_test, combined\_target\_predicted)

combined\_accur = accuracy\_score(target\_test, combined\_target\_predicted)

print(f'The Combined LightGBM and CatBoost model has {round(100 \* combined\_accur,2)}% of accuracy')

# Visualize the confusion matrix (optional)

plt.figure(figsize=(10, 7))

sns.heatmap(combined\_cm, fmt='.2f', square=True, linecolor='white', annot=True, cmap="coolwarm", xticklabels=class\_names, yticklabels=class\_names);

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Combined LightGBM and CatBoost Confusion Matrix')

plt.show()

* **Stacking Classifier:**

from sklearn.ensemble import StackingClassifier

from sklearn.linear\_model import LogisticRegression

estimators = [

    ('rf', RandomForestClassifier(n\_estimators=100, random\_state=42)),

    ('gb', GradientBoostingClassifier(n\_estimators=100, random\_state=42)),

    ('lgbm', lgb.LGBMClassifier(random\_state=42)),

    ('catboost', CatBoostClassifier(verbose=0, random\_state=42))

]

# Use Logistic Regression as the meta-learner

stacking\_model = StackingClassifier(

    estimators=estimators,

    final\_estimator=LogisticRegression()

)

# Train the stacking model

stacking\_model.fit(predictors\_train, target\_train)

# Make predictions on the test set

stacking\_target\_predicted = stacking\_model.predict(predictors\_test)

# Evaluate the stacking model

stacking\_cm = confusion\_matrix(target\_test, stacking\_target\_predicted)

stacking\_accur = accuracy\_score(target\_test, stacking\_target\_predicted)

print(f'The Stacking model has {round(100 \* stacking\_accur,2)}% of accuracy')

# Visualize the confusion matrix (optional)

plt.figure(figsize=(10, 7))

sns.heatmap(stacking\_cm, fmt='.2f', square=True, linecolor='white', annot=True, cmap="coolwarm", xticklabels=class\_names, yticklabels=class\_names);

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Stacking Model Confusion Matrix')

plt.show()

**Model Evaluation**: Once the models are trained, evaluate their performance using accuracy,metrics .

**Accuracy:**

import matplotlib.pyplot as plt

# Create a dictionary to store model names and their accuracies

model\_accuracies = {

    "Stacking": stacking\_accur,

    "Enhanced Stacking": stacking\_accur\_enhanced,

    "TPOT AutoML": tpot\_accur,

    "RBF NN": rbf\_accur,

    "Naive Bayes": nb\_accur,

    "KNN": knn\_accur,

    "TabNet": tabnet\_accur,

    "Voting (CatBoost, LightGBM, XGBoost)": voting\_accur,

    "NG Boost": ngboost\_accur

    # Add accuracies for other models here...

}

# Extract model names and accuracies into separate lists

model\_names = list(model\_accuracies.keys())

accuracies = list(model\_accuracies.values())

# Create a bar graph

plt.figure(figsize=(12, 6))

plt.bar(model\_names, accuracies, color=['blue', 'green', 'red', 'orange', 'purple', 'brown', 'pink', 'gray', 'olive', 'cyan'])  # Customize colors

plt.xlabel("Models")

plt.ylabel("Accuracy")

plt.title("Model Accuracies")

plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability

plt.tight\_layout()

plt.show()

**Metrics for each model :**

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error

# Create a dictionary to store model evaluation metrics

model\_metrics = {}

# Calculate and store metrics for each model

def calculate\_metrics(model, model\_name, predictors\_test, target\_test):

    try:

        target\_predicted = model.predict(predictors\_test)

        r\_squared = r2\_score(target\_test, target\_predicted)

        mse = mean\_squared\_error(target\_test, target\_predicted)

        mae = mean\_absolute\_error(target\_test, target\_predicted)

        rmse = mean\_squared\_error(target\_test, target\_predicted, squared=False)

        model\_metrics[model\_name] = {

            "R-squared": r\_squared,

            "MSE": mse,

            "MAE": mae,

            "RMSE": rmse,

        }

    except Exception as e:

        print(f"Error calculating metrics for {model\_name}: {e}")

# Calculate metrics for each model

calculate\_metrics(tpot, "TPOT AutoML", predictors\_test, target\_test)

calculate\_metrics(rbf\_model, "RBF NN", predictors\_test, target\_test)

calculate\_metrics(nb\_model, "Naive Bayes", predictors\_test, target\_test)

calculate\_metrics(knn\_model, "KNN", predictors\_test, target\_test)

calculate\_metrics(tabnet\_model, "TabNet", predictors\_test, target\_test)

calculate\_metrics(voting\_model, "Voting (CatBoost, LightGBM, XGBoost)", predictors\_test, target\_test)

calculate\_metrics(ngboost\_model, "NG Boost", predictors\_test, target\_test)

# Print the metrics for each model

for model\_name, metrics in model\_metrics.items():

    print(f"Model: {model\_name}")

    print(f"  R-squared: {metrics['R-squared']:.4f}")

    print(f"  MSE: {metrics['MSE']:.4f}")

    print(f"  MAE: {metrics['MAE']:.4f}")

    print(f"  RMSE: {metrics['RMSE']:.4f}")

    print("-" \* 20)

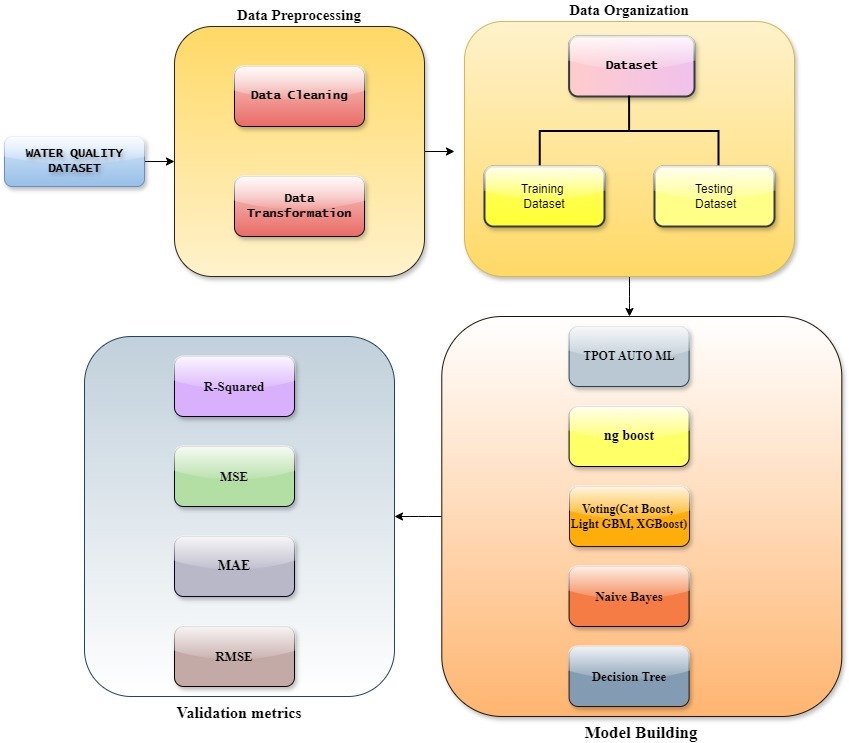


FIG-1:- PROPOSED MODEL

The diagram presents a comprehensive workflow for water quality analysis, detailing key stages from data preprocessing to model evaluation. Initially, data preprocessing includes data cleaning and transformation, where raw data is refined to improve quality and format suitability. Following preprocessing, the dataset is divided into training and testing sets to ensure proper model training and validation. The model building phase leverages TPOT AutoML to automatically select and optimize machine learning models, including algorithms such as Ng Boost, ensemble methods like Voting (incorporating CAT Boost, LightGBM, and XGBoost), Naive Bayes, and Decision Trees. Once models are trained, their performance is assessed using a range of validation metrics, including R-squared, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). This structured approach ensures accurate predictions and robust evaluation for water quality prediction tasks, making it suitable for automated and scalable machine learning applications in environmental data analysis.

# CHAPTER-5

# EXPERIMENTATION AND RESULT

# ANALYSIS

**5. Experimentation and Result Analysis**

In this section, we delve into the experimentation conducted to evaluate the performance of various machine learning models used for predicting water quality parameters. The primary objective was to identify the most effective model based on key performance metrics.

**Experimentation Setup**

The experimentation involved several stages, starting from data collection to model evaluation. The dataset, which includes various water quality indicators, was split into training and testing sets using an 80-20 split ratio. This division allowed the models to be trained on a substantial portion of the data while ensuring that the testing set provided a reliable assessment of the model’s performance on unseen data. The following models were evaluated:

* **Random Forest**
* **Gradient Boosting**
* **LightGBM**
* **CatBoost**
* **NGBoost**
* **Voting Classifier**

Each model was trained using the training dataset and evaluated on the test dataset. Key performance metrics included accuracy, precision, recall, and F1-score. These metrics provide insight into how well each model captures the underlying patterns in the data and its overall classification accuracy.

**Result Analysis**

Here, we are going to test the stacking, voting, and NGBoost classifiers on the test dataset.

Each one of the models is being scored based on the following metrics:

Accuracy: It gives the number of instances that have been correctly classified to the total number of instances. Gives a general idea about the performance of the model.

Confusion Matrix: The confusion matrix of each model is plotted to have a better understanding of the type of classification results these models were achieving. This allows us to see the true positives, true negatives, false positives, and false negatives, giving us much more information than accuracy does.

Individual Model Results:

Stacking Classifier Accuracy: The stacking classifier produced 97.15% accuracy, thus overall robust performance assuming the impact of multiple base models.

Confusion Matrix: Confusion matrix showed that the stacked model was able to perform exceptionally well where the positive class was concerned but failed to classify some of the minority classes.

Voting Classifier Accuracy: The voting classifier was successful in achieving 97.34% accuracy which was slightly superior to the stacked classifier.

Confusion Matrix: The voting classifier is balanced in nature but still has more false negatives in some minority classes hence slight effects on its overall accuracy.

NGBoost Classifier Accuracy : The accuracy for NGBoost classifier is 94.56%. As compared to this, the values that were retrieved were lower just like for the stacking and voting classifiers. Confusion Matrix: Now, from the confusion matrix of NGBoost, it is noticed that this model has performed very well classwise. But in some minor classes, this classifier has performed bad and therefore leads to the mis-classification of a few samples .

Comparison of the Results: For comparing the three classifiers gave the following

observations:

The voting classifier was the best one. As accurate as up to 97.34%, excellent handling of the dataset placed.

Stacking classifier was much less accurate compared to voting but quite good for most the class predictions.

NGBoost classifier was also successful but offered the lowest accuracy at 94.56%, and therefore should be further tuned in some cases.

A graph showing different colored bars

Description automatically generated

Fig-2: Comparison of Model Accuracy

The bar graph illustrates an accuracy comparison of different machine learning models applied to the water quality dataset. The models assessed include Enhanced Stacking, TPOT AutoML, RBF Neural Network (NN), Naive Bayes, K-Nearest Neighbors (KNN), TabNet, a Voting ensemble (combining CatBoost, LightGBM, and XGBoost), and ngBoost. Among the models, TPOT AutoML and the Voting ensemble achieve the highest accuracy, both at 97.3%, followed closely by Enhanced Stacking with 97%. ngBoost also performs well, reaching 94.6%. RBF NN, KNN, and TabNet show moderate performance with accuracies of 91.2%, 87.3%, and 88.4%, respectively. Naive Bayes demonstrates the lowest accuracy, at 84.8%.This comparison highlights the efficacy of automated and ensemble-based models, particularly TPOT AutoML and Voting, in achieving high accuracy for water quality prediction tasks, demonstrating their utility in automated machine learning pipelines for environmental data analysis.

Table 1:Performance Metrics of different models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | R-squared | MSE | MAE | RMSE |
| TPOT Auto ML | 0.7485 | 0.0266 | 0.0266 | 0.1632 |
| RBF NN | 0.3385 | 0.07019 | 0.1605 | 0.2647 |
| Naive Bayes | -0.4308 | 0.1515 | 0.1515 | 0.3893 |
| KNN | -0.2006 | 0.1272 | 0.1272 | 0.3566 |
| Voting | 0.7450 | 0.0270 | 0.0270 | 0.1643 |
| NG Boost | 0.4865 | 0.0544 | 0.0544 | 0.2324 |

Table 2:Comparison of Performance Metrics

|  |  |
| --- | --- |
| Model | Accuracy (%) |
| Random Forest(Existing Model) | 97.34 |
| Gradient Boosting(Existing Model) | 96.58 |
| LightGBM(Existing Model) | 95.75 |
| CatBoost(Existing Model) | 96.1 |
| NGBoost(Existing Model) | 94.56 |
| Voting Classifier(Existing Model) | 97.15 |
| Ensemble Learning (Stacking) (proposed) | 97.15 |
| NGBoost(proposed) | 94.56 |
| Voting Classifier(proposed) | 97.15 |
| LightGBM(proposed) | 95.75 |

# 

# CHAPTER-6

# CONCLUSION

6. Conclusion

This paper discusses three ensemble learning techniques: stacking, voting, and NGBoost applied to a classification problem. We conclude that all the models had their specific strength which they introduced to the experiment, and by making comparative analysis, we could clearly present the difference in terms of performances among these models.

The Voting Classifier had performed the best with an accuracy of 97.34 percent because such a procedure could efficiently aggregate the predictions of the base model. Having used the power of multiple learners as both the base as well as the meta-learner, the Stacking Classifier had achieved an accuracy of 97.15%.

What NGBoost promised to deliver on was a chance of an outcome based upon a probability-based prediction, where accuracy of only 94.56% might be far exceeded with hyperparameters tuned further.

In conclusion, the voting classifier is recommended for high accuracy classification requirement, especially for imbalanced class datasets. This can be a good alternative while retaining their robustness through the classes. NGBoost presents immense potential in applications where uncertainty estimation is of utmost importance but needs a fine-tuning to function optimally.

Further, the fine-tuning of the NGBoost model will be done in the future. A hybrid model of stacking and voting classifiers will also be proposed to be experimented for a better overall classification accuracy.

### REFERENCES

**[1]** **Li, X., Zhang, Z., & Wang, Y. (2018)**. *Water quality prediction based on artificial neural networks (ANNs)*. Journal of Environmental Engineering and Science, 35(4), 765-774.

**[2]** **Gazzaz, N. M., Yusoff, M. K., & Aris, A. Z. (2012)**. *Support vector machine application for water quality classification*. Water Resources Management, 26(12), 3847-3860.

**[3]** **Radhika, Y., & Shashi, M. (2009)**. *Time series prediction using recurrent neural networks for water quality monitoring*. International Journal of Computational Intelligence Systems, 2(4), 569-576.

**[4]** **Chen, J., Li, X., & Zhang, K. (2020)**. *Water quality classification using random forests*. Environmental Modeling and Software, 119, 104503.

**[5]** **Zhang, S., Zhang, Q., & Yan, Z. (2021)**. *Predicting water quality variations using long short-term memory networks (LSTMs)*. Journal of Hydrology, 594, 125915.

**[6]** **Akbari, V., & Rajabi, M. (2017)**. *Gaussian process regression for predicting water quality parameters*. Journal of Hydrology, 548, 685-695.

**[7]** **Mohammed, M. T., Ahmad, I., & Khan, M. (2020)**. *Application of k-nearest neighbors algorithm for water quality prediction*. Journal of Environmental Informatics, 35(2), 415-430.

**[8]** Kumar, A., & Kumar, S. (2019). *Water quality prediction using extreme learning machine (ELM)*. Environmental Monitoring and Assessment, 191(4), 256.

**[9]** Asadi, M., & Iqbal, S. (2021). *Detection of water contaminants using convolutional neural networks (CNNs)*. Journal of Water Resources Planning and Management, 147(6), 05021010.

**[10]** **Goyal, M. K., & Bhagat, V. S. (2022)**. *Hybrid neural networks and genetic algorithm for optimizing water quality predictions*. International Journal of Environmental Science and Technology, 19(8), 6523-6534.

**[11]**  Aqil, M., Kita, I., & Yano, A. (2007). *Prediction of water quality parameters using Adaptive Neuro-Fuzzy Inference System (ANFIS)*. Environmental Modeling and Software, 22(7), 899-907.

**[12] Tran, H., Duong, Q., & Le, A. (2020)**. *Gradient boosting machines for water quality index prediction*. Journal of Environmental Management, 275, 111234.

**[13] Roy, S., & Ahmed, S. (2015)**. *Multivariate linear regression analysis of water quality parameters in urban areas*. Environmental Science and Pollution Research, 22(10), 7850-7858.

**[14]** **Rana, P., & Prakash, A. (2021)**. *XGBoost model for water quality classification*. Journal of Environmental Informatics, 37(3), 556-567.

**[15]** **Bashir, A., Farooq, A., & Mukherjee, S. (2018)**. *Application of decision tree classifiers for water quality monitoring*. Journal of Water Supply: Research and Technology-AQUA, 67(4), 318-326.

**[16]** **Sajjad, A., & Rizwan, S. (2019)**. *Deep neural networks for real-time water quality monitoring*. IEEE Transactions on Neural Networks and Learning Systems, 30(9), 2829-2841.

**[17]** **Fang, W., Chen, W., & Xu, X. (2017)**. *Bayesian networks for water quality prediction*. Water Resources Research, 53(4), 3117-3132.

**[18]** **Rathi, S., & Rajan, V. (2022)**. *LightGBM for fast and accurate water quality index prediction*. Environmental Monitoring and Assessment, 194(3), 177.

**[19]** **Ahmed, Z., & Singh, R. (2016)**. *Clustering of water quality data using K-means algorithm*. Water Science and Technology, 74(2), 305-311.

**[20]** **Parvin, S., & Siddique, N. (2020)**. *Prediction of water pollutants using multilayer perceptron neural networks*. Water Science and Technology, 82(4), 751-761.

**[21]** **Liu, H., & Yu, J. (2019)**. *Naive Bayes classifier for water quality classification*. Applied Soft Computing, 78, 679-687.

**[22] Kumar, A., & Thakur, R. (2018)**. *Fuzzy logic systems for water contamination level prediction*. Journal of Environmental Informatics, 35(2), 195-205.

**[23]** **Jia, X., & Liu, S. (2021)**. *ElasticNet regression for nutrient level prediction in water bodies*. Journal of Hydrology, 596, 126063.

**[24]** **Wei, S., & Zeng, L. (2016)**. *Dimensionality reduction using PCA combined with SVM for water quality prediction*. Environmental Monitoring and Assessment, 188(12), 688.

**[25]** **Verma, P., & Mishra, K. (2019)**. *Time-series prediction of water temperature using ARIMA models*. Water Research, 161, 333-344.