Science and Technology Indonesia
e-ISSN:2580-4391 *p*-ISSN:2580-4405
Vol. 10, No. 4, October 2025

Research Paper

Decision Tree Algorithms in Water Quality Classification: A Comparative Study of Random Forest, XGBoost, and C5.0

Dewi Asiah Shofiana[1*], Melan Caniadi[1], Ridho Sholehurrohman[2], Aristoteles[1]

[1]*Department of Computer Science, Faculty of Mathematics and Natural Science, Universitas Lampung, Lampung, 35145, Indonesia*
[2]*Department of Mathematics, Faculty of Mathematics and Natural Science, Universitas Lampung, Lampung, 35145, Indonesia*
***Corresponding author**: dewi.asiah@fmipa.unila.ac.id

Abstract
Safe drinking water is more than a convenience; public health officials often call it a cornerstone of survival. United Nations International Children's Emergency Fund (UNICEF) reported that, shockingly, roughly two billion people still drink water that is neither clean nor tested. Pathogenic bacteria from human feces and livestock waste taint roughly 70% of available sources, creating a silent epidemic. Scientists express water quality into five levels: poor, marginal, fair, good, and excellent – named as the Water Quality Index (WQI) designed by the Canadian Council of Ministers of the Environment (CCME). This research measured the performance of three decision-tree classifiers, including Random Forest, XGBoost, and C5.0 to predict water quality. The preprocessing pipeline was thorough, involving label encoding, use of synthetic minor over-sampling technique (SMOTE) for balancing imbalanced classes, and an exploratory phase to examine outliers and irregularities within the dataset. According to the findings, Random Forest finished at an impressive test result with 98% of accuracy. XGBoost and C5.0 follows close behind at about 96%, but the latter turned out to be the fastest, edging out both XGBoost and Random Forest, making C5.0 a preferable when a time-sensitive or emergency decision is needed. In short, this research highlights the importance of modern preprocessing tools combined with machine learning algorithms in monitoring water quality.

Keywords
Water Quality, Decision Tree, Random Forest, Extreme Gradient Boosting, C5.0

Received: 30 January 2025, Accepted: 13 June 2025
https://doi.org/10.26554/sti.2025.10.4.999-1011

1. INTRODUCTION

Access to clean water and reliable sewage disposal ranks among the oldest yet most urgent of human rights, especially in densely populated settings (Filho et al., 2022). The sixth Sustainable Development Goal (SDGs) codifies that expectation by requiring that both services be available and managed in an environmentally sustainable manner (Pradhan et al., 2017). In practice, however, the availability of such services falls far short. United Nations Children's Fund (UNICEF) data lays bare the shortfall: roughly 2.2 billion people still drink unprocessed water, 4.2 billion rely on sanitation facilities that fail basic safety tests, and 3 billion households lack a simple place to wash their hands with soap. The sheer numbers spell out an urgent maintenance and upgrade task for pipes, latrines, and hygiene stations across continents (UNICEF, 2019). Experts warn that a 70 percent decline in water quality worldwide has been driven chiefly by fecal contamination, with pathogens such as *E. coli*, *Shigella sp.*, *Vibrio cholerae*, and *Salmonella* in the mix. Most observers agree that rising population density puts the heaviest strain on these fragile systems, with automatic knock-on effects for public health. Outbreaks and chronic illness are not accidental-they follow directly from this mismatch between growth and infrastructure (Holcomb and Stewart, 2020).

Over the past decade, researchers have increasingly turned to machine-learning to tackle environmental challenges such as water management-issues once thought impossible because of data scarcity and analytical limitations. A machine-learning system can be understood as a type of computer program that executes tasks autonomously after ingesting and processing large amounts of historical data via statistical models. Typically, the accuracy of its forecasts improves with continued access to new information. At the core of the process, data mining extracts relevant information from a multitude of databases, providing the critical insights for the next actions to be undertaken (Shen, 2018; Yuan et al., 2020).

Machine learning is appearing in environmental management at almost every stage of fieldwork today. Recent studies show the method being used for climate modelling (Eyring

et al., 2024), for real-time checks of water quality (Zhu et al., 2022), even for the early spotting of urban pollution (Xu et al., 2022). By integrating machine learning, organizations can improve their proactive responses to environmental challenges, leading to the development of sustainable solutions and data-driven decision-making practices. In practice the result usually firms up analytical workflows while steering policy toward more robust and sustainable outcomes (Rolnick et al., 2022; Pansara et al., 2024).

There has been a number of works in this sphere of research. In the year 2022, Nasir et al. (2022) managed to classify the water quality by employing multiple algorithms, where CATBoost performed the best with an accuracy of above 94.5%. Other studies more recently tried to model water quality parameters prediction through the use of KNN, Naive Bayes, CatBoost, ID3 and Random Forest. Unfortunately, their accuracies are still around 90% as well (Ilić et al., 2022; Yogeshwari et al., 2023; Mutoffar et al., 2022).

The current investigation analyzes water quality data using three decision tree algorithms, Random Forest Classifiers, Extreme Gradient Boosting (XGBoost), and C5.0. Unlike most previous studies, this research tries to incorporate several preprocessing techniques at the dataset's initial stage. Some preprocessing steps include the removal of some non-useful data attributes and creating a new attribute called Quality, which is a subset of the overall water quality index value. Besides, EDA is conducted with the help of descriptive statistics and visual tools, enabling the examination of data structure and patterns to measure maximum insights and identify outliers and anomalies. The study also applies SMOTE because of the class imbalance in the dataset. The differences in data splitting and classification have been achieved through two techniques; hold-out method and stratified k-fold cross-validation with the aim of determining the performance of the algorithms under each technique. This method ensures that sufficient analysis is conducted and enhances the reliability of the classification results.

2. EXPERIMENTAL SECTION

The research proceeds through a sequenced series of stages, each designed to scrutinize how decision-tree algorithms perform under various conditions. A preliminary literature survey offers a doctrinal grounding and identifies gaps that the present work intends to fill. Once the theoretical framework is in place, a dataset is gathered from publicly accessible repositories.

Data cleaning and feature engineering then take center stage, as missing values are imputed, outliers trimmed, and categorical variables are encoded in order to produce a tidy and coherent dataset. After preprocessing, the data are partitioned in two distinct ways: a simple hold-out split for baseline checks and a stratified k-fold arrangement that ensures balanced class representation in every fold. Building the models follows the splits; with three decision-tree variants including Random Forest, XGBoost, and C5.0 are fitted and tuned on the training subsets.

Model performance is gauged using the classical confusion-matrix and further distilled into singular numbers such as accuracy, precision, and recall. Side-by-side comparisons reveal where each variant excels or falters. Figure 1 diagrams the entire pipeline from acquisition through evaluation and visually reinforces the step-wise logic of this study.

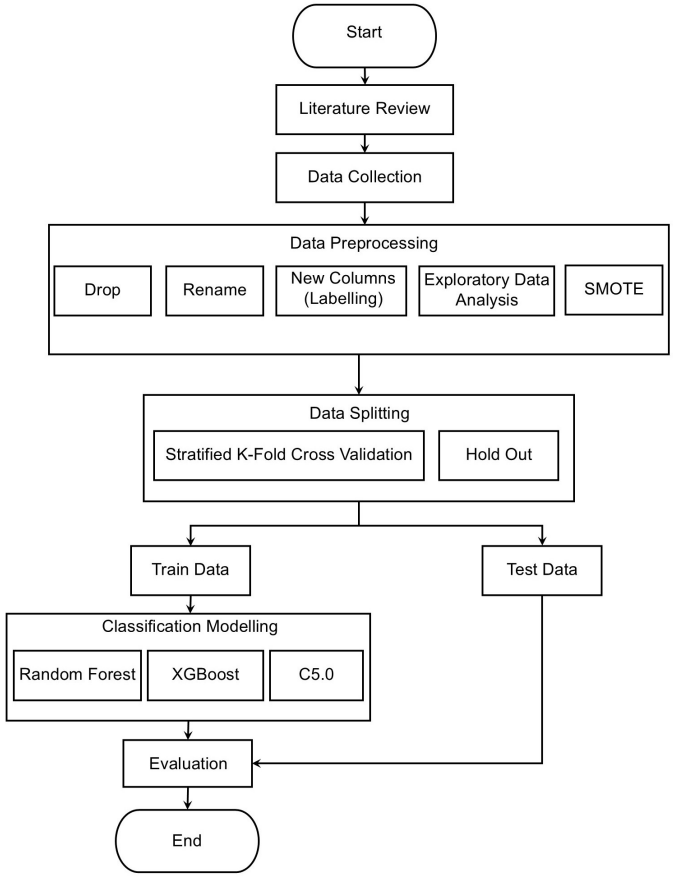

Figure 1. Research Workflow

2.1 Data Collection

This research utilized a dataset acquired from Kaggle (https://www.kaggle.com/datasets/hailla/wqi-parameter-scores-1994-2013) provided in CSV format, consisting of 13 attributes and a total of 971 instances. The data was collected by the Washington State Department of Ecology's River and Stream Monitoring Program, covering 62 rivers and streams in the United States from 1995 to 2014.

2.2 Data Preprocessing

Before diving into data mining, it is paramount that data undergoes precursory processing, where information is extracted and transformed into a comprehensible format suitable for analysis (Garcia et al., 2015). By addressing missing values and discrepancies, this process improves data quality and sets the stage for more precise and thorough data analysis (García et al., 2016).

2.2.1 Drop Unnecessary Attributes

Successful data mining tasks are fundamentally determined by careful attribute selection. Removing those that do not contribute improves quality by trimming away noise and clutter. This helps maximize model performance by reducing noise, prevention of overfitting and increasing accuracy. In comparison with attributes selection, these actions add structural sinew to each attribute so that the data sanity (quality, reliability, and consistency) is improved. During this phase, the following 6 attributes were dropped: station, station name, year, address, plus code, and location1. The remaining features are carried forward to the next phase, as Table 1 shows the information of each attribute.

2.2.2 Rename Attributes

Certain features have been adjusted to enhance clarity and eliminate any potential ambiguity. Renaming the columns improve their interpretability allowing them to be used directly in subsequent stages of the analysis without needing further explanation.

2.2.3 New Columns (Labelling)

In this study, the water quality index is based on the Canadian Council of Ministers of the Environment (CCME) Water Quality Index (WQI). The calculated WQI values were then classified into categories under the "Quality" attribute. The range of categories for CCME WQI is shown in Table 2 (Gikas et al., 2020).

2.2.4 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) provides a systematic approach for revealing hidden structures, correlations, and irregularities within a dataset. Anomalies, such as stray outliers, frequently emerge during this inspection and serve as early warning signals for possible data-quality troubles. EDA also probes the interdependence of variables and sketches the overall shape of the dataset long before any formal modelling takes place. Its toolkit is eclectic, combining summary statistics, histograms, scatter plots, and correlation matrices to visualize how values are distributed and how they vary across different columns of the table. Intuitive graphics serve a second purpose: they nudge researchers toward fresh questions that might merit further study (Majumder et al., 2022; Komorowski et al., 2016). Sample outputs from this phase are collected in Figures 2 and 3 for closer examination.

Data visualizations presented in Figure 2 permit a preliminary examination of regional water quality and reveal several pronounced fluctuations around the statistical mean. Dissolved oxygen concentrations cluster around a moderate baseline, implying the habitat remains broadly tenable for resident biota. pH readings drift toward the alkaline end of the scale, a tendency that hints at prospective chemical inputs capable of neutralizing natural acidity. Trace-element matrices for nitrogen and phosphorus show predominantly low values, yet isolated spikes suggest episodic contributions from runoff linked to row-crop fertilization or light-industrial discharge. The temperature distribution shows that many of the samples are warmer, which might reflect seasonal changes or thermal pollution.

The correlation matrix displayed in Figure 3 reveals several robust interrelationships among the water-quality variables. Most striking is the strong inverse association between the composite measure of water quality and key pollutants-fecal matter, sediment, nitrogen, phosphorus, and turbidity. As concentrations of these contaminants increase, the overall quality rating declines. Such declines are probably exacerbated by agricultural runoff and episodic soil erosion that release phosphorus and suspended particles simultaneously. A second, much weaker relationship appears between temperature and dissolved oxygen, implying that elevated temperatures may impair oxygen availability for aquatic organisms. Collectively, the data point to the necessity of vigilant monitoring and active management of these parameters to restore and sustain acceptable water quality.

2.2.5 Synthetic Minority Over-Sampling Technique (SMOTE)

Rather than creating simple copies of existing examples, SMO TE tackles the issue of class imbalance within datasets by formulating new synthetic examples for the minority class. This technique SMOTE employs increases the number of occurrences for the minority class, creating better balance among the different classes, leading to improvement of model precision, recall, and F1 measures for the minority class. Furthermore, SMOTE alleviates overfitting, improving the generalization of the model by biasing the classifier towards the minority class, resulting in better performance on new data (Pradipta et al., 2021).

2.3 Data Splitting

2.3.1 Hold Out

Splitting a dataset can be accomplished through the hold-out method, a practice that assigns separate portions for training and testing (Ghazvini et al., 2014). For the current investigation, 90% of the total observations (2115 samples) were reserved for model development, while the remaining 10% (235 samples) were kept apart for final testing.

2.3.2 Stratified K-Fold Cross Validation (SKCV)

SKCV is one of the techniques for model validation that requires partitioning data into training and testing sets. SKCV also has folds like k-fold cross-validation, but every single fold is class proportionate and therefore better than k-fold cross-validation as it is class sensitive (Prusty et al., 2022). A ten-block (k = 10) design used in the present study, with 90% of samples featured in every training run and 10% set aside for immediate performance testing. Such an arrangement guarantees that every subset reflects the complete class landscape, thereby fostering evaluations that are more robust and widely applicable.

Table 1. Information of Selected Attributes

Attribute Name	Information
WQI FC	Fecal Index. Fecal are organisms that originate from the digestive tract and waste of humans and animals.
WQI Oxy	Oxygen Index. The amount of oxygen dissolved in water comes from photosynthesis and oxygen diffusion from the air.
WQI pH	pH Index. pH is the degree of acidity of a solution. A pH value below 7 indicates the solution is acidic, above 7 indicates the solution is alkaline, where a value of 7 is considered neutral.
WQI TSS	Total suspended sediment (TSS) Index. TSS is the total mass of solid particles floating in the soil, without taking into account dissolved or sinking particles. TSS includes dust, soil, mud, organic debris and particles suspended in water.
WQI Temp	Temperature Index. It is a measure of the hot or cold intensity of water, affecting the quality and the sustainability of aquatic ecosystems.
WQI TPN	Nitrogen Index. Nitrogen is a compound that comes from agricultural, industrial and domestic waste.
WQI TP	Phosphorus Index. Phosphorus is a compound that comes from agricultural fertilizer, domestic waste or water flow from the surface.
WQI Turb	Turbidity Index. Turbidity is a measure of the extent to which solid particles are dispersed and dissolved in water.

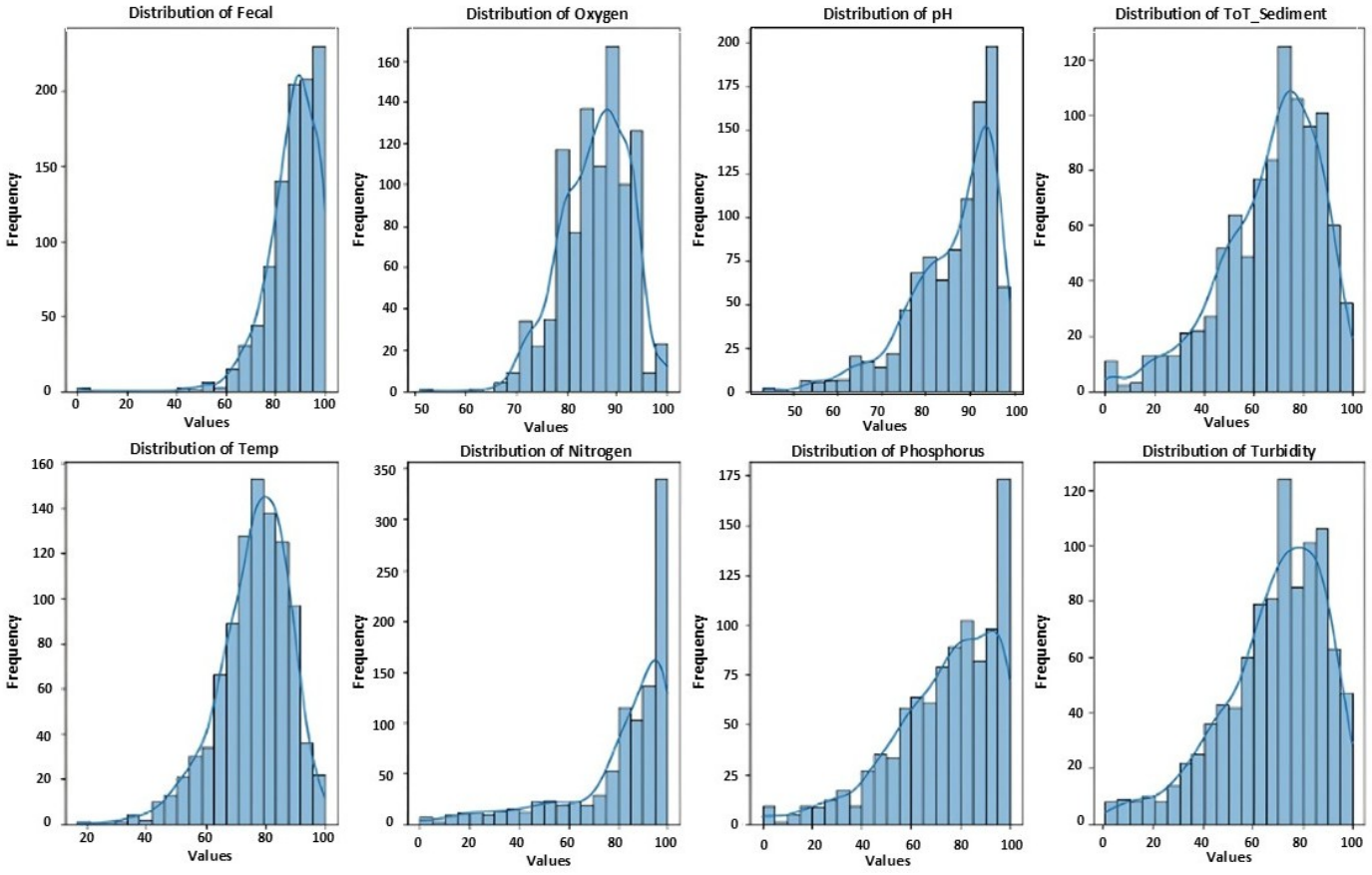

Figure 2. Numerical Feature Distribution of the Dataset

2.4 Classification with Decision Tree

2.4.1 Random Forest

Random Forest is a classification approach based on an ensemble of several decision trees, each built from a sample of the data. In this method, a random subset of attributes, denoted as F, is chosen to determine how to split each node at a decision tree (Parmar et al., 2019). With these features, Random

Table 2. CCME WQI Categories

Category	Range	Quality
1	95 – 100	Excellent
2	80 – 94	Good
3	65 – 79	Fair
4	45 – 64	Marginal
5	0 – 44	Poor

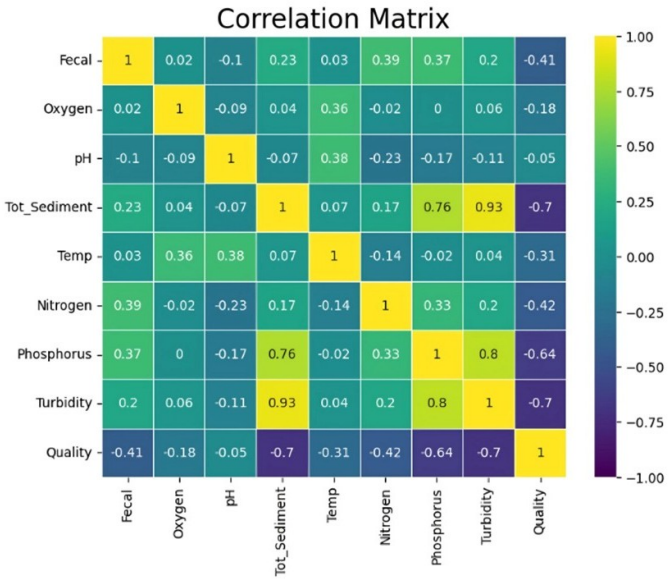

Figure 3. Correlation Matrix

Table 3. Dataset Comparison After SMOTE

Class	Before	After
1	27	470
2	346	470
3	470	470
4	88	470
5	40	470
Total	971	2350

Forest has been shown to outperform other learning methods because it has lower error rates and higher classification performance, as well as being able to work with large volume of training data and even incomplete information (Primajaya and Sari, 2018). Nurdin et al. (2024) showed that Random Forest Regression is better than Support Vector Regression in predicting vehicle fuel consumption, while Fitriyana et al. (2024) showed that Support Vector Machine is better at classifying Glycosylation in Lysine Protein Sequences compared to Random Forest. The algorithm of Random Forest uses a technique called bootstrap sampling, which involves creating multiple datasets by randomly selecting samples from the original dataset with replacement. Each decision tree in the forest is trained on a different bootstrap sample. If the original dataset has N samples, we generate a new bootstrap sample S_b of size N by randomly selecting samples from the dataset, with replacement, as formulated in Equation 1, with $x_i \in$ Dataset with replacement. This means some of the data points might appear multiple times in the same bootstrap sample, while others might not appear at all.

$$S_b = \{x_1, x_2, \ldots, x_N\} \tag{1}$$

This algorithm also employs the Gini index, defined in Equation 2, which quantifies the impurity of a split with respect to a population composition in a certain branch of the tree. Because of its dependability and consistency with complex datasets, this approach has become prevalent in areas such as medical diagnosis, financial forecasting, and recommendation system development (Dikananda et al., 2022). P_i is a symbol that shows the frequency probability of the i-th class in the dataset, whereas C represents the number of classes. Gini index measures the impurity of a node, and a lower value indicates a more "pure" node, meaning it's more likely to contain data points from a single class. Therefore, choosing the feature with the lowest Gini index for the root node (and subsequent nodes) helps create a more reliable decision tree (Pavlov, 2000).

$$Gini(S) = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{2}$$

In addition to using the Gini index, decision tree construction can also utilize entropy as a measure of attribute impurity. Entropy quantifies the uncertainty within a dataset and helps in determining how well an attribute can separate the data into distinct classes. The entropy value can be calculated using Equation 3, with S represents the set of datasets.

$$Entropy(S) = -\sum_{i=1}^{c} p_i \cdot \log_2(p_i) \tag{3}$$

Random Forest in classification tasks applies majority voting in predicting the class label. If each tree in the forest predicts a class, the final prediction is the class that appears the most across all tree, as shown in Equation 4, with T is the number of trees, and $\hat{y}_i$ is the prediction of tree i.

$$\hat{y} = \text{mode}(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T) \tag{4}$$

Random Forest has a feature called Out-of-Bag (OOB) error estimation, where for each training sample, the model uses the trees that did not include that sample in their bootstrap sample to test the prediction. This gives an unbiased estimate of the model's accuracy, without needing a separate validation set. The formula is represented in Equation 5, where N_{OOB} is the number of out-of-bag samples, and $I(y_i \neq \hat{y}_i)$ is an indicator

function that equals 1 if the true value y_i is different from the predicted value y_i (Pavlov, 2000).

$$OOB_{\text{error}} = \frac{1}{N_{\text{OOB}}} \sum_{i \in \text{OOB}} I(y_i \neq \hat{y}_i) \tag{5}$$

2.4.2 XGBoost

XGBoost utilizes an ensemble approach, improving upon the traditional gradient tree boosting algorithm. This increases its effectiveness for large-scale machine learning tasks. XGBoost performs well at complex models because of advanced features built into the system to enhance computing speed and minimize overfitting (Chen and Guestrin, 2016). Like most machine learning systems, the first step for XGBoost is to define an objective function. Mathematically, for multi-class classification, the objective function is a combination of the loss function and regularization. The training loss in this case is an error of a prediction that is made and which has to be minimized. To control overfitting, complexity of the model is also controlled by a regularization term. As highlighted, XGBoost's balance of error minimization and complexity control helps it achieve a level of robustness and generalization with its models. The structure of this function is provided in detail in Equation 6, where $\mathscr{L}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$ is the loss function for multi-class, $\Omega(f_k)$ is the regularization term that penalizes tree complexity, is T the total number of trees, and N is the number of training samples (He, 2023).

$$\mathcal{O}(f) = \sum_{i=1}^{N} \mathscr{L}(y_i, \hat{y}_i) + \sum_{k=1}^{T} \Omega(f_k) \tag{6}$$

The loss function used for multi-class classification is the *softmax* loss, which generalizes the binary log loss to multiple classes, as formulated in Equation 7, where K is the number of classes, $\mathbb{1}(y_i = k)$ is an indicator function that equals 1 if the true class of the sample i is class k, and 0 otherwise, and $\hat{p}_{ik} = \frac{e^{f_k(x_i)}}{\sum_{j=1}^{K} e^{f_j(x_i)}}$ is the predicted probability for class k for sample i, given the model output $f_k(x_i)$. The regularization term $\Omega(f_k)$ is designed to penalize overly complex trees. This term is computed for each decision tree in the model as in Equation 8, with γ controls the number of leaves in the tree and λ is the L2 regularization term, which penalizes large weights (w_j) in the tree's leaf nodes.

$$\mathscr{L}(y_i, \hat{y}_i) = -\sum_{k=1}^{K} \mathbb{1}(y_i = k) \log \hat{p}_{ik} \tag{7}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{8}$$

XGBoost refines predictions iteratively, incorporating new information at each step, denoted as $\hat{y}_i^{(t)}$ (XGBoost Developers, 2023), as defined in Equation (9). The symbol $\hat{y}_i^{(t)}$ represents the prediction at the t-th iteration for the i-th data point, with $f_k(x_i)$ denoting the predictor function produced by the k-th model at the k-th iteration, and t as the total number of iterations.

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{9}$$

XGBoost also uses gradient boosting to minimize the loss function, and it uses both the first-order gradient (g) and second-order Hessian (h) to update the model in each iteration. For multi-class classification, the calculations are extended to handle each class. The gradient of the softmax loss function with respect to the predicted value $f_k(x_i)$ for class k is denoted in Equation 10, and the Hessian (second derivative) for class k with respect to $f_k(x_i)$ is denoted in Equation 11.

$$g_{ik} = \frac{\partial \mathscr{L}(y_i, \hat{y}_i)}{\partial f_k(x_i)} = \hat{p}_{ik} - \mathbb{1}(y_i = k) \tag{10}$$

$$h_{ik} = \frac{\partial^2 \mathscr{L}(y_i, \hat{y}_i)}{\partial f_k(x_i)^2} = \hat{p}_{ik}(1 - \hat{p}_{ik}) \tag{11}$$

At each node of the tree, XGBoost chooses the best feature and split to minimize the objective function. This is done by calculating the gain for each possible split. For a given split in the decision tree, the gain is the reduction in the objective function (the total loss) caused by splitting the data at that node, as calculated in Equation 12, with G_L and G_R are the gradients for the left and right child nodes and H_L and H_R are the Hessians for the left and right child nodes (He, 2023).

$$\text{Gain} = \frac{1}{2}\left(\frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \frac{G_L^2}{H_L + \lambda} - \frac{G_R^2}{H_R + \lambda}\right) - \gamma \tag{12}$$

After training all the trees, the final predictions for each class are computed using the *softmax* function. If the final prediction for each class k at iteration t is represented as $f_k^{(t)} = \sum_{j=1}^{T} f_{k,j}(x)$, with $f_{k,j}(x)$ is the score from class k from the j-th tree, the probability for class k is then given in Equation 13, where $\hat{p}_{ik}$ is the predicted probability for class k for sample i, and the denominator is the sum of exponentials over all classes to normalize the probabilities.

$$\hat{p}_{ik} = \frac{e^{f_k^{(t)}}}{\sum_{k'=1}^{K} e^{f_{k'}^{(t)}}} \tag{13}$$

The final class prediction for a given sample is the class with the highest predicted probability, denoted as $\hat{y}_i = \arg\max_k \hat{p}_{ik}$.

2.4.3 C5.0

The C5.0 algorithm is an enhancement to the classification approach of data mining of decision trees containing advanced features like ID3 and C4.5 by Ross Quinlan (Pandya and Pandya, 2015). C4.5 and C5.0 both calculated entropy and information gain for the decision trees, but C5.0 has further enhancement for selection of attributes by splitting nodes using the gain ratio. This guarantees that the most informative attribute is at the parent node, thereby increasing the accuracy and efficiency of the decision tree (Myint and Tin, 2021). The entropy calculation is similar as in other decision trees algorithms, such as the entropy in Random Forest given in Equation 3. Information gain measures how much entropy is reduced by splitting the data based on a particular feature. The goal is to select the feature that provides the largest reduction in uncertainty (entropy). The information gain for an attribute A is calculated as in Equation 14, with $E(D)$ is the entropy for the entire dataset, $Values(A)$ is the set of possible values for the attribute A, D_v is the subset of D where attribute A has value v, and $E(D_v)$ is the entropy of the subset D_v.

$$IG(D, A) = E(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} E(D_v) \tag{14}$$

The bias towards distinct valued attributes is shown by the gain ratio, detailed in Equation 15, which makes C5.0 a very tough and reliable system to use for various classification tasks. Equation 15 shows the calculation of gain ratio for a feature A, with $IG(D, A)$ represents the information gain for attribute A and $E(A)$ represents the split information for attribute A. Split Information is a measure of how much information is needed to describe the possible splits of an attribute. It is used to avoid bias toward attributes with many distinct values (which might result in overfitting). The split information for an attribute is the entropy of the split that the attribute creates when it divides the dataset into distinct subsets, as defined in Equation 16 (Myint and Tin, 2021).

$$GR(D, A) = \frac{IG(D, A)}{E(A)} \tag{15}$$

$$E(A) = -\sum_{v \in values(A)} \frac{|D_v|}{|D|} \log_2 \left(\frac{|D_v|}{|D|} \right) \tag{16}$$

The decision tree is built recursively by splitting the dataset at each node based on the attribute that provides the highest gain ratio. The steps are as follows: (1) Calculate the entropy of the dataset; (2) Compute the information gain for each attribute and the gain ratio; (3) Split the data based on the attribute with the highest gain ratio; and (4) Repeat the process recursively on the resulting subsets until a stopping criterion is met (e.g., maximum depth or minimum subset size). After the tree is constructed, C5.0 applies a pruning step to remove branches that do not contribute to improving the model's performance. The pruning process is typically post-pruning, meaning it occurs after the full tree is constructed. During pruning, C5.0 evaluates the error rate at each node, and if pruning a node result in a lower error rate, the node is removed.

2.5 Evaluation Metrics

This study evaluated the classification model using test data with known actual values through a confusion matrix. A confusion matrix is a tabular description of a classification process that offers detailed insights thorough comparison of actual classifications and predicted classifications (Joloudari et al., 2020). The classical binary class confusion matrix (as illustrated in Table 4), consists of consists of Predicted and Actual value combinations includes: True Positive (TP); True Negative (TN); False Positive (FP); and False Negative (FN) (Theissler et al., 2022). In this context, TP indicates correct positive prediction, TN indicates correct negative prediction, FP indicates incorrect positive prediction, and FN indicates incorrect negative prediction. Accuracy, precision, recall, and F1 score, which are key performance indicators, can be derived using the confusion matrix main components as shown in Equation 17 to Equation 20. Such metrics are critical for assessing overall model effectiveness, error types, and steps toward enhanced model accuracy and reliability.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{17}$$

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{20}$$

However, for a multiclass problem, the confusion matrix is slightly extended, and metrics calculations are slightly modified to evaluate the performance of a multi-class classification model. The illustration of a multi-class confusion matrix is presented in Table 5, and the modified formulas are available in Equation 21 to Equation 24, with MC refers to multiclass, and L_i indicates that the calculation is for the specific class L_i (Markoulidakis et al., 2021).

$$MC\_Accuracy = \frac{\sum_{i=1}^{C} TP_{L_i}}{\sum_{i=1}^{C} \sum_{j=1}^{C} L_{ij}} \tag{21}$$

$$MC\_Precision_{L_i} = \frac{TP_{L_i}}{TP_{L_i} + FP_{L_i}} \tag{22}$$

Table 4. Evaluation Result of Each Class Using Hold-Out

Class	Accuracy			Precision			Recall			F1-Score		
	RF	XGB	C5.0	RF	XGB	C5.0	RF	XGB	C5.0	RF	XGB	C5.0
1	1.000	0.995	0.991	1.000	0.979	0.979	1.000	1.000	0.979	1.000	0.989	0.979
2	0.957	0.940	0.931	0.870	0.821	0.803	0.940	0.920	0.900	0.903	0.867	0.849
3	0.953	0.927	0.919	0.923	0.885	0.857	0.818	0.704	0.681	0.867	0.784	0.759
4	0.991	0.970	0.970	0.961	0.905	0.890	1.000	0.960	0.980	0.980	0.932	0.933
5	0.995	0.987	0.991	1.000	0.976	1.000	0.976	0.953	0.953	0.988	0.964	0.976

Table 5. Average Results with Hold Out

Method	Accuracy	Precision	Recall	F1-Score	Runtime
Random Forest	0.979	0.950	0.946	0.947	429.806 ms
XGBoost	0.964	0.913	0.907	0.907	315.998 ms
C5.0	0.960	0.906	0.898	0.899	62.28 ms

Table 6. Classical Binary Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 7. Multi-Class Confusion Matrix

Predicted\Actual	Class 1	Class 2	⋯	Class C
Class 1	TP_1	$FP_{1,2}$	⋯	$FP_{1,C}$
Class 2	$FP_{2,1}$	TP_2	⋯	$FP_{2,C}$
⋯	⋯	⋯	⋯	⋯
Class C	$FP_{C,1}$	$FP_{C,2}$	⋯	TP_C

$$MC\_Recall_{L_i} = \frac{TP_{L_i}}{TP_{L_i} + FN_{L_i}} \tag{23}$$

$$MC\_F1Score_{L_i} = 2 \times \frac{(MC\_Precision_{L_i} \times MC\_Recall_{L_i})}{(MC\_Precision_{L_i} + MC\_Recall_{L_i})} \tag{24}$$

3. RESULTS AND DISCUSSION

Classification tasks are executed using the Python programming language. The evaluation is carried out in three phases, each based on the data-splitting approach to ensure a comprehensive assessment of the model's performance. However, the parameters for each algorithm remained consistent across all splitting approaches.

The Random Forest algorithm was configured with n_estimators = 100, meaning it builds 100 decision trees to make predictions. Each tree is trained independently on a random sample of the data, and the final decision is made by combining the outputs of all trees through majority voting. To prevent the trees from becoming overly complex and overfitting the training data, the maximum depth of each tree was limited to 5. This depth restriction helps the model capture meaningful patterns while maintaining its ability to generalize well to unseen data.

For the XGBoost algorithm, the parameter n_estimators = 100 sets the total number of decision trees that the model will build one after another. Each tree tries to fix the mistakes made by the previous ones, helping the model improve step by step. The learning_rate of 0.1 controls how much each new tree affects the overall prediction. A smaller learning rate means the model learns more slowly but often ends up generalizing better to new data. To keep the trees simple and avoid overfitting, max_depth is limited to 2, meaning each tree can only grow to two levels deep. Additionally, the model uses subsample = 0.8 and colsample_bytree = 0.8, which means that in each training round, only 80% of the data and 80% of the features are randomly selected. This randomness helps the model avoid relying too much on any specific data or feature and makes it more robust.

On the other hand, the C5.0 algorithm works a bit differently. Setting subset = True allows it to automatically pick the most relevant subsets of features, making the model focus on what really matters. By turning off winnow (setting it to False), the model keeps all features during training instead of trying to filter out less important ones. The confidence factor (CF = 0.1) controls how aggressively the model prunes the decision tree-a lower value means more pruning to keep the tree simpler and reduce the risk of overfitting. With minCases = 10, the algorithm only splits a node if it has at least ten data points, which also helps keep the model from getting too complex. The training process is further improved by earlyStopping=True, which stops training early if the model's performance stops getting

Table 8. Evaluation Result of Each Class Using SKCV

Class	Accuracy			Precision			Recall			F1-Score		
	RF	XGB	C5.0	RF	XGB	C5.0	RF	XGB	C5.0	RF	XGB	C5.0
1	0.998	0.994	0.995	0.993	0.979	0.985	1.000	0.994	0.991	0.996	0.987	0.988
2	0.965	0.954	0.950	0.912	0.876	0.877	0.912	0.902	0.872	0.912	0.888	0.875
3	0.946	0.929	0.929	0.887	0.858	0.833	0.838	0.774	0.808	0.862	0.814	0.820
4	0.979	0.963	0.968	0.928	0.888	0.905	0.972	0.936	0.942	0.950	0.911	0.923
5	0.997	0.990	0.989	0.993	0.976	0.978	0.995	0.974	0.968	0.994	0.975	0.973

better. Like in XGBoost, the maximum depth of trees is set to 2 with maxDepth = 2 to keep the trees shallow. The boosting process runs for 100 iterations (trials = 100) to gradually build a stronger model. Other parameters like bands = 0, sample = 0, and fuzzyThreshold = False are left at default, meaning no special sampling or fuzzy logic is applied.

Overall, these settings reflect a careful balance between building a model that's accurate but not too complicated. By limiting tree depth, introducing randomness through sampling, and applying pruning and early stopping, the models are designed to generalize well without overfitting. Choosing these parameters thoughtfully is key to creating effective and interpretable machine learning models.

3.1 Evaluation Using the Hold-Out Method

In the initial experiment, the dataset is split using the hold-out method, with 90% of the data allocated for training and 10% for testing. Table 6 juxtaposes three classification methods-Random Forest (RF), XGBoost (XGB), and C5.0 - presenting their performance across all classes. The performance metrics calculated for each class include accuracy, precision, recall, and F1-score.

Table 6 shows that Random Forest (RF) performs well for all of the metrics and classes throughout. For instance, in Class 1, Random Forest achieves astounding results, having received an accuracy, precision, recall, and F1 score all equal to 1.000, outperforming XGBoost and C5.0 for this class. XGBoost and C5.0 also perform well but show slight variations in their metrics.

Table 7 summarizes average performance outcomes for three decision-tree variants, each evaluated through the standard hold-out scheme of 90:10. Random Forest (RF) tops every available metric, yet its mean runtime of 429.806 milliseconds reveals a significant computational burden. In contrast, the C5.0 algorithm, while lagging slightly behind on the same measurements, completes its run in just 62.28 milliseconds and thus earns the title of fastest algorithm among them. XGBoost, parked between the extremes of accuracy and speed, surfaces as a reasonable, middle-ground alternative.

3.2 Evaluation Using the Stratified K-Fold Cross Validation (SKVC) Method

In the next experiment, the dataset is split using SKCV with k = 10. Table 8 displays the performance outcomes for the three classification methods. Similar to the hold-out results, RF consistently achieves slightly higher metrics results compared to XGBoost and C5.0. Interestingly, the class with the highest metrics for all algorithms is the one that was initially a minority class before applying SMOTE (see Table 3), whereas Class 3, which had no instances added after SMOTE, scores the lowest. The recall metric for XGBoost in Class 3 is lower compared to the other algorithms, indicating a higher rate of false negatives. The lower recall for XGBoost suggests it is less reliable for detecting all positive instances in Class 3, which could be due to the characteristics of the data. Despite this, both XGBoost and C5.0 also demonstrate competitive results overall.

The average performance results for three classification methods-Random Forest (RF), XGBoost (XGB), and C5.0 - are detailed in Table 9. In line with the hold-out results, Random Forest demonstrates the highest performance across all metrics. However, it has the longest runtime at 8584.102 milliseconds, attributed to its nature of creating multiple trees. XGBoost and C5.0 exhibit very similar performance metrics, with both methods achieving an accuracy of 0.966. Precision and recall are nearly identical for both. Despite this, XGBoost's runtime is significantly longer, at 4181.01 milliseconds, which is more than twice that of C5.0's 1771.632 milliseconds. This makes C5.0 preferable over XGBoost in this particular experiment, especially when computational efficiency is a priority.

3.3 Comparison of the Methods

Figure 4 shows the comparison of training and validation accuracy for Random Forest, XGBoost, and C5.0 algorithms using the SKCV method. The results indicate that the accuracy values for training and validation are very close for all models, which is a good sign. This means the models have learned well from the training data without overfitting (too closely fitting the training data) or underfitting (not learning enough). Training accuracy reflects how well the model performs on the data it was trained on, while validation accuracy indicates how well it generalizes to new, unseen data during training. When these two metrics are both high and similar, it typically means the model is balanced and likely to perform well on new data. Conversely, a large gap with higher training accuracy can signal overfitting, and low values for both may indicate underfitting. Overall, the results in Figure 4 suggest that the models are well-tuned and capable of delivering reliable predictions for this dataset.

Figure 5 summarizes confusion-matrix results for three

Table 9. Average Results with SKCV

Method	Accuracy	Precision	Recall	F1-Score	Runtime
Random Forest	0.977	0.943	0.943	0.943	8584.102 ms
XGBoost	0.966	0.915	0.916	0.915	4181.01 ms
C5.0	0.966	0.916	0.916	0.916	1771.632 ms

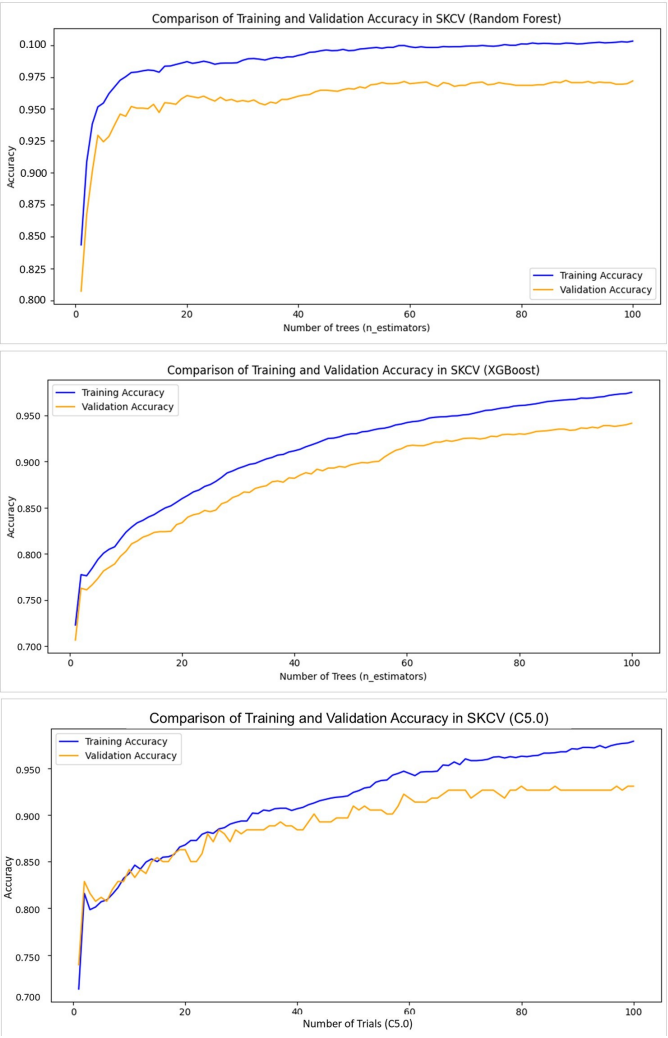

Figure 4. Comparison of Training and Validation Accuracy of the Decision Tree Algorithms

popular decision-tree algorithms applied to the test set. Across both the simple hold-out partition and the more exhaustive stratified k-fold cross-validation, the Random Forest variant regularly records the highest numbers. Accuracy values for that method hover just under 0.98 in both settings. In the same comparisons, precision, recall, and the combined F1 score remain reassuringly strong, signaling the model's overall reliability. C5.0 and XGBoost trail by only a few decimal points, both landing in the 0.96. A curious rebound shows with SKCV, however, where C5.0 sometimes equals or narrowly exceeds XGBoost on the measurements, hinting at the classifier's sensitivity to how the data is partitioned.

Examination of the runtimes presented in Tables 4 and 6 in Sections 3.1 and 3.2 reveals an inherent imbalance when hold-out and SKCV times are weighed against one another. The former executes a single 90:10 partition, while the k = 10 SKCV scheme repeats that split ten separate times. Even so, both splitting strategies conclude the same lesson about Random Forest: its ensemble nature drives up wall time, a liability that shows up when response time is crucial. C5.0, by contrast, gives nearly identical accuracy numbers yet finishes the work much faster, making it the obvious pick for high-throughput tasks. XGBoost lands midway between the two, faster than Random Forest but still slower than C5.0, and while that stride can feel satisfactory it sits slightly behind the latter in sheer speed. Based on these findings, the best model depends on the application scenario as shown in Table 10.

In the context of predicting water quality, RF's accuracy is ideal in some tasks, but it may not be suitable for applications with low latency requirements; for example, in systems deployed for water quality monitoring in real time. For scenarios where speed is more desirable such as in real-time classification, or settings with limited resources like IoT-based water monitoring, C5.0 works efficiently. XGBoost, which works better than C5.0 but is slower, is ideal for medium to large datasets where reasonable accuracy with efficient processing is needed. If subsequent studies include more features or other parameters of water, XGBoost might have better scalability than Random Forest. Also, in water quality monitoring, balanced evaluation is necessary because some levels of contamination are not represented adequately in the unprocessed data. SKCV makes the estimates more applicable to the real world by minimizing the variance observed in the results.

From the evaluation results, it is evident that some water quality classes are more challenging to classify correctly than others. Notably, Class 3 (Fair quality) consistently shows lower recall and precision across all models compared to other classes as shown in Table 6 and Table 8. This difficulty likely arises because the feature distributions for Class 3 significantly overlap with those of Class 2 (Good) and Class 4 (Marginal), making it hard for the models to distinguish between them. Although SMOTE was applied to address class imbalance, the subtle differences in important parameters such as nitrogen concentration and turbidity still lead to misclassifications among these classes.

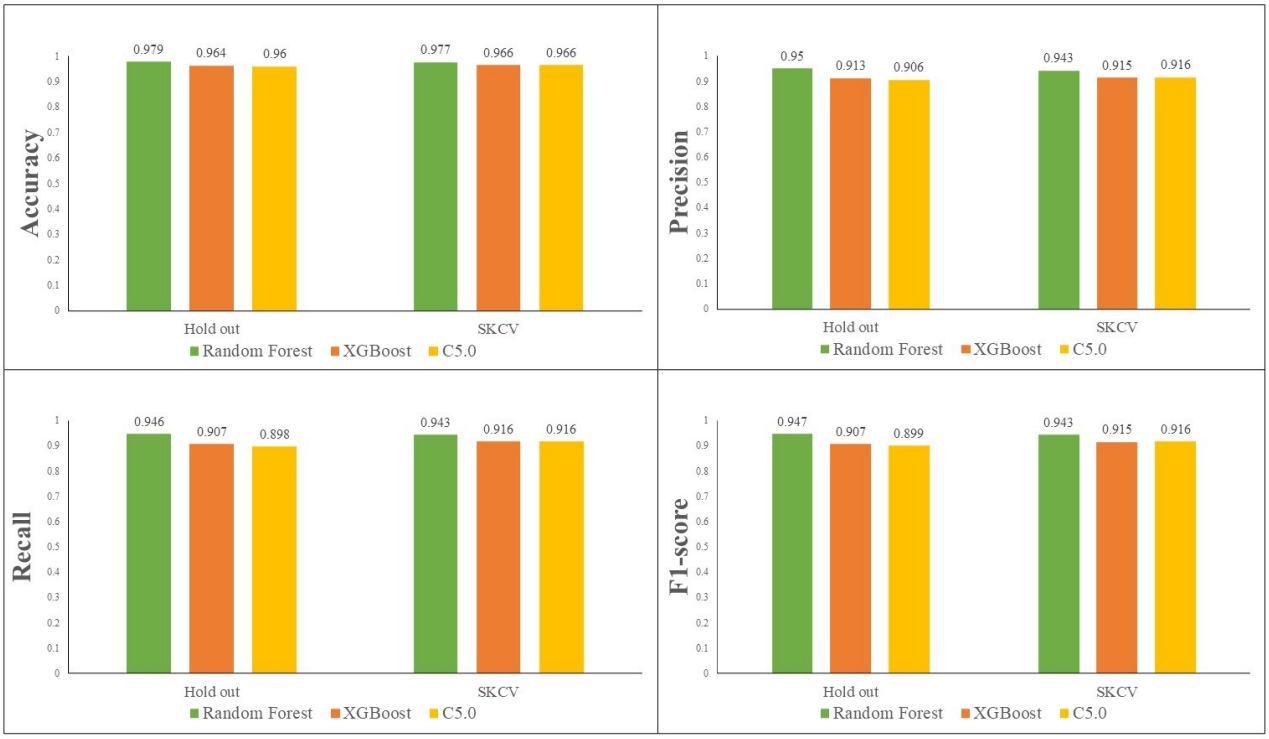

Figure 5. Comparison of Decision Tree Algorithms (RF, XGBoost, and C5.0)

Table 10. Best Decision Tree based on Application Scenario

Scenario	Best Decision Tree	Reason
Highest Accuracy	Random Forest	Best predictor among the three but has high computational cost.
Fastest Execution	C5.0	Maintains high accuracy with minimal runtime.
Balanced Trade-Off	XGBoost	Good accuracy and efficiency but slower than C5.0.
Real-Time Water Quality Monitoring	C5.0 or XGBoost	Fast response time is critical.
Large-scale Datasets	XGBoost	Scales better than Random Forest in big data.

These misclassifications can have practical implications. Incorrectly labeling water quality levels could result in inappropriate treatment decisions or insufficient monitoring, potentially impacting environmental management and public health outcomes. To address this, future research could focus on incorporating additional or more discriminative features that better separate closely related classes. Applying explainability techniques such as SHapley Additive exPlanations (SHAP) or Local Interpretable Model-agnostic Explanations (LIME) would help uncover which features contribute most to the model's predictions and identify sources of confusion.

Furthermore, exploring alternative machine learning models or ensemble approaches that combine strengths of different classifiers could enhance accuracy. A hybrid data augmentation strategy that blends oversampling (like SMOTE) and undersampling methods may better balance the dataset and reduce overfitting. Lastly, expanding the dataset size and improving data quality, particularly for underrepresented classes, will likely improve model robustness and reduce misclassification rates.

4. CONCLUSIONS

Through systematic analysis, this research assesses the effectiveness of three decision tree-based models: Random Forest, XGBoost, and C5.0, in order to forecast water-quality ratings. The classification step is preceded by a thorough data analysis and preprocessing step to get better insight from the dataset. The data was processed using the SMOTE technique to enhance the class imbalance problem. Although Random Forest achieved the best accuracy of almost 0.98, the model is unsuitable for scenarios where time is of the essence such as real-time water monitoring system, due to a longer runtime. C5.0 is much faster and has a run time of approximately 0.96 which makes the model a good-fit for cases where speed is important. XGBoost performed similarly to C5.0, but was less efficient in terms of runtime. All experimentation relies on a medium-sized set of 971 rows with 10 feature columns. For

bigger and complicated datasets, XGBoost might outperform, whereas Random Forest will have even slower performance because of its process of multiple-tree generation. In future studies, the efficiency of Random Forest and XGBoost models can be improved by using the pruning techniques alongside parallel computing or graphic processing unit (GPU) acceleration, which is expected to solve the models' high runtime. In addition, to improve transparency, predicting SVM, KNN, or even Neural Networks performance can be benchmarked against other non-tree classifiers alongside SHAP and LIME explainability techniques. Lastly, considering the issues of overfitting, dataset scalability, and generalizability will fortify the study's concern for real world application.

5. ACKNOWLEDGMENT

The authors sincerely thank the University of Lampung for its financial assistance from the research fund, which has contributed immensely towards completing this work and furthering our research.

REFERENCES

Chen, T. and C. Guestrin (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 785–794

Dikananda, A. R., S. Jumini, N. Tarihoran, S. Christinawati, W. Trimastuti, and R. Rahim (2022). Comparison of Decision Tree Classification Methods and Gradient Boosted Trees. *TEM Journal*, **11**(1); 316–322

Eyring, V., W. D. Collins, P. Gentine, E. A. Barnes, M. Barreiro, T. Beucler, and L. Zanna (2024). Pushing the Frontiers in Climate Modelling and Analysis with Machine Learning. *Nature Climate Change*, **14**(9); 916–928

Filho, W. L., A. M. Azul, L. Brandli, A. L. Salvia, and T. Wall (2022). *Clean Water and Sanitation*. Springer International Publishing

Fitriyana, S., A. Syarif, F. Rossyking, and M. R. Faisal (2024). Application of Random Forest Method Classification for Glycosylation in Lysine Protein Sequences. *Integra: Journal of Integrated Mathematics and Computer Science*, **1**(2); 49–54

Garcia, S., J. Luengo, F. Herrera, et al. (2015). *Data Preprocessing in Data Mining*, volume 72. Springer

García, S., S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera (2016). Big Data Preprocessing: Methods and Prospects. *Big Data Analytics*, **1**(1); 1–22

Ghazvini, A., J. Awwalu, and A. Abu Bakar (2014). Comparative Analysis of Algorithms in Supervised Classification: A Case Study of Bank Notes Dataset. *International Journal of Computer Trends and Technology*, **17**(1); 39–43

Gikas, G. D., G. K. Sylaios, V. A. Tsihrintzis, I. K. Konstantinou, T. Albanis, and I. Boskidis (2020). Comparative Evaluation of River Chemical Status Based on WFD Methodology and CCME Water Quality Index. *Science of the Total Environment*, **745**; 140849

He, Y.-H. (2023). *Machine Learning in Pure Mathematics and Theoretical Physics*. World Scientific

Holcomb, D. A. and J. R. Stewart (2020). Microbial Indicators of Fecal Pollution: Recent Progress and Challenges in Assessing Water Quality. *Current Environmental Health Reports*, **7**; 311–324

Ilić, M., Z. Srdjević, and B. Srdjević (2022). Water Quality Prediction Based on Naive Bayes Algorithm. *Water Science and Technology*, **85**(4); 1027–1039

Joloudari, J. H., M. Haderbadi, A. Mashmool, M. Ghasemigol, S. S. Band, and A. Mosavi (2020). Early Detection of the Advanced Persistent Threat Attack Using Performance Analysis of Deep Learning. *IEEE Access*, **8**; 186125–186137

Komorowski, M., D. C. Marshall, J. D. Salciccioli, and Y. Crutain (2016). Exploratory Data Analysis. In MIT Critical Data, editor, *Secondary Analysis of Electronic Health Records*. Springer Cham, pages 185–203

Majumder, M. G., S. D. Gupta, and J. Paul (2022). Perceived Usefulness of Online Customer Reviews: A Review Mining Approach Using Machine Learning & Exploratory Data Analysis. *Journal of Business Research*, **150**; 147–164

Markoulidakis, I., G. Kopsiaftis, I. Rallis, and I. Georgoulas (2021). Multi-Class Confusion Matrix Reduction Method and Its Application on Net Promoter Score Classification Problem. In *Proceedings of the 14th Pervasive Technologies Related to Assistive Environments Conference*. pages 412–419

Mutoffar, M. M., M. Naseer, and A. Fadillah (2022). Klasifikasi Kualitas Air Sumur Menggunakan Algoritma Random Forest. *NARATIF: Jurnal Ilmiah Nasional Riset Aplikasi dan Teknik Informatika*, **4**(2); 138–146

Myint, K. L. and H. H. K. Tin (2021). Analyzing the Comparison of C4.5, CART and C5.0 Algorithms on Heart Disease Dataset Using Decision Tree Method. In *ICIDSSD 2020*. page 174

Nasir, N., A. Kansal, O. Alshaltone, F. Barneih, M. Sameer, A. Shanableh, and A. Al-Shamma'a (2022). Water Quality Classification Using Machine Learning Algorithms. *Journal of Water Process Engineering*, **48**; 102920

Nurdin, M., Wamiliana, A. Junaidi, and F. R. Lumbanraja (2024). Comparison of Support Vector Regression and Random Forest Regression Performance in Vehicle Fuel Consumption Prediction. *Integra: Journal of Integrated Mathematics and Computer Science*, **1**(2); 60–67

Pandya, R. and J. Pandya (2015). C5.0 Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning. *International Journal of Computer Applications*, **117**(16); 18–21

Pansara, R. R., B. Y. Kasula, A. B. Bhatia, and P. Whig (2024). Enhancing Sustainable Development Through Machine Learning-Driven Master Data Management. In *International Conference on Sustainable Development Through Machine Learning, AI and IoT*. pages 332–341

Parmar, A., R. Katariya, and V. Patel (2019). A Review on Random Forest: An Ensemble Classifier. In *International Conference on Intelligent Data Communication Technologies and*

Internet of Things (ICICI) 2018. pages 758–763

Pavlov, Y. L. (2000). *Random Forests*. VSP

Pradhan, P., L. Costa, D. Rybski, W. Lucht, and J. P. Kropp (2017). A Systematic Study of Sustainable Development Goal (SDG) Interactions. *Earth's Future*, **5**(11); 1169–1179

Pradipta, G. A., R. Wardoyo, A. Musdholifah, I. N. H. Sanjaya, and M. Ismail (2021). SMOTE for Handling Imbalanced Data Problem: A Review. In *2021 Sixth International Conference on Informatics and Computing (ICIC)*. pages 1–8

Primajaya, A. and B. N. Sari (2018). Random Forest Algorithm for Prediction of Precipitation. *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)*, **1**(1); 27–31

Prusty, S., S. Patnaik, and S. K. Dash (2022). SKCV: Stratified K-Fold Cross-Validation on ML Classifiers for Predicting Cervical Cancer. *Frontiers in Nanotechnology*, **4**; 972421

Rolnick, D., P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. (2022). Tackling Climate Change with Machine Learning. *ACM Computing Surveys (CSUR)*, **55**(2); 1–96

Shen, C. (2018). A Transdisciplinary Review of Deep Learning Research and Its Relevance for Water Resources Scientists. *Water Resources Research*, **54**(11); 8558–8593

Theissler, A., M. Thomas, M. Burch, and F. Gerschner (2022). ConfusionVis: Comparative Evaluation and Selection of Multi-Class Classifiers Based on Confusion Matrices. *Knowledge-Based Systems*, **247**; 108651

UNICEF (2019). Progress on Household Drinking Water, Sanitation and Hygiene, 2000–2017. Accessed: 2025-06-13

XGBoost Developers (2023). XGBoost Release 1.5.0-Dev. Accessed: 2025-06-13

Xu, X., T. Lai, S. Jahan, F. Farid, and A. Bello (2022). A Machine Learning Predictive Model to Detect Water Quality and Pollution. *Future Internet*, **14**(11); 324

Yogeshwari, A., J. Anubama, M. J. M. M. Jenitha, M. C. Geetha, M. D. Ramalakshmi, and B. Pavithra (2023). Water Quality Prediction Using CatBoost Classifier Algorithm. *Journal of Survey in Fisheries Sciences*, **10**(4S); 2850–2855

Yuan, Q., H. Shen, T. Li, Z. Li, S. Li, Y. Jiang, H. Xu, W. Tan, Q. Yang, J. Wang, et al. (2020). Deep Learning in Environmental Remote Sensing: Achievements and Challenges. *Remote Sensing of Environment*, **241**; 111716

Zhu, M., J. Wang, X. Yang, Y. Zhang, L. Zhang, H. Ren, L. Ye, et al. (2022). A Review of the Application of Machine Learning in Water Quality Evaluation. *Eco-Environment & Health*, **1**(2); 107–116

