CAUCHY – Jurnal Matematika Murni dan Aplikasi
Volume 11 (2) (2026), Pages 1618–1634
p-ISSN: 2086-0382; e-ISSN: 2477-3344

Optimization of CTGAN for Imbalanced Classification using RSCV-Ensemble Learning for Clean Water Quality Classification

Favian Sis Bagus Febrianto, Umu Sa'adah*, and Imam Nurhadi Purwanto

Department of Mathematics, Faculty of Mathematics and Natural Sciences, Brawijaya University, Malang, Indonesia

Abstract

Class imbalance is a common challenge in classification tasks because it often reduces model performance in identifying minority class observations. This study evaluates the performance of an Ensemble Learning model optimized using Randomized Search Cross Validation (RSCV) for clean water quality classification under imbalanced class distributions. The dataset was derived from the Water Pollution and Disease dataset, which originally contained 3,000 observations and 24 features without class labels. A data wrangling and labeling process produced a binary classification dataset with a 7:3 class ratio. To address the imbalance problem, Conditional Tabular Generative Adversarial Network (CTGAN) and SMOTENC oversampling techniques were applied and compared. Model performance was evaluated using accuracy, precision, recall, F1-score, and AUC-ROC across 25 data splits and statistically examined using two-sided t-tests. The best performance was achieved by the RSCV Ensemble Learning model combined with CTGAN, yielding mean values of 92.80% accuracy, 90.23% precision, 90.98% recall, 90.82% F1-score, and 96.80% AUC-ROC. The results indicate that CTGAN consistently outperforms SMOTENC and that RSCV optimization further improves classification performance.

Keywords: CTGAN; SMOTENC; Random Forest; Randomized Search CV; Water Quality.

Copyright © 2026 by Authors, Published by CAUCHY Group. This is an open access article under the CC BY-SA License (https://creativecommons.org/licenses/by-sa/4.0)

1. Introduction

Data classification is one of the main techniques in machine learning and has been widely applied in various fields such as healthcare, environment, business, and finance. In this process, the model learns patterns from historical data and then classifies new observations into predefined categories. However, classification tasks often face the challenge of class imbalance [1]. This condition occurs when the number of samples in one class is much larger than in others, which causes the model to favor the majority class and weakens its ability to accurately identify the minority class [2]. In the environmental context, a similar issue is found in clean water quality datasets, where samples representing unsafe water conditions are generally much fewer than those representing safe water. This imbalance can reduce the effectiveness of classification models in detecting high-risk environmental conditions, ultimately compromising the reliability of water quality monitoring systems [3][4].

Various studies have shown that class imbalance can affect the validity of model evaluation metrics [5]. A model may achieve a high accuracy score, yet still perform poorly in recognizing

*Corresponding author. E-mail: u.saadah@ub.ac.id

Received: April 27, 2026 Revised: May 28, 2026 Accepted: June 23, 2026
DOI: https://doi.org/10.18860/cauchy.v11i2.42071

observations from the minority class. In environmental cases, this limitation can lead to serious consequences. For instance, contaminated water can be misclassified as safe water, which can endanger public health and threaten the sustainability of ecosystems [6][7]. Therefore, methods are needed that not only address the imbalance of data, but also maintain the natural relationships between variables in complex environmental datasets [8] [9].

Generative Adversarial Networks (GANs) have recently gained attention as an effective solution to handle imbalanced data [10]. One of their developments, the Conditional Tabular GAN (CTGAN), is specifically designed for tabular datasets containing both numeric and categorical variables. Several previous studies have reported that CTGAN can enhance the representation of minority classes in various domains [11]. Using CTGAN in conjunction with machine learning models has also been shown to improve classification accuracy and sensitivity in medical datasets [12]. Moreover, CTGAN has demonstrated the ability to generate synthetic samples that contribute to a more balanced and stable classification performance [13]. In Smart IoT applications, CTGAN has even been reported to enhance multi-class classification results through higher-quality synthetic data [14]. These findings indicate that CTGAN is highly relevant for environmental datasets as it can preserve complex nonlinear feature relationships.

In addition to data balancing, the performance of classification models is also highly influenced by the algorithms used and the hyperparameter optimization strategies applied [15][16]. Ensemble Learning is one approach that is widely used because it combines multiple learners to produce predictions that are generally more accurate and stable[17]. This method is particularly useful for complex datasets as it enhances generalization ability while reducing the risk of overfitting. To further optimize model performance, hyperparameter tuning is necessary, and Randomized Search is considered an efficient method because it can explore parameter combinations more practically compared to exhaustive search techniques [18]. Additionally, the integration of CTGAN, Ensemble Learning, and Randomized Search for environmental classification tasks, particularly for clean water quality datasets, remains relatively limited.

This research aims to develop a classification framework that integrates CTGAN with Ensemble Learning based on Randomized Search to enhance performance on clean water quality data with imbalanced class distribution. CTGAN is used to generate synthetic samples for the minority class while preserving the original structure of the environmental dataset. Meanwhile, Ensemble Learning is applied as a classification method, and Randomized Search is employed to obtain more effective hyperparameter tuning configurations. The integration of these techniques is expected to improve classification performance, particularly in terms of accuracy, recall, sensitivity, and model stability, while producing a more accurate system to support environmental monitoring and decision-making.

2. Methods

This Research involves several stages. The research was conducted using Python Programming on Google Collaboratory and utilized the Water Pollution & Disease dataset. The first step was labeling clean water data according to standards and categorizing it into two distinct labels by WHO Table 1. The second step involved splitting the data with 25 trials split randomly, so that the training and testing datasets resulted in 25 different compositions, while maintaining a consistent split ratio of 80% training data and 20% testing data. The third step, all 25 datasets were tested using several model combinations, including RSCV-Ensemble and the CTGAN oversampling technique, and the test results were then statistically validated to determine the significance of the comparison of the best-performing model. After obtaining the best model, it was used as the primary model to classify clean water and contaminated water. The sequence of research steps is presented in Fig. 1 below.

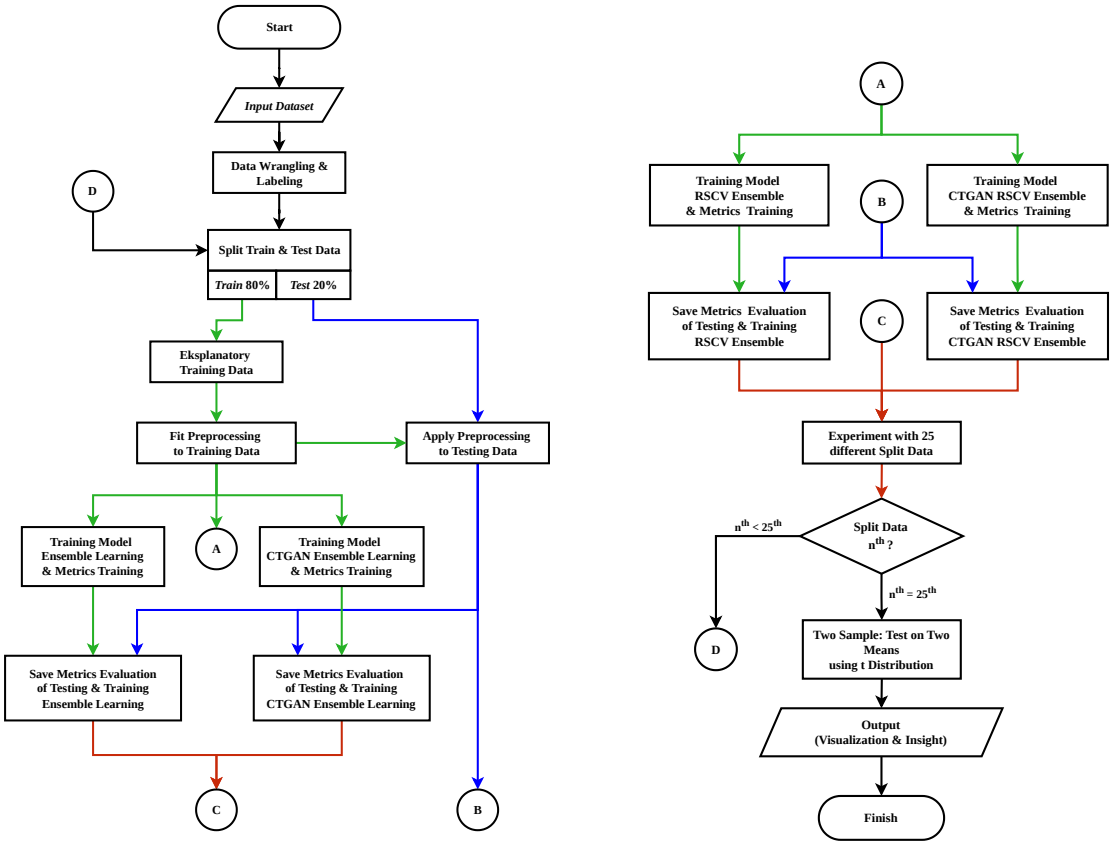


Fig. 1: Research Flow

2.1. Dataset

The research data was obtained from the Kaggle website [19]. The data consists of 3000 samples and 24 features without labels. Labels were derived using the Data Wrangling method from 5 features: pH Level, Nitrate Level (mg/L), Lead Concentration (μg/L), Turbidity (NTU), and Dissolved Oxygen (mg/L). These five features will be combined into a single label, where if they meet the requirements by WHO Table 1, they will be classified as the Clean Water Label, otherwise will be classified as the Polluted Water Label.

Table 1: Standard Quality of Drinking Water [20]

Parameters	Standard Quality
pH Level	6.5-8.5
Nitrate Level (mg/L)	≤ 50
Lead Concentration (μg/L)	≤ 10
Turbidity (NTU)	≤ 5
Dissolved Oxygen (mg/L)	≥ 5

The features used to construct the label were eliminated, and the target labeling process is shows Fig. 2. Features that were not used to create the label were retained as input features for clean water classification, thereby preventing data leakage during the classification process. After labeling the dataset, we obtain two classes of labeled that classified as Clean Water and Contaminated Water. However, those two class weren't at the same amount of samples. Fig. 3 show the comparison between minority and majority class, this imbalanced class can caused bias in classification model.

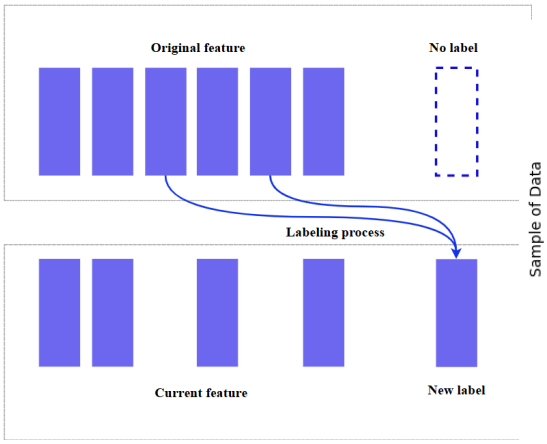


Fig. 2: Labeling process

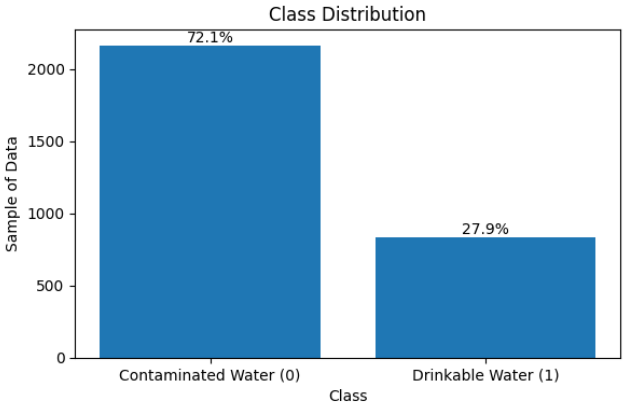


Fig. 3: Class Distribution

2.2. Pre-Processing Data

Before oversampling, a correlation analysis of the features used was conducted. Fig. 4 shows that there is no significant correlation between the features, indicating that the features are relatively independent of each other. The lack of significant correlation suggests that multicollinearity is not an issue for the model, allowing the machine learning algorithm to effectively learn the individual effects of each feature.



Fig. 4: Feature Correlation

After performing the correlation analysis, handling of missing, duplicate, and imbalanced data was also carried out. The focus of this study is on handling imbalanced data using CTGAN, where oversampling is performed by generating synthetic data that can handle mixed data for Clean Water Classification Problem.

2.3. CTGAN

Generative Adversarial Networks (GAN) offer significant advantages compared to oversampling methods such as SMOTE (Synthetic Minority Over-Sampling Technique) or ROS (Random Oversampling) in addressing the problem of imbalanced classes [21]. SMOTE and ROS only generate synthetic data by interpolating between existing samples, while GAN allows for the generation of more realistic and varied data. This is due to the adversarial training process, in which the Generator creates synthetic data attempt to mimic real data, while the Discriminator tries to distinguish between real and synthetic data [22].

CTGAN can overcome the limitations of GAN, where GAN tends to struggle when used for tabular data with categorical variables. CTGAN is capable of generating synthetic data based on specific conditions or labels, allowing for further control over the generated data. In CTGAN, the Generator receives two random noise inputs, z and condition y, as shown in Eqs. (1), which enables the generation of data that aligns with certain categories or attributes [23].

$$x_{generated} = G(z, y) \tag{1}$$

where:

z	:	Random noise sampled from the latent distribution,
y	:	Condition or class label associated with the generated sample,
$G(z, y)$	:	Generator network,
$x_{generated}$	:	Synthetic data generated by the generator.

Thus, new synthetic data can be generated, as illustrated in Fig. 5. The Generator and Discriminator are trained adversarially, where the Generator is optimized using the loss function defined in Eq. (2).

$$\mathcal{L}_G = -\mathbb{E}_{y \sim p_y}[\log D(G(z, y), y)] \tag{2}$$

where:

$\mathcal{L}_G$	:	Generator loss function,
$\mathbb{E}$	:	Expectation operator,
$y \sim p_y$	:	Condition vector sampled from the distribution of conditions,
p_y	:	Probability distribution of the condition vector,
z	:	Random noise sampled from the latent distribution
$G(z, y)$	:	Synthetic sample generated by the generator conditioned on y,
$D(G(z, y), y)$	:	Discriminator output representing the probability that the generated sample conditioned on y is real,
log	:	Natural logarithm function,

and the Discriminator Loss function in Eqs. (3), where both models enhance their capabilities. The Generator strives to become better at producing realistic fake data, while the Discriminator aims to become better at distinguishing between real and synthetic data.

$$\mathcal{L}_D = -\mathbb{E}_{(x,y) \sim p_{data}(x,y)}[\log D(x, y)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z, y), y))] \tag{3}$$

where:

$\mathcal{L}_D$	:	Discriminator loss function,
$\mathbb{E}$	:	Expectation,
$p_{data}(x, y)$	:	Distribution of real data and their corresponding labels,
$p_z(z)$	:	Distribution of noise fed to the generator,
$G(z, y)$	:	Generator that produces synthetic data conditioned on label y,
$D(x, y)$	:	Discriminator conditioned on label y that determines whether x is real or synthetic.

The algorithm 1 describes how the generator minimizes the term $\log(1 - D(G(z, y), y))$, while the discriminator maximizes $\log D(x, y)$ for real samples and $\log(1 - D(G(z, y), y))$ for synthetic samples. Thus, CTGAN allows for the generation of synthetic data that is more consistent with the specified labels or conditions.

Algorithm 1: CTGAN for Synthetic Data Oversampling

Input : Training data (X_{train}, y_{train}), minority class label y_m, number of synthetic samples N.
Output : Balanced dataset (X', y').
Initialize Generator G and Discriminator D;
for $e \leftarrow 1$ **to** E **do**
  Shuffle training data (X_{train}, y_{train})
  foreach *mini-batch* $\mathcal{B}$ **do**
    Sample noise vectors z
    Generate synthetic samples $\hat{x} = G(z, y)$
    Compute discriminator loss $\mathcal{L}_D$
    Update discriminator parameters
    Compute generator loss $\mathcal{L}_G$
    Update generator parameters
$X_{synthetic} \leftarrow \emptyset$; $y_{synthetic} \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** N **do**
  Sample noise vector z_i
  Generate synthetic sample $x_{new} = G(z_i, y_m)$
  $X_{synthetic} \leftarrow X_{synthetic} \cup x_{new}$
  $y_{synthetic} \leftarrow y_{synthetic} \cup y_m$
$X' \leftarrow X_{train} \cup X_{synthetic}$; $y' \leftarrow y_{train} \cup y_{synthetic}$; **return** (X', y').

2.4. RSCV-Random Forest Model

Random Forest is an ensemble learning algorithm that builds multiple decision trees for classification or regression. This study uses a Random Forest Classifier because it effectively reduces overfitting compared to individual decision trees [24]. Each tree is trained on a random subset of the data, which helps lower variance and improve accuracy. The method relies on bootstrap aggregating (bagging) and random feature selection, where each tree is built from randomly sampled data and features. Final predictions are produced in Eqs. (4) by aggregating the outputs of all trees through voting.

$$\vec{y}_{RF} = \text{Majority Vote } (\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_B) \tag{4}$$

where:

$\hat{y}_{RF}$	:	Final prediction of the Random Forest model,
$\hat{y}_b$	:	Prediction of the b-th decision tree,
B	:	Total number of decision trees in the Random Forest,
mode$(\cdot)$	:	Majority voting function that selects the most frequent class label.

The objective function of Random Forest is to minimize the total classification errors from all trees in the forest. For classification, the objective function used is Gini Impurity in Eqs. (5). To reduce the entropy value before and after the split, Information Gain is used [25]. Information Gain measures the reduction in uncertainty (entropy) after dividing the data based on a specific feature. In the context of decision trees, Information Gain (IG) measures how well a feature divides the dataset in such a way that it reduces uncertainty or entropy at the node, as shown in the Eqs. (6).

$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2 \tag{5}$$

where:

D : Dataset at a node,
C : Number of classes,
p_i : Proportion of samples belonging to class i in dataset D.

$$IG(D, F) = Entropy(D) - \sum_{v \in Values(F)} \frac{|D_v|}{|D|} Entropy(D_v) \tag{6}$$

where:
D : Dataset at the current node,
F : Feature used for splitting,
$Values(F)$: Set of all possible values of feature F,
D_v : Subset of D where feature $F = v$,
$|D|$: Number of samples in dataset D,
$|D_v|$: Number of samples in subset D_v,
$Entropy(\cdot)$: Entropy function measuring impurity of a dataset.

Algorithm 2: Random Forest with Random Search for Hyperparameter Optimization

Input : Training data $\vec{X}_{train} \in \mathbb{R}^{n \times d}$, target labels $\vec{y}_{train} \in \mathbb{R}^n$, hyperparameter search space Θ, maximum number of iterations N.
Output : Trained Random Forest model $RF = \{Tree_1, Tree_2, \ldots, Tree_T\}$ with optimized hyperparameters.

$RF \leftarrow \emptyset$ `// Initialize empty forest`
for $t \leftarrow 1$ **to** N **do**
  `// Hyperparameter optimization loop`
  `// Random Search: Sample hyperparameters from search space`
  Choose hyperparameters θ_t randomly from Θ
  `// Build Random Forest using selected hyperparameters`
  for $i \leftarrow 1$ **to** T **do**
    `// Bootstrap sampling: select random subset of data`
    Take bootstrap sample $(\vec{X}_{train_t}, \vec{y}_{train_t}) \sim \text{Bootstrap}(\vec{X}_{train}, \vec{y}_{train})$
    `// Select` m `random features for each node split`
    Select random subset of m features from total d features
    `// Build Decision Tree with selected data and features`
    Build Decision Tree with $(\vec{X}_{train_t}, \vec{y}_{train_t})$ and selected features by calculating:

$$IG(D, F) = \text{Entropy}(D) - \sum_{v \in \text{Values}(F)} \frac{|D_v|}{|D|} \cdot \text{Entropy}(D_v)$$

    `// Add Decision Tree to forest`
    $RF \leftarrow RF \cup \{Tree_t\}$
  `// Evaluate the performance of Random Forest with` θ_t
  Evaluate RF with hyperparameters θ_t: $y_t \leftarrow f(\theta_t)$
  `// Update hyperparameter performance dataset`
  $D_t \leftarrow D_{t-1} \cup \{(\theta_t, y_t)\}$
Output : Trained Random Forest model $RF = \{Tree_1, Tree_2, \ldots, Tree_T\}$ with optimized hyperparameters θ^*.

Randomized Search is a technique for hyperparameter optimization in machine learning models aimed at saving time and finding the optimal combination of hyperparameters when the hyperparameter search space is vast and model evaluation requires high computational costs [8] [26]. Let $\theta_1, \theta_2, \ldots, \theta_n$ be the hyperparameters used by the model, then there exists a search

space S_i for each hyperparameter θ_i. Randomized Search selects random values $\theta_(i)^*$ for each hyperparameter θ_i from the predefined search space S_i, as shown in Eqs. (7)

$$\theta_i^* \sim U(S_i), \quad i = 1, 2, \ldots, n \tag{7}$$

where:

θ_i^*	:	Sampled value of the i-th hyperparameter
$U(S_i)$	:	Uniform distribution over search space S_i
S_i	:	Search space (range of values) for hyperparameter θ_i
n	:	Number of hyperparameters

Where $U[S_i]$ represents the random distribution taken from the search space S_i for each hyperparameter. Random Search is used to find the best hyperparameters for Random Forest. Therefore, algorithm 2 shows the algorithm that demonstrates how the hybrid learning works.

2.5. Proposed Method

This article uses an Oversampling approach to address the issue of class imbalance. The proposed oversampling method is the CTGAN method Fig. 5, which enables the resampling of synthetic data that is more structured and relevant to the desired conditions or categories. The Ensemble Learning models used as classification models are Random Forest, which are combined with Randomized Search CV Optimization for hyperparameter tuning to optimize the classification model.

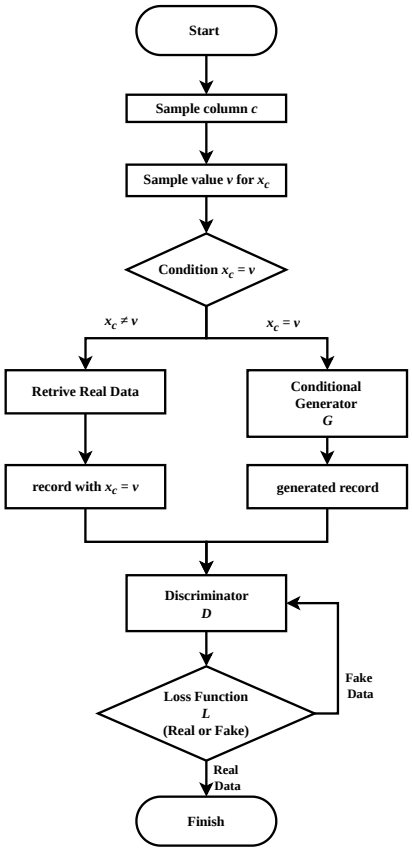


Fig. 5: CTGAN Algorithm

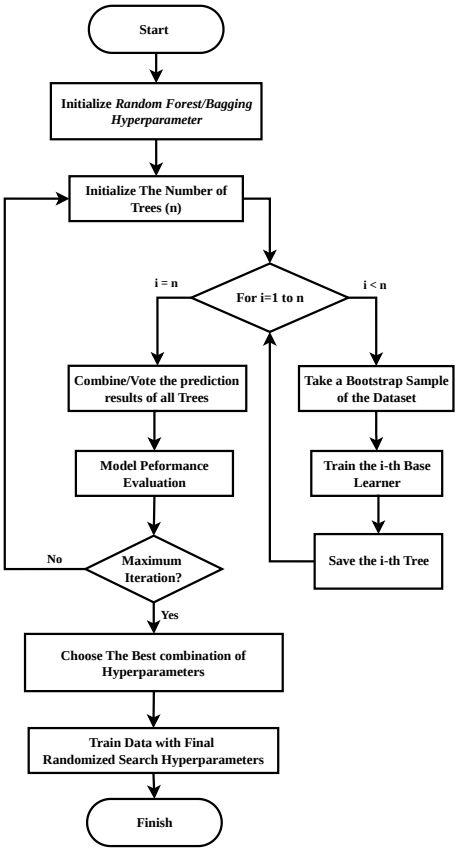


Fig. 6: RSCV-Ensemble Algorithm

This research uses Ensemble Learning models, Random Forest, as classification models, combined with Randomized Search to optimize and determine the optimal hyperparameter combinations for training the model. Fig. 6 shows that for Random Forest, combinations of the

hyperparameters will be sought based on the evaluation results using the AUC metric and the Classification Report.

2.6. Experimental Validation

The t-statistic (t_{stat}) is used to assess whether the mean difference between two paired observations is statistically significant [27]. In the context of this study, the t_{stat} is applied to compare model performance before and after the implementation of a specific resampling technique or to compare model performance before and after the implementation of a specific optimization techniques . Based on [28] where d_i is $x_i - y_i$, where $i = 1, 2, \ldots, n$, while x_i is observation i^{th} from first model, and y_i is observation i^{th} from second model, then the t_{stat} is calculated as Eqs (8),

$$t_{stat} = \frac{\bar{d} - d_0}{s_d \sqrt{n}} \tag{8}$$

where:

$\bar{d}$	:	Mean of d_i,
d_0	:	Constant,
s_d	:	Standard deviation of d_i,
n	:	Number of paired samples.

The calculation of two-side paired t_{stat} with significance level ($\alpha = 0.05$) is when $t_{stat} > |t_{(\alpha/2),v}|$, $v = n - 1$. Then, the null hypothesis H_0 is rejected, meaning that between two model there is a significant different result like in Eqs. (10). Otherwise the difference is considered not statistically significant like in Eqs. (9).

$$H_0 : \mu_D = d_0 \text{ (There is statistically no significant different)} \tag{9}$$

$$H_1 : \mu_D \neq d_0 \text{ (There is statistically significant different)} \tag{10}$$

where μ_D is Mean of paired observation, and D is $X - Y$.

3. Results and Discussion

This section presents the experimental results and discusses the effectiveness of the proposed approach for clean water quality classification under imbalanced class distributions. We begin by examining the application of oversampling techniques, namely SMOTENC and CTGAN, to address class imbalance in mixed data. Next, we evaluate the contribution of Randomized Search Cross Validation (RSCV) in optimizing the Random Forest model through hyperparameter tuning and performance comparison. Finally, statistical tests are performed to determine whether the observed differences in model performance are statistically significant.

3.1. Oversampling

Before developing the classification models, the class imbalance problem must first be addressed because the clean water class contains substantially fewer observations than the polluted water class. This imbalance may bias the learning process toward the majority class and reduce the model's ability to correctly identify clean water observations.

To mitigate this issue, two oversampling techniques, SMOTENC and CTGAN, were employed because both methods are designed to generate synthetic samples for mixed datasets consisting of numerical and categorical features. The CTGAN model was implemented using the parameter configuration presented in Table 2, while the effects of both oversampling methods on class distribution are discussed in the following analysis.

The oversampling results obtained using SMOTENC and CTGAN indicate that both methods are effective in balancing the class distribution. As shown in Fig. 7, the main difference between the two approaches lies in the mechanism used to generate synthetic minority samples. SMOTENC

Table 2: CTGAN Parameter Configuration [17]

Parameters	Value
Epochs	200
Batch Size	128
PAC	1
Embedding Dimension	64
Generator Dimension	128
Discriminator Dimension	128
Verbose	False

creates new observations by interpolating minority class instances while preserving categorical features. In contrast, CTGAN employs a generative adversarial framework to learn the joint distribution of numerical and categorical variables and subsequently generates synthetic samples that resemble the original data distribution.

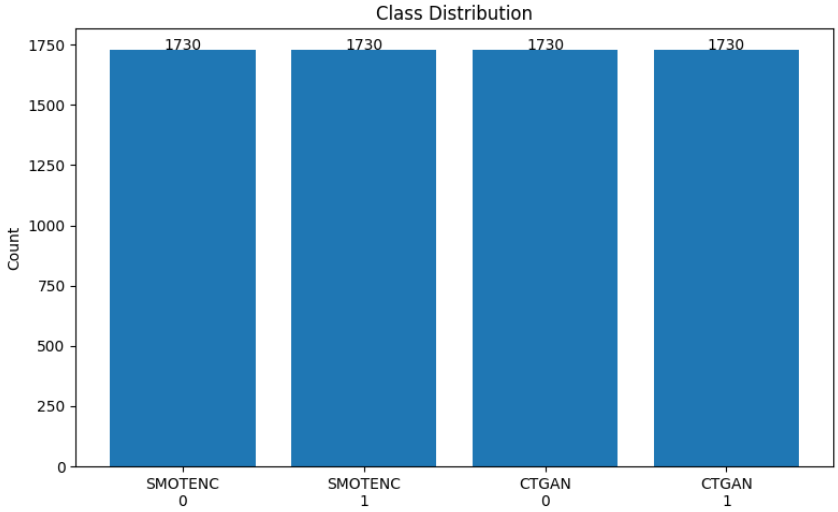


Fig. 7: Class Distribution of Oversampling

In the CTGAN framework, the generator produces synthetic observations, whereas the discriminator attempts to distinguish between real and synthetic data. Through this adversarial learning process, the generator progressively improves its ability to produce synthetic samples that closely represent the characteristics of the original minority class.

3.2. RSCV Random Forest

Randomized Search Cross Validation (RSCV) was applied to the Random Forest model to evaluate different hyperparameter configurations. The hyperparameters considered in this study include n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, bootstrap, max_leaf_nodes, and class_weight. The default values and the hyperparameter search space used during the RSCV process are showed in Table 3.

The comparison between the basic ensemble model and the model optimized using RSCV shows a significant improvement across all evaluation metrics. The application of RSCV after handling class imbalance further enhances the model's performance in clean water quality classification. Fig. 8 illustrate the performance difference between the model without RSCV Optimization and with RSCV Optimization. The comparison shows improvement at each oversampling handling stage, indicating that RSCV optimization can enhance the results of oversampling models in class imbalance cases.

The comparison between SMOTENC, CTGAN, and the models with and without RSCV optimization shows improvements through model evaluation indicators. RSCV is capable of

Table 3: Random Forest Hyperparameter Configuration

Parameters	Default RF Value	RSCV RF Search Space
n_estimators	100	[100, 200, 300, 500]
max_depth	None	[10, 20, 30, None]
min_samples_split	2	[2, 5, 10]
min_samples_leaf	1	[1, 2, 4]
max_features	sqrt	[sqrt, log2]
bootstrap	True	[True, False]
max_leaf_nodes	None	[10, 20, 50, None]
class_weight	None	[None, balanced]

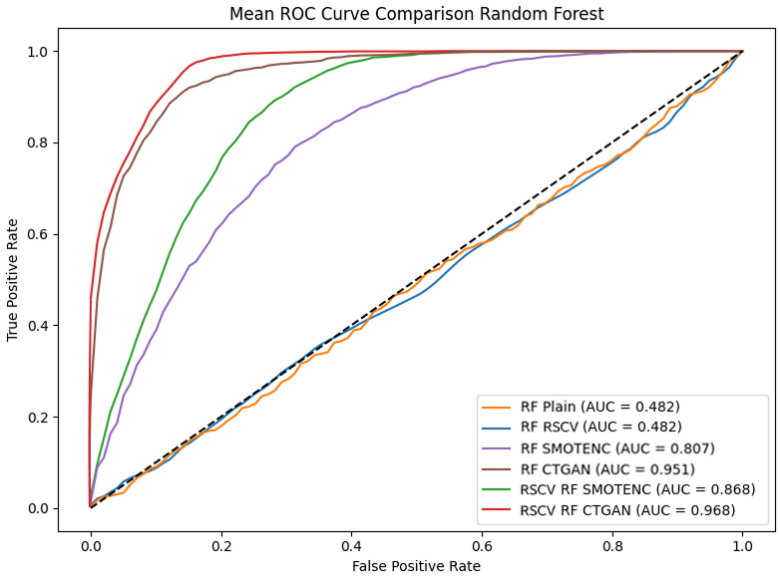


Fig. 8: AUC ROC of Random Forest

optimizing ensemble learning models like Random Forest. This improvement significantly enhances classification performance.

Table 4: Comparison of Random Forest Model Performance

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
RF	0.7171 ± 0.0013	0.4805 ± 0.1225	0.4990 ± 0.0018	0.4243 ± 0.0039
RF SMOTENC	0.7619 ± 0.0013	0.6154 ± 0.1225	0.6398 ± 0.0018	0.7303 ± 0.0039
RF CTGAN	0.9018 ± 0.0130	0.8920 ± 0.0387	0.9053 ± 0.0144	0.9031 ± 0.0346
RSCV RF	0.7213 ± 0.0550	0.4029 ± 0.0105	0.5001 ± 0.0109	0.4202 ± 0.0126
RSCV RF SMOTENC	0.7854 ± 0.0212	0.7139 ± 0.0213	0.7193 ± 0.0212	0.7688 ± 0.0212
RSCV RF CTGAN	**0.9281 ± 0.0188**	**0.9023 ± 0.0183**	**0.9098 ± 0.0188**	**0.9082 ± 0.0189**

RSCV optimization, as shown in Table 4, indicates that almost all evaluation metrics across all Random Forest models improved after being optimized with RSCV. The accuracy of the RF_CTGAN model increased from 0.9018 to 0.9281 after RSCV optimization. The CTGAN model itself significantly improved model performance across all models compared to the handling with SMOTENC. The accuracy value obtained from the RF_SMOTENC model increased from 0.7619 to 0.9082 in the RF_CTGAN model. Similarly, with the use of RSCV, CTGAN also enhances model performance, making it even better.

3.3. Evaluation

To validate the performance differences between models, a paired t-test was conducted on the accuracy scores obtained from 25 repeated experiments. The test was performed at a significance level of $\alpha = 0.05$ with 24 degrees of freedom. Since a two-tailed test was employed, the value of t-critical was ± 2.063899 ($t_{0.025,24}$). The null hypothesis was rejected when the absolute value of the calculated t_{stat} exceeded the t-critical, indicating that the observed performance differences were statistically significant. $H_0 : \mu_D = 0$ $H_1 : \mu_D \neq 0$ The null hypothesis H_0 was rejected when $|t_{\text{stat}}| > 2.063899$. Otherwise, there was insufficient evidence to reject H_0.

Table 5: Significance of Accuracy's Model

Model 1	Model 2	t_{stat}	p-value	Decision	Best μ Model
RF	RF_SMOTENC	-57.3911	3.3203e-29	Reject H_0	Model 2
RF	RF_CTGAN	-94.1087	2.4498e-33	Reject H_0	Model 2
RF	RSCV_ RF_SMOTENC	-61.8268	5.6263e-30	Reject H_0	Model 2
RF	RSCV_ RF_CTGAN	-132.8770	6.3130e-38	Reject H_0	Model 2
RSCV_ RF	RF_SMOTENC	-77.7176	2.3851e-32	Reject H_0	Model 2
RSCV_ RF	RF_CTGAN	-110.6325	5.0914e-36	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_SMOTENC	-121.3037	5.6066e-37	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_CTGAN	-197.5249	4.6972e-42	Reject H_0	Model 2
RF_SMOTENC	RF_CTGAN	-41.8161	6.1557e-26	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_SMOTENC	-2.0525	5.1178e-04	Accept H_0	Both Model
RF_SMOTENC	RSCV_ RF_CTGAN	-35.6649	2.6447e-24	Reject H_0	Model 2
RF_CTGAN	RSCV_ RF_SMOTENC	20.2056	1.4198e-18	Reject H_0	Model 1
RF_CTGAN	RSCV_ RF_CTGAN	-6.5265	9.4734e-09	Reject H_0	Model 2
RSCV_ RF_SMOTENC	RSCV_ RF_CTGAN	-33.8791	8.8667e-24	Reject H_0	Model 2

Table 6: Significance of Recall's Model

Model 1	Model 2	t_{stat}	p-value	Decision	Best μ Model
RF	RSCV_ RF	2.3619	2.6632e-02	Reject H_0	Model 1
RF	RF_SMOTENC	-51.7796	3.8492e-26	Reject H_0	Model 2
RF	RF_CTGAN	-95.7777	1.6084e-32	Reject H_0	Model 2
RF	RSCV_ RF_SMOTENC	-71.5308	1.7320e-29	Reject H_0	Model 2
RF	RSCV_ RF_CTGAN	-148.8205	4.1735e-37	Reject H_0	Model 2
RSCV_ RF	RF_SMOTENC	-59.1118	1.6419e-27	Reject H_0	Model 2
RSCV_ RF	RF_CTGAN	-97.5301	1.0420e-32	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_SMOTENC	-105.5987	1.5533e-33	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_CTGAN	-187.6016	1.6171e-39	Reject H_0	Model 2
RF_SMOTENC	RF_CTGAN	-42.2342	4.8630e-24	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_SMOTENC	-6.5686	8.5649e-07	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_CTGAN	-35.6799	2.6185e-22	Reject H_0	Model 2
RF_CTGAN	RSCV_ RF_SMOTENC	20.3049	1.2701e-16	Reject H_0	Model 1
RF_CTGAN	RSCV_ RF_CTGAN	-2.0671	4.9667e-02	Reject H_0	Model 2
RSCV_ RF_SMOTENC	RSCV_ RF_CTGAN	-33.9371	8.5172e-22	Reject H_0	Model 2

3.4. Discussion

The results presented in Table 5, Table 7, Table 8 show that almost all model comparisons result in statistically significant differences. Specifically, the models using CTGAN, both with and without RSCV, show significant improvements compared to the baseline and SMOTENC-based models. The only non-significant comparison occurred between RF_SMOTENC and RSCV_ RF_SMOTENC, suggesting that optimization alone does not substantially improve performance when SMOTENC is applied. These findings confirm that the observed performance improvements in the previous section are supported by statistical evidence, rather than random fluctuations.

Addressing class imbalance also requires evaluating other performance metrics Beyond overall accuracy, precision, and recall, such as Recall, which reflects the model's effectiveness in correctly

Table 7: Significance of Precision's Model

Model 1	Model 2	t_{stat}	p-value	Decision	Best μ Model
RF	RSCV_ RF	-1.0060	3.2442e-01	Accept H_0	Both Model
RF	RF_SMOTENC	-54.1709	1.3141e-27	Reject H_0	Model 2
RF	RF_CTGAN	-107.3149	1.0557e-33	Reject H_0	Model 2
RF	RSCV_ RF_SMOTENC	-155.4175	1.4752e-37	Reject H_0	Model 2
RF	RSCV_ RF_CTGAN	-70.7828	2.2265e-29	Reject H_0	Model 2
RSCV_ RF	RF_SMOTENC	-10.8526	9.7147e-11	Reject H_0	Model 2
RSCV_ RF	RF_CTGAN	-15.6185	4.4893e-14	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_SMOTENC	-16.5229	1.2986e-14	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_CTGAN	-12.5721	4.7371e-12	Reject H_0	Model 2
RF_SMOTENC	RF_CTGAN	-44.5652	1.3606e-24	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_SMOTENC	-35.9216	2.2333e-22	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_CTGAN	-6.4691	1.0875e-06	Reject H_0	Model 2
RF_CTGAN	RSCV_ RF_SMOTENC	1.8453	7.7359e-02	Accept H_0	Both Model
RF_CTGAN	RSCV_ RF_CTGAN	-21.7073	2.7691e-17	Reject H_0	Model 2
RSCV_ RF_SMOTENC	RSCV_ RF_CTGAN	-34.3878	6.2438e-22	Reject H_0	Model 2

Table 8: Significance of F1-Score's Model

Model 1	Model 2	t_{stat}	p-value	Decision	Best μ Model
RF	RSCV_ RF	22.9510	7.7140e-18	Reject H_0	Model 1
RF	RF_SMOTENC	-57.3911	3.3203e-27	Reject H_0	Model 2
RF	RF_CTGAN	-94.1087	2.4499e-32	Reject H_0	Model 2
RF	RSCV_ RF_SMOTENC	-132.8771	6.3130e-36	Reject H_0	Model 2
RF	RSCV_ RF_CTGAN	-61.8268	5.6263e-28	Reject H_0	Model 2
RSCV_ RF	RF_SMOTENC	-77.7176	2.3851e-30	Reject H_0	Model 2
RSCV_ RF	RF_CTGAN	-110.6325	5.0914e-34	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_SMOTENC	-197.5249	4.6972e-40	Reject H_0	Model 2
RSCV_ RF	RSCV_ RF_CTGAN	-121.3037	5.6066e-35	Reject H_0	Model 2
RF_SMOTENC	RF_CTGAN	-41.8161	6.1557e-24	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_SMOTENC	-35.6649	2.644e-22	Reject H_0	Model 2
RF_SMOTENC	RSCV_ RF_CTGAN	-6.5265	9.4734e-07	Reject H_0	Model 2
RF_CTGAN	RSCV_ RF_SMOTENC	2.0525	5.1178e-02	Accept H_0	Both Model
RF_CTGAN	RSCV_ RF_CTGAN	-20.2056	1.4198e-16	Reject H_0	Model 2
RSCV_ RF_SMOTENC	RSCV_ RF_CTGAN	-33.8791	8.8667e-22	Reject H_0	Model 2

identifying positive instances in imbalanced datasets. As shown in Table 6, the majority of model comparisons indicate statistically significant differences. Using CTGAN whether combined with RSCV or not, clear gains in Recall over both the baseline and SMOTENC based model approaches. The only exception is the comparison between RF_SMOTENC and RSCV_RF_SMOTENC, indicating that hyperparameter optimization alone has limited impact when SMOTENC is already applied. These results support that the observed improvements in Recall are robust and not due to random chance.

The best classification model in Table 4 shows that CTGAN combined with the RSCV ensemble can effectively handle imbalanced data for clean water classification. This model was subsequently used to analyze feature importance and identify factors associated with clean water quality classification. As shown in Fig. 9, Rainfall emerged as the most influential feature, with an importance value exceeding 0.08. From an environmental perspective, this result is plausible because rainfall can affect water quality through surface runoff, soil erosion, and the transport of pollutants into water sources. Variations in rainfall may therefore influence several water quality indicators and contribute to differences between clean and non-clean water conditions. In addition to Rainfall, Access_to_Clean_Water and Water_Source_Type were identified as important variables, suggesting a strong relationship between water accessibility, source characteristics, and the classification outcome.

Although these findings provide useful insights into factors associated with clean water quality classification, they should be interpreted with caution. The feature importance values were

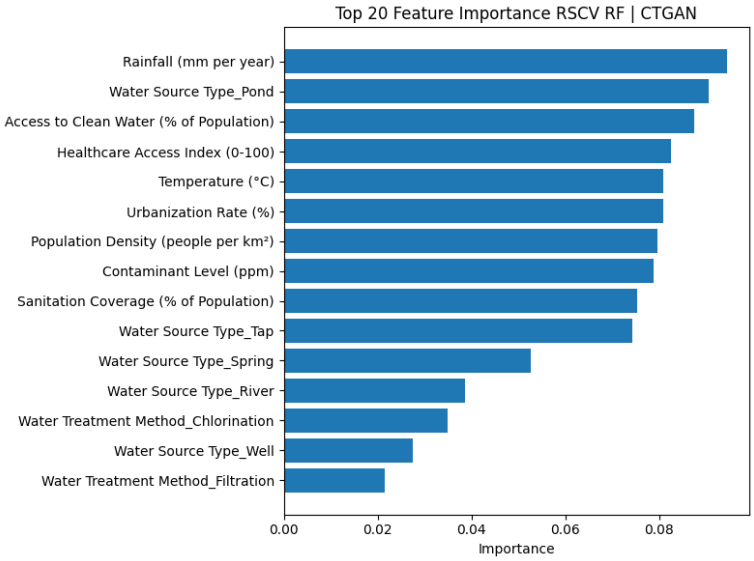


Fig. 9: Feature importance of Clean Water Classification

obtained from a model trained using CTGAN samples, and the data generation process may influence the relative importance of certain variables. Consequently, the prominence of Rainfall and other variables may partially reflect patterns learned from the synthetic data distribution. Further validation using independent real-world datasets is necessary to confirm whether these feature importance rankings remain consistent across different data sources. These results suggest that the combination of CTGAN and Random Search Optimization may improve classification performance while providing interpretable information regarding variables associated with clean water quality. However, additional studies are required to assess the generalizability of these findings and to investigate potential biases introduced during synthetic data generation.

4. Conclusion

The results show that CTGAN was associated with higher model performance compared to SMOTENC and models without oversampling on the dataset used in this study. CTGAN consistently achieved high evaluation scores across multiple performance metrics. RSCV optimization further improved model performance, particularly when combined with CTGAN, resulting in the highest observed performance with an accuracy of 0.92821, precision of 0.9023, recall of 0.9098, f1-score of 0.9082, and an AUC score of 0.968. Based on these results, the RSCV_RF_CTGAN model was selected as the best-performing model in this study and was subsequently used for classifying clean water quality datasets.

From Table 5, Table 6, Table 7, and Table 8, the statistical test results reinforce the reliability of the observed performance differences across all models. Most paired comparisons show clear differences in accuracy, particularly for models that implement CTGAN, both with and without Random Search Optimization, when compared to baseline-based and SMOTENC-based approaches. The only comparison that did not show a significant difference appeared between RF_SMOTENC and RSCV_RF_SMOTENC, suggesting that parameter optimization alone may provide limited additional benefit when combined with the SMOTENC-based oversampling approach.

The results also suggest that CTGAN-generated synthetic samples were able to preserve important characteristics of the training data, which may have contributed to the improved classification performance. Nevertheless, the possibility of synthetic data bias and overfitting cannot be completely ruled out. Since CTGAN generates new samples by learning the underlying data distribution, some generated patterns may not fully represent real-world conditions. Therefore, the performance gains observed in this study should be interpreted with caution and validated

using external datasets.

This study has several limitations. First, the class imbalance ratio in the dataset is relatively moderate, and the effectiveness of CTGAN under more severe imbalance conditions remains unclear. Second, the use of synthetic data introduces potential overfitting risks, particularly when the generated samples closely resemble the training data. Third, model performance depends on the selected hyperparameter configuration for both CTGAN and Random Forest. Finally, CTGAN requires substantially greater computational resources and training time than SMOTENC, which may limit its applicability in resource-constrained environments.

Future research could investigate techniques for reducing overfitting in CTGAN-generated datasets and evaluate the proposed approach using different ensemble learning models, such as XGBoost, LightGBM, AdaBoost, or CatBoost. In addition, further experiments could be conducted using more extreme class imbalance scenarios, particularly with minority-to-majority class ratios exceeding 1:9. Future studies may also compare CTGAN with other generative oversampling approaches and validate the proposed framework using independent datasets from different geographical regions.

CRediT Authorship Contribution Statement

Favian Sis Bagus Febrianto: Conceptualization, Methodology, Writing–Original Draft. **Umu Sa'adah:** Data Curation, Formal Analysis, Writing–Review, Editing & Supervision. **Imam Nurhadi Purwanto:** Data Curation, Validation, Co-Supervision.

Declaration of Generative AI and AI-assisted technologies

In preparing this manuscript, Generative AI (ChatGPT by OpenAI) was used to aid inrefining grammar, clarity, and overall flow, in addition to providing paraphrasing and language polishing support.

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

This research received no external funding. The authors would like to extend their heartfelt appreciations to kaggle.com for access the dataset.

Data and Code Availability

The dataset used in this research is the Water Pollution & Disease of 3000 Observation. The dataset obtained from kaggle repository[1]. The dataset is the open access dataset. Further details regarding the data are discussed in Subsection 2.1.

References

[1] A. Ali, S. M. H. Shamsuddin, and A. L. Ralescu. "Classification with class imbalance problem: A review". In: *Soft Computing Models in Industrial and Environmental Applications*. (2015). https://api.semanticscholar.org/CorpusID:26644563.

[2] Y. Wu. "Imbalanced prediction in epidemiological study: A machine learning-based analysis". In: *Annals of Epidemiology* 109 (2025), pp. 83–92. DOI: https://doi.org/10.1016/j.annepidem.2025.07.023.

[1] https://www.kaggle.com/datasets/khushikyad001/water-pollution-and-disease/data

[3] W. Chen, D. Xu, B. Pan, Y. Zhao, and Y. Song. "Machine Learning-Based Water Quality Classification Assessment". In: *Water* 16.20 (2024), p. 2951. DOI: https://doi.org/10.3390/w16202951.

[4] N. Nasir, A. Kansal, O. Alshaltone, F. Barneih, M. Sameer, A. Shanableh, and A. Al-Shamma'a. "Water quality classification using machine learning algorithms". In: *Journal of Water Process Engineering* 48 (2022), p. 102920. DOI: https://doi.org/10.1016/j.jwpe.2022.102920.

[5] D. Yu and M. Wang. "Harnessing the hybrid machine learning methods for stroke risk classification". In: *Computer Methods in Biomechanics and Biomedical Engineering* (2025), pp. 1–18. DOI: https://doi.org/10.1080/10255842.2025.2501636.

[6] J. Zhai, J. Qi, and C. Shen. "Binary imbalanced data classification based on diversity oversampling by generative models". In: *Information Sciences* 585 (2022), pp. 313–343. DOI: 10.1016/j.ins.2021.11.058.

[7] J. Chen, X. Zhou, J. Yao, and S. Tang. "Application of machine learning in higher education to predict students' performance, learning engagement and self-efficacy: a systematic literature review". In: *Asian Education and Development Studies* 14.2 (2025), pp. 205–240. DOI: 10.1108/AEDS-08-2024-0166.

[8] Adi Fajri Firmansyah, Basuki Rahmat, and Muhammad Muharrom Al Haromainy. "Optimization of the Random Forest Algorithm Using Random Search for Potable Water Quality Classification". In: *Journal of Artificial Intelligence and Engineering Applications* 5.1 (2025). DOI: 10.59934/jaiea.v5i1.1221. https://ioinformatic.org/index.php/JAIEA/article/view/1221.

[9] H. Woldesellasse and S. Tesfamariam. "Prediction of Lateral Spreading Displacement Using Conditional Generative Adversarial Network (cGAN)". In: *Soil Dynamics and Earthquake Engineering* 156 (2022), p. 107214. DOI: https://doi.org/10.1016/j.soildyn.2022.107214.

[10] P. Gogoi and J. A. Valan. "Enhancing date fruit classification using machine learning, CTGAN, and SHAP-based explainability". In: *Food Measure* 19 (2025), pp. 6851–6872. DOI: https://doi.org/10.1007/s11694-025-03428-x.

[11] A. Alzahrani. "Early detection of lung cancer using predictive modeling incorporating CTGAN features and Tree-Based learning". In: *IEEE Access* 13 (2025), pp. 34321–34333. DOI: https://doi.org/10.1109/ACCESS.2025.3543215.

[12] R. Shafique, A. S. Al-Shamayleh, S. K. Posa, A. Ishaq, F. Rustam, and G. S. Choi. "Advancing ovarian cancer outcomes with CTGAN-enhanced hybrid machine learning approach". In: *Knowledge-Based Systems* 328 (2025), p. 114206. DOI: https://doi.org/10.1016/j.knosys.2025.114206.

[13] A. Alabdulwahab. "Enhancing deep learning-based side-channel analysis using feature engineering in a fully simulated IoT system". In: *Expert Systems with Applications* 266 (2025), p. 126079. DOI: https://doi.org/10.1016/j.eswa.2024.126079.

[14] Y. Chen, W. Pedrycz, C. Zhang, J. Wang, and J. Yang. "Oversampling with GAN via Meta-learning for imbalanced data". In: *IEEE Transactions on Multimedia* (2025), pp. 1–16. DOI: https://doi.org/10.1109/TMM.2025.3607712.

[15] S. Mukherjee. "SMOTE-ENN resampling technique with bayesian optimization for multi-class classification of dry bean varieties". In: *Applied Soft Computing* 181 (2025), p. 113467. DOI: https://doi.org/10.1016/j.asoc.2025.113467.

[16] D. R. Jones, M. Schonlau, and W. J. Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492. DOI: https://doi.org/10.1023/A:1008306431147.

[17] E. Zhang, F. Zhou, H. Xi, X. Duan, and J. Liu. "Predicting cycle-to-cycle variations in liquid methane engines using CTGAN-augmented machine learning". In: *Journal of Marine Science and Engineering* 13 (2025), p. 1513. DOI: https://doi.org/10.3390/jmse13081513.

[18] E. Hong and J.-S. Yi. "Sequence image layout generation for construction accident simulation using domain-tuned NER by ZSL-PLM dan scene graph learning". In: *Advanced Engineering Informatics* 68 (2025), p. 103673. DOI: https://doi.org/10.1016/j.aei.2025.103673.

[19] Kaggle. *Water Pollution and Disease*. Accessed: 2026-06-09. (2024). https://www.kaggle.com/datasets/khushikyad001/water-pollution-and-disease.

[20] World Health Organization (WHO). *Guidelines for Drinking-water Quality, 4th edition*. Diakses 15 September 2025. 2011. https://www.who.int/publications/i/item/9789240045064.

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial networks". In: *arXiv preprint* (2014). https://arxiv.org/abs/1406.2661.

[22] I. N. M. Adiputra, P.-C. Lin, and P. Wanchai. "The effectiveness of generative adversarial network-based oversampling methods for imbalanced multi-class credit score classification". In: *Electronics* 14.4 (2025), p. 697. DOI: https://doi.org/10.3390/electronics14040697.

[23] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. "Modeling Tabular Data Using Conditional GAN". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019. https://arxiv.org/abs/1907.00503.

[24] L. Breiman. "Random forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

[25] A. Bhattacharyya, J. Vaughan, and V. N. Nair. "Behavior of hyper-parameters for selected machine learning algorithms: an empirical investigation". In: *arXiv* (2022). https://arxiv.org/abs/2211.08536.

[26] J. Bergstra and Y. Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.10 (2012), pp. 281–305. https://www.jmlr.org/papers/v13/bergstra12a.html.

[27] A. Field. *Discovering Statistics Using IBM SPSS Statistics*. 5th ed. SAGE Publications, 2020. https://us.sagepub.com/en-us/nam/discovering-statistics-using-ibm-spss-statistics/book258032.

[28] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye. *Probability & Statistics for Engineers & Scientists*. Ninth Edition, Global Edition. Edinburgh Gate, Harlow, Essex CM20 2JE, England: Pearson Education Limited, (2016). https://www.pearsonglobaleditions.com.