Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

Diabetes Mellitus Prediction from Primary Health Care Laboratory Data Using Random Forest, Extreme Gradient Boosting, and Light Gradient Boosting Machine with Resampling and Optuna-Based Hyperparameter Optimization

Umar Zaky*[1], Yuwanis Fazlina Agustia[2]

[1]Informtion System, Universitas Teknologi Yogyakarta, Indonesia
[2]Medical Informatics, Universitas Teknologi Yogyakarta, Indonesia

Email: [1]umarzaky@uty.ac.id

Received : Mar 10, 2026; Revised : May 7, 2026; Accepted : Jul 2, 2026; Published : Jul 9, 2026

Abstract

This study examines the use of machine learning models to classify diabetes mellitus status based on laboratory test data. The dataset consists of 484 laboratory test results with 10 clinical parameters, which were used as the main input for model development. Three algorithms, Random Forest, Extreme Gradient Boosting, and Light Gradient Boosting Machine, were compared by applying several resampling techniques and hyperparameter tuning using Optuna to address class imbalance and improve overall model performance. The results show that each algorithm responded differently to the applied resampling methods and tuning strategies, indicating that model performance is influenced by these approaches. Among the evaluated models, Random Forest combined with Synthetic Minority Oversampling Technique and hyperparameter optimization achieved the best performance, with an accuracy of 72.60% and an area under the receiver operating characteristic curve of 76.74%. This performance indicates a moderate ability to distinguish between diabetes and non-diabetes cases based on the available laboratory parameters. Overall, the findings suggest that machine learning can be considered as a potential tool to support clinical decision making, especially when using structured laboratory data. However, given that the performance is still not optimal, further improvement, validation, and exploration of additional data are necessary before considering its implementation in real clinical settings.

Keywords : *Diabetes Mellitus Prediction, Laboratory test data, Machine learning, Optuna, Resampling Techniques.*

1. INTRODUCTION

Diabetes mellitus is a chronic metabolic disease characterized by elevated blood glucose levels due to impaired insulin production or function. As the primary hormone regulating carbohydrate, fat, and protein metabolism, insulin imbalance can affect various organ systems. Without proper management, this condition can lead to serious complications in vital organs such as the heart, kidneys, eyes, and nerves [1]. Therefore, the management of diabetes mellitus does not only focus on controlling blood glucose, but also requires continuous monitoring, patient education, and consistent lifestyle adjustments [2].

In Indonesia, cases of diabetes mellitus continue to increase in line with changes in people's lifestyles. The habit of consuming foods high in sugar, lack of physical activity, and rising obesity rates contribute to the acceleration of this disease. National data shows that the prevalence of diabetes begins to increase in adulthood and is even higher in the elderly [3]. At the primary health care level, such as at the Mlati II Community Health Center in Sleman, diabetes mellitus is recorded as one of the diseases with a number of cases that continues to increase over time. This increase has a direct impact on the burden of care, where health workers not only focus on therapy but also on monitoring patients'

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

conditions regularly to prevent complications. This situation highlights the need for a more efficient and structured approach to support the management of diabetes patients in the community health center setting.

Advances in machine learning technology have opened up new opportunities for more systematic utilization of health data. Through its ability to recognize patterns in historical data, machine learning can be used to generate more objective predictions about patient conditions. In the context of healthcare services, this approach is beginning to be applied as a data-based disease classification tool, including in supporting the determination of Diabetes Mellitus and Non-Diabetes Mellitus status, so that the decision-making process does not solely rely on manual interpretation [4].

Despite its potential, the application of machine learning to health data also faces a number of challenges. Laboratory data generally consists of various variables with inconsistent scales, requiring a normalization process to ensure that each feature contributes equally to model formation [5]. A number of studies show that normalization plays an important role in improving the stability and accuracy of models in medical datasets [6]. In addition, differences in the amount of data between patients with diabetes mellitus and those without diabetes mellitus often cause bias in the prediction results [7]. To mitigate these issues, resampling techniques are widely used because they can improve class distribution by increasing data representation in minority classes [8].

In algorithm selection, ensemble approaches such as Random Forest, Extreme Gradient Boosting (XGBoost), and Light Gradient Boosting Machine (LightGBM) are widely used due to their ability to capture complex patterns in health data and produce stable classification performance [9]. Random Forest is known for its good resistance to overfitting, while XGBoost and LightGBM are capable of efficient step-by-step learning with a boosting mechanism that adapts to data distribution [10]. However, in order for these algorithms to work optimally, parameter settings are required through a hyperparameter tuning process [11]. Several studies have also reported that automatic parameter optimization can significantly improve model performance compared to using default parameters, both in terms of accuracy and model generalization capabilities [12].

A number of studies show that the performance of diabetes prediction models is greatly influenced by data quality and the preprocessing steps applied. Class balancing techniques have been shown to help improve the model's ability to recognize diabetic patients, who are relatively fewer in number, while numerical feature normalization plays a role in maintaining model stability in medical data with a wide range of values [13]. In addition, parameter optimization is an important step because algorithms with good theoretical performance do not necessarily produce optimal performance without appropriate adjustments. Various studies report that hyperparameter tuning, especially with an automated approach, can improve accuracy and balance precision and recall more efficiently [14],[12]. However, these studies often evaluate these approaches separately or focus on a single model, so the combined effect of resampling techniques and systematic hyperparameter optimization across multiple ensemble models is still not fully explored.

On the other hand, many previous studies still focus on public datasets, so the results obtained do not fully reflect the variation in data in primary health care services. In fact, real patient laboratory test data has more diverse characteristics and is more challenging to analyze. Based on these limitations, there is still a need for studies that evaluate machine learning models using real clinical laboratory data while integrating preprocessing strategies and optimization techniques in a more comprehensive manner. Based on these conditions, this study presents a comparative study of three machine learning algorithms, namely Random Forest, XGBoost, and LightGBM, in classifying Diabetes Mellitus and Non-Diabetes Mellitus. The modeling process was carried out by applying normalization techniques, handling class imbalance through various resampling methods, and hyperparameter optimization based on Optuna, thereby obtaining a more objective and relevant comparison of model performance for real-

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

world application contexts. The contributions of this study include the use of real laboratory data from primary healthcare, the evaluation of multiple resampling techniques across different ensemble models, and the analysis of hyperparameter optimization using Optuna in a clinical dataset context.

2. METHOD

The method used for the testing flow consists of six stages, starting from data cleaning, data split, normalization, resampling, hyperparameter optimization, model training, and model evaluation. The testing flow stages are illustrated in Figure 1 as follows.

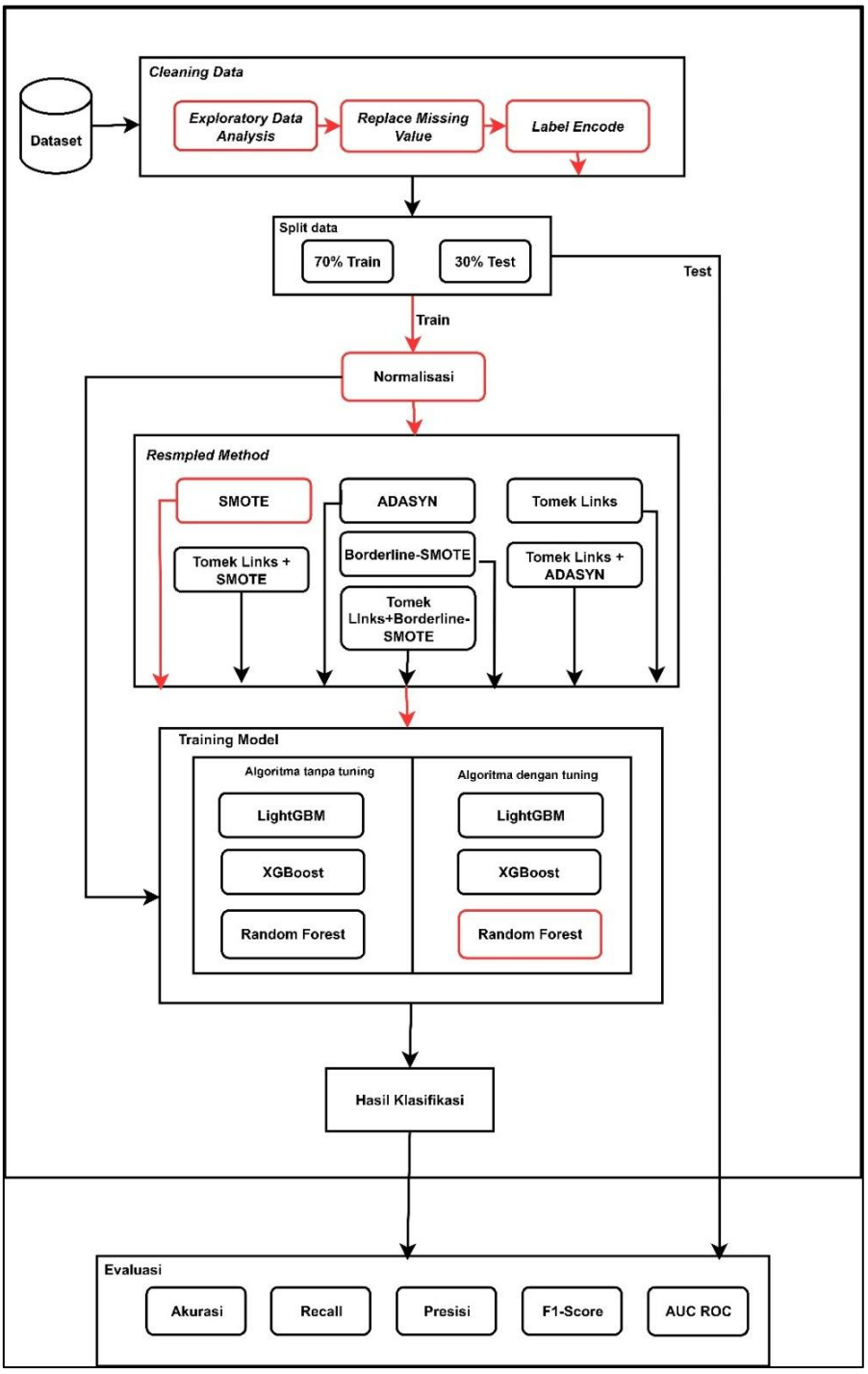

Figure 1. Testing Process Flow

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

Figure 1 shows the research flow starting from the processing of the laboratory examination dataset. The initial stage involves exploratory data analysis (EDA) to understand the characteristics of the data, such as the distribution of values and potential problems found in the dataset. Next, missing values are handled using KNNImputer, and categorical labels are converted to numerical values through the label encoding technique so that the data can be processed by machine learning algorithms.

The processed data is then divided into training data and test data. Normalization is applied by fitting the scaler on the training data and then applying the same transformation to the test data to maintain consistency and avoid data leakage. After the normalization process, the training data then goes through a resampling stage to address the imbalance in the amount of data between the Diabetes Mellitus and Non-Diabetes Mellitus classes.

The next step is to train the model using normalized and balanced training data. During this process, hyperparameter optimization is also performed for each algorithm to obtain a more optimal model configuration. The optimization process was conducted using a fixed number of trials to ensure a consistent evaluation setting across all models. The resulting models are then evaluated using several metrics, namely accuracy, precision, recall, F1-score, and Area Under Curve (AUC), so that the performance of each model can be analyzed comprehensively. In this study, the diabetes mellitus class was treated as the positive class in calculating evaluation metrics. In addition, a comparison of model performance with and without optimization and various combinations of resampling techniques, including combination with Tomek Links, is carried out to see the effect of each approach used.

2.1. Diabetes Mellitus

Diabetes mellitus is defined as a chronic hyperglycemic condition caused by impaired insulin secretion, impaired insulin action, or both. Insulin plays an important role in regulating carbohydrate, fat, and protein metabolism, so that impaired insulin function can trigger metabolic imbalances that affect various body tissues, such as adipose tissue, skeletal muscle, and the liver [1]. In the context of this study, diabetes mellitus is viewed as a chronic disease that requires ongoing management, not only through blood glucose control, but also through regular monitoring and evaluation of the patient's condition to reduce the risk of long-term complications. [2].

The diagnosis of diabetes mellitus is based on the results of blood glucose and glycated hemoglobin (HbA1c) tests, which are performed enzymatically using venous plasma samples. The diagnosis is not only based on the presence of glucosuria, but also takes into account the clinical condition experienced by the patient. Common symptoms include classic diabetes mellitus complaints such as polyuria, polydipsia, and unexplained weight loss. In addition, patients may experience other complaints such as fatigue, tingling, itching, blurred vision, erectile dysfunction in men, and vulvar pruritus in women [15]. The criteria for diagnosing diabetes mellitus can be seen in Table 1.

Table 1. Diagnostic Criteria for Diabetes Mellitus

No	Kriteria
1	Fasting plasma glucose test ≥ 126 mg/dL. Fasting with no calorie intake for at least 8 hours.
2	Plasma glucose test ≥ 200 mg/dL 2 hours after an Oral Glucose Tolerance Test (OGTT) with a glucose load of 75 grams.
3	Fasting plasma glucose test ≥ 200 mg/dL with classic symptoms or hyperglycemic crisis.
4	HbA1c test ≥ 6.5% standardized by the National Glycohemoglobin Standardization Program (NGSP) and the Diabetes Control and Complications Trial assay (DCCT).

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

The data used in this study were obtained from laboratory test results of patients enrolled in the Chronic Disease Management Program (Prolanis), which included patients with diabetes mellitus and non-diabetes mellitus at primary health care facilities. The dataset consisted of 484 laboratory test data with ten clinical parameters as research variables. Details of the parameters used in the analysis and modeling process are presented in Table 2. All patient data used in this study were anonymized to ensure that no personally identifiable information was included in the dataset. The data collection process was conducted with official permission from the related primary health care facility and complied with applicable ethical considerations for research purposes.

Table 2. Medical Check-up Indicators for Prolanis Diabetes Patients

Parameters	Meaning	Function
GDP	Fasting Blood Glucose	Detecting diabetes or glucose disorders.
TG	*Trigliserida*	Assessing blood fat levels, risk of heart disease.
CHOL TOTAL	Total Cholesterol	Total cholesterol (HDL, LDL, etc.) in the blood.
HDL	High-Density Lipoprotein	Transports cholesterol to the liver; protective against heart disease.
LDL	Low-Density Lipoprotein	Can accumulate in blood vessels; risk of atherosclerosis.
UREUM	Ureum	Assessing kidney function, the result of protein metabolism.
CREAT	Kreatinin	Kidney function indicators.
SGPT	Serum Glutamic Pyruvic Transaminase	Assessing damage or inflammation of the liver.
HBA1C	Hemoglobin A1C	Measuring the average blood sugar level over the past 2–3 months.
ALBUMIN URIN	Albumin in Urine	Assessing renal protein leakage (especially in diabetic patients).

2.2. Exploratory Data Analysis

EDA is conducted without any initial assumptions or hypotheses, so the results of the analysis are used as a basis for determining the preprocessing strategy, feature selection, and appropriate modeling methods. The EDA process includes descriptive statistical analysis, exploration of categorical and numerical variables, data visualization, and correlation analysis between features to support data-driven decision making in the next stage [16].

2.3. Replace Missing Value

Replacing missing values is a data preprocessing step that aims to handle empty values in a dataset, because the existence of missing values can affect data quality and reduce the performance of machine learning models. In this study, missing values were imputed using K-Nearest Neighbors Imputer (KNNImputer), an imputation method that estimates missing values based on the average feature values of a number of closest neighbors with similar characteristics, thereby maintaining the data distribution pattern and making the data more representative for the modeling process [17]. In this study, the imputation process was applied prior to splitting the dataset into training and testing sets to ensure consistency in handling missing values across all samples. However, this approach may introduce a

potential risk of data leakage, as information from the test data could be indirectly involved in the imputation process. The KNNImputer calculation can be seen in the equation (1).

$$D_{xy} = \sqrt{w \cdot \sum_{i=1}^{n} (x_i - y_i)^2} \quad (1)$$

Explanation:

- D_{xy} = *Weighted Euclidean distance.*
- w = weight
- n = Number of dimensions or features of vectors x and y.
- $(x_i - y_i)^2$ = The square difference between the i-th element of x and y, measuring the difference in that dimension.

Using weights (2):

$$w = \frac{total\ number\ of\ coordinates}{number\ of\ coordinates\ available} \quad (2)$$

Explanation:

- Total number of coordinates = The number of features or dimensions that should ideally be available in each data set.
- Number of available coordinates = Number of features that are not missing (not empty/missing) in specific data.

2.4. Split Data

The data splitting process helps prevent overfitting, which is a condition where the model only works well on training data but performs poorly on new data. In addition, the selection of the appropriate data splitting scheme and ratio affects the stability and accuracy of the model evaluation results, so it needs to be adjusted to the characteristics of the dataset used [18].

This study uses a 70:30 ratio. To maintain consistency in class distribution in the training and test data, the data division process is carried out using stratified split based on class labels (stratified y). The use of stratified split is considered effective, especially in classification problems with unbalanced class distribution, because it can improve model performance stability in the testing stage [19].

2.5. Normalization

The normalization used in this study employs the StandardScaler method to standardize the scale of each numerical feature before the model training process is carried out. This approach works by transforming the data so that it has a mean value of zero and a standard deviation of one, without changing its basic distribution pattern. Scale normalization is necessary to avoid the dominance of features with larger value ranges, especially in machine learning algorithms that are sensitive to scale differences, so that the model learning process can proceed more stably and objectively [6].

2.6. Method Resampling

Resampling methods were used in this study to address class imbalance by applying oversampling and undersampling approaches and a combination of both, namely Tomek links with Minority Oversampling Technique, Tomek Links with Adasyn, and Tomek Links with Borderline SMOTE, where oversampling increases the number of samples in the minority class and undersampling reduces the samples in the majority class so that the training data distribution becomes more balanced and supports the formation of a more stable model [20].

2.6.1. Synthetic Minority Oversampling Technique

In an effort to address class imbalance, this study applies the Synthetic Minority Oversampling Technique (SMOTE) method. This approach generates new synthetic samples in the minority class through an interpolation process based on data proximity (k-nearest neighbors), so that the resulting data is not duplicated and continues to represent the characteristics of the minority class more naturally.

The application of the SMOTE method aims to improve the representation of minority classes in training data so that class distribution becomes more balanced. With a more proportional data distribution, the classification model training process can run more optimally and reduce potential bias towards majority classes. The working mechanism of SMOTE is shown in Figure 2. This method generates new synthetic samples by utilizing the proximity between data in the minority class, so that the resulting data is not a direct duplication and is able to reduce the risk of overfitting. The process of forming synthetic samples is carried out through the calculation of the nearest neighbors and feature value interpolation [21].

2.6.2. Adaptive Synthetic Sampling

Adaptive Synthetic Sampling (ADASYN) works by generating synthetic samples in minority classes adaptively based on the local distribution of data, so that sample addition is focused on areas with higher classification difficulty. This approach helps machine learning models become more responsive to pattern variations in minority classes and reduces bias towards majority classes [22].

The ADASYN mechanism works by measuring the difficulty level of each sample in the minority class through the proportion of closest neighbors originating from the majority class. Minority samples located in areas dominated by the majority class are considered more difficult to study, so they are prioritized in the synthetic data generation process. This adaptive approach allows ADASYN to focus on adding samples in complex areas, thereby increasing the model's sensitivity to minority classes without adding data evenly [23].

2.6.3. Borderline Synthetic Minority Oversampling Technique

Borderline-SMOTE is designed to strengthen the representation of minority classes in critical areas around the decision boundary. Unlike standard SMOTE, which performs oversampling evenly, this method only generates synthetic data from minority samples in the danger category, i.e., samples that are very close to the majority class and at high risk of being misclassified [24]. By focusing the oversampling process on borderline regions, Borderline-SMOTE is able to improve classification accuracy in areas that are most sensitive to errors.

After the minority samples in the danger category are determined, Borderline-SMOTE generates synthetic data through interpolation with the nearest minority neighbors. This approach strengthens the representation of data in the area around the decision boundary, thereby helping the model form a more accurate class separation.

2.6.4. Tomek Links

Tomek Links is a data cleaning technique used to reduce overlap between classes by identifying pairs of samples from different classes that are closest to each other. Samples that form Tomek Links pairs are considered to be around the decision boundary and can be removed to reduce noise and clarify class separation. The application of this method in the preprocessing stage has been proven to improve data distribution quality and classification performance, especially in minority classes, as illustrated in Figure 5 [25].

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

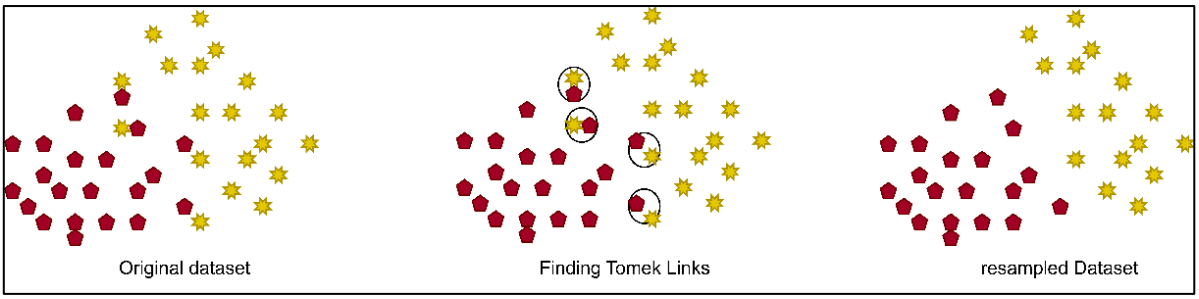

Figure 2. Illustration of How Tomek Links Works

Tomek Links identifies sample pairs from different classes that are closest neighbors and located around the decision boundary. Samples from the majority class in these pairs are then removed to reduce overlap between classes, resulting in a cleaner data distribution and allowing the model to learn class separation more effectively.

2.7. Training Model

The testing process in this study involved three machine learning algorithms as comparators, namely Random Forest, Extreme Gradient Boosting, and Light Gradient Boosting Machine. These three algorithms were used to evaluate the differences in model performance in classifying data in the same test scenario.

2.7.1. Random Forest

Random Forest (RF) is an ensemble-based machine learning algorithm that combines multiple decision trees to produce more consistent and reliable predictions. This algorithm was introduced by Leo Breiman in 2001 as an improvement on conventional decision trees, which are prone to overfitting on training data [26].

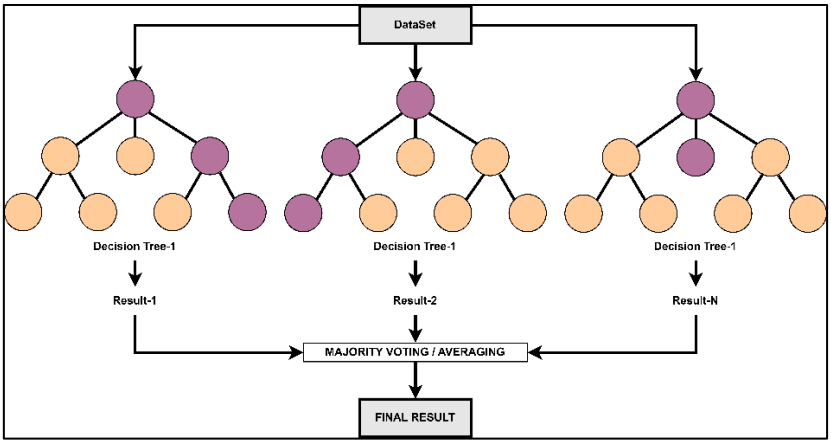

Figure 3. Random Forest

In Figure 6, the formation process shows that RF builds a number of decision trees using training data subsets selected randomly through the bootstrap sampling technique [27]. Each tree provides a prediction, which is then combined through a majority voting mechanism to determine the final output of the model. This approach makes Random Forest more resilient to noise and outliers than single tree methods or certain boosting techniques [28]. The process of forming a decision tree in RF is done by determining the root node based on the entropy value in equation (3) and information gain in equation (4), or using the Gini index in equation (5) followed by the Gini split process in equation (6).

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

$$\boldsymbol{Entropy\ (S) = \sum_{i=1}^{n} -P_i log_2 P_i} \quad (3)$$

Explanation:

- $\boldsymbol{Entropy\ (S)}$ = measure the level of uncertainty or diversity in a data set S.
- $\boldsymbol{P_i}$= Proportion of elements in the i-th class of the total data.
- n = Number of classes.

$$\boldsymbol{Gain\ (S,J) = Entropy\ (S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Entropy(S_i)} \quad (4)$$

Explanation:

- $\boldsymbol{S}$ = Initial data set.
- $\boldsymbol{J}$ = The attribute used for splitting.
- $\boldsymbol{S_i}$ = The i-th subset of data resulting from division by attribute J.

$$\boldsymbol{Gini\ (S) = \sum_{i=1}^{n} (P_i)^2} \quad (5)$$

Explanation:

- $\boldsymbol{Gini\ (S)}$ = Measuring the purity of a data set.
- $\boldsymbol{P_i}$= Proportion of data in class i.
- n = Number of classes.

$$\boldsymbol{Gini_{split} = \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Gini(S_i)} \quad (6)$$

Explanation:

- $\boldsymbol{Gini\ (S)}$ = Average Gini Index.
- $\boldsymbol{S}$ = Initial data set.
- $\boldsymbol{S_i}$ = The i-th subset of data resulting from division by attribute J.
- n = Number of classes.

2.7.2. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a gradient boosting-based ensemble algorithm that builds decision trees incrementally to improve previous prediction errors. Each new tree is trained using residual values, making the learning process more efficient and targeted. XGBoost is equipped with regularization and parallel computing mechanisms that help reduce overfitting and improve model stability. Learning complexity and speed are controlled through parameters such as max depth, learning rate, min child weight, subsample, and n estimators [29]. With this approach, XGBoost is capable of producing high classification and regression performance, as shown in the objective function in equation (7).

$$\boldsymbol{Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_1^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)} \quad (7)$$

Explanation:

- $\boldsymbol{l}$ = loss function.

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

- $\boldsymbol{n}$ = Total number of data samples in training.
- $\boldsymbol{y_i}$ = The actual label (ground truth) of the i-th sample.
- $\boldsymbol{f_k}$ = The k-th decision tree in the ensemble.
- $\Omega(\boldsymbol{f_k})$ = regularisation

Regulation calculation in equation (8).

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

Explanation:

- $\boldsymbol{T}$ = r of leaves on a tree.
- $\boldsymbol{w_j^2}$ = Weight (prediction value) on leaf j.
- $\boldsymbol{\gamma}$ = One-leaf penalty (complexity regularization).
- $\boldsymbol{\lambda}$ = L2 regularization to hold down the weight of the leaf so that the model does not overfit.

To build the t-th tree, XGBoost calculates the first derivative and second derivative using equation (9).

$$\boldsymbol{g_i} = \frac{\boldsymbol{\partial l(y_i, \hat{y}_i)}}{\boldsymbol{\partial \hat{y}_i}}, \boldsymbol{h_i} = \frac{\boldsymbol{\partial^2 l(y_i, \hat{y}_i)}}{\boldsymbol{\partial \hat{y}_i}^2} \tag{9}$$

Explanation:

- $\boldsymbol{g_i}$ = First derivative (gradient).
- $\boldsymbol{h_i}$ = Second derivative (Hessian).

Then the Gain formula for determining the best split is calculated in equation (10).

$$\boldsymbol{Gain} = \frac{\mathbf{1}}{\mathbf{2}}\left[\frac{\boldsymbol{G_L^2}}{\boldsymbol{H_L + \lambda}} + \frac{\boldsymbol{G_R^2}}{\boldsymbol{H_R + \lambda}} - \frac{\boldsymbol{(G_L + G_R)^2}}{\boldsymbol{H_L + H_R + \lambda}} + \right] - \boldsymbol{\gamma} \tag{10}$$

Explanation:

- $\boldsymbol{G_L}$& $\boldsymbol{G_R}$ = Total gradient (first derivative) on the left and right sides of the split.
- $\boldsymbol{H_L}$ & $\boldsymbol{H_R}$= Total Hessian (second derivative) on the left and right sides of the split.

2.7.3. Light Gradient Boosting Machine

Light Gradient Boosting Machine (LightGBM) is a decision tree-based gradient boosting algorithm developed for computational efficiency and scalability in high-dimensional data. LightGBM builds trees using a leaf-wise growth approach by prioritizing nodes with the greatest loss reduction, making the learning process more effective. To accelerate training without sacrificing accuracy, this algorithm utilizes Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques [30]. Model performance is controlled through parameter settings such as num_leaves, tree depth, learning rate, and L1 and L2 regularization, with the aim of minimizing the objective function as shown in equation (11).

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_1{}^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \tag{11}$$

Explanation:

- l = loss function.
- n = Total number of data samples in training.

- y_i = The actual label (ground truth) of the i-th sample.
- f_k = The k-th decision tree in the ensemble.
- $\Omega(f_k)$ = regularization (12)

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum w_j^2 \quad (12)$$

Explanation:

- T = Number of leaves on a tree.
- w_j^2 = Weight (prediction value) on leaf j.
- γ = One-leaf penalty (complexity regularization).
- λ = L2 regularization to hold down the weight of the leaf so that the model does not overfi

To build a new tree, LightGBM uses a second-degree Taylor expansion approach so that the gain of a split is calculated using equation (13).

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} + \right] - \gamma \quad (13)$$

Explanation:

- G_L & G_R = Total gradient (first derivative) on the left and right sides of the split.
- H_L & H_R= Total Hessian (second derivative) on the left and right sides of the split.

With the leaf score calculated in equation (14).

$$w_j = -\frac{G_j}{H_j + \lambda} \quad (14)$$

Explanation:

- w_j = The output value of leaf j, which is the model's contribution to that iteration.
- G_j = Total gradient of all samples entering leaf j.
- H_j= Total hessian of all samples on leaf j.
- λ = L2 regularization coefficient, smoothing leaf weights.

2.8. Hyperparameter Tuning

Hyperparameter tuning is the process of optimizing model parameters that are not directly learned during training, with the aim of obtaining a configuration that can improve classification performance compared to using default parameters [31]. In this study, the hyperparameter tuning process was carried out using Optuna, an optimization framework that works by adaptively evaluating various combinations of hyperparameters through an objective function and a pruning mechanism to stop suboptimal experiments early [32]. To ensure reproducibility, a fixed random state of 42 was used throughout the experiments. The tuning used in this test can be seen in Table 3.

Table 3 shows the hyperparameter search space used in the model testing and optimization process for each algorithm. In Random Forest, testing focused on the number of trees, tree depth, and minimum sample size in the node splitting process to control model complexity. For XGBoost, the parameter range included the number of estimators, tree depth, learning rate, and regularization and subsampling parameters that play a role in balancing accuracy and generalization ability. Meanwhile, LightGBM was tested with variations in the number of leaves, tree depth, learning rate, and L1 and L2 regularization to obtain an efficient and stable configuration. The range of values set aims to provide flexibility in parameter search without increasing the risk of overfitting. The optimization process was conducted for 20 trials for each model using the training data, where model performance was evaluated iteratively to identify the best parameter combination.

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

Table 3. Test Parameters

Algorithm	Parameter	Range
Random Forest	n_estimators	50-150
	max_depth	3-8
	min_samples_split	10-50
	min_samples_leaf	5-40
	max_features	"sqrt"
XGBoost	n_estimators	50-200
	max_depth	2-6
	learning_rate	0.01-0.15
	subsample	0.5-0.8
	colsample_bytree	0.5-0.8
	min_child_weight	5-20
	gamma	0-5
	reg_alpha	0-5
	reg_lambda	0-5
LightGBM	num_leaves	8-32
	max_depth	3-6
	learning_rate	0.01-0.1
	n_estimators	50-200
	min_child_samples	50-200
	lambda_l1	0-5
	lambda_l2	0-5

2.9. Model Evaluation

The confusion matrix in Table 4 is used as the basis for evaluating the performance of the classification model by analyzing the combination of correct and incorrect predictions, which are represented by the values True Positive, False Positive, False Negative, and True Negative [33]. In this study, the diabetes mellitus class is defined as the positive class, while the non-diabetes mellitus class is defined as the negative class. Comparing the evaluation results between training data and test data provides an overview of the model's generalization ability and the potential for overfitting [34]. Model performance is then measured using accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) metrics. These provide an overview of the model's generalization capabilities and the potential for overfitting.

Table 4. Confusion Matrix

Predicted Class	**True Class**	
	Positive	**Negative**
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negatif (FN)	True Negatif (TN)

Accuracy measures how often the model makes correct predictions and can be calculated using equation (15).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (15)$$

Recall or Sensitivity measures how well the model finds all true positive cases, calculated using equation (16).

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

$$Recall = \frac{TP}{TP + FN} \quad (16)$$

Precision indicates how accurate the model's positive predictions are. Of all those predicted as positive, how many are actually positive, calculated as (17).

$$Precision = \frac{TP}{TP + FP} \quad (17)$$

The F1-score is the harmonic mean between precision and recall. This metric is useful when the balance between the two is important, especially on imbalanced data, and can be calculated using equation (18).

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (18)$$

The AUC-ROC value indicates the model's ability to distinguish between positive and negative classes across different threshold settings. An AUC-ROC value of 1 indicates perfect classification performance, while a value closer to 0.5 indicates performance similar to random guessing [35].

3. RESULT

3.1. Diabetes Mellitus Data

The dataset used in this study consisted of 484 laboratory test results with 10 test parameters, covering 313 data points from patients with diabetes mellitus and 171 data points from patients without diabetes mellitus. In the initial stage of data processing, empty values were still found in several test variables. To ensure that all data could be optimally utilized in the modeling process, these empty values were filled using the KNNImputer method. In addition, label encoding was performed to convert categorical data in the labels into numerical form so that it could be processed by machine learning algorithms.

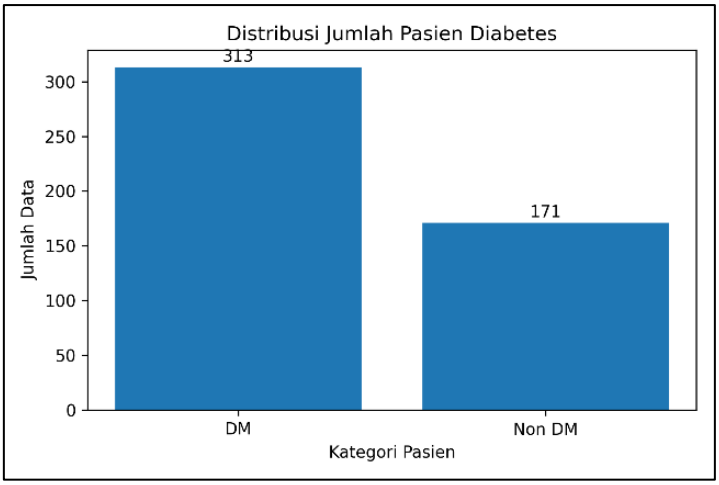

Figure 4. Distribution of data on patients with diabetes mellitus and non-diabetes mellitus

Based on Figure 7, it can be seen that the amount of data for patients with diabetes mellitus is greater than that for patients without diabetes mellitus. This condition indicates an imbalance in class distribution in the dataset, requiring further handling so that the model training process does not tend to be biased towards the class with the more dominant amount of data.

Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

3.2. Split Data

After all data has gone through the preprocessing stage, the dataset is then divided into training data and test data to support the model training and evaluation process. The data is divided with a ratio of 70% as training data and 30% as test data so that the model has enough data to learn and can be tested objectively.

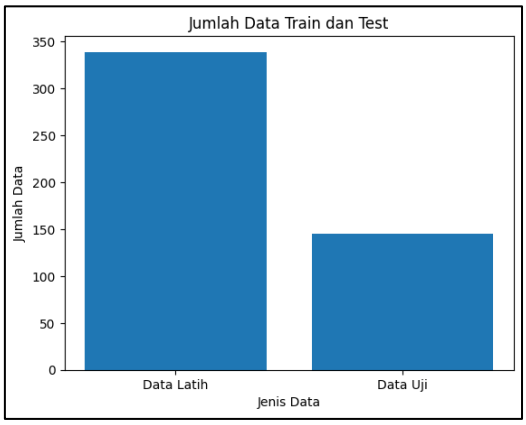

Figure 5. Comparison of training data and test data resulting from dataset division

Based on Figure 8 of the total available data, 338 data were used as training data and 146 data as test data, each with 10 examination features. The class distribution in the training data consisted of 119 patients with non-diabetes mellitus and 219 patients with diabetes mellitus, while in the test data there were 52 patients with non-diabetes mellitus and 94 patients with diabetes mellitus. This division was done to ensure that the model obtained sufficient data during the training stage and could still be evaluated objectively using data that had never been seen before.

3.3. Normalization

After the dataset is divided into training data and test data, the next step is to normalize the training data. Normalization aims to equalize the scale between laboratory test features that have different value ranges, so that no particular feature dominates the model training process. In this study, normalization was performed using the StandardScaler method, which transforms data into a standard scale.

=== Original X_train Data ===

	GDP	TG	CHOL\nTOTAL	HDL	LDL	UREUM	CREAT	SGPT	HBA1C	ALBUMIN\nURIN
0	142.00	266.0	220.0	43.7	123.0	12.0	0.64	9.0	9.40	18.3
1	134.62	151.0	151.0	36.7	89.3	25.0	0.95	28.0	5.92	82.5
2	67.00	153.0	205.0	52.1	166.4	22.0	0.63	11.0	6.91	15.0
3	285.20	119.0	171.0	58.9	88.1	46.0	1.44	13.6	13.33	300.0
4	101.20	91.0	138.0	58.0	47.8	76.0	1.89	12.8	7.87	300.0

(a)

=== Scaled X_train Data ===

	GDP	TG	CHOL\nTOTAL	HDL	LDL	UREUM	CREAT	SGPT	HBA1C	ALBUMIN\nURIN
0	0.270982	1.262545	0.345110	-0.386453	-0.000868	-1.410042	-0.693258	-0.828679	1.307390	-0.452024
1	0.127879	0.002367	-1.133064	-1.024787	-0.839138	-0.324330	0.165569	1.185483	-0.902372	0.106231
2	-1.183312	0.024283	0.023768	0.379547	1.078683	-0.574879	-0.720962	-0.616662	-0.273733	-0.480719
3	3.047713	-0.348292	-0.704608	0.999642	-0.868987	1.429512	1.523068	-0.341040	3.802898	1.997513
4	-0.520154	-0.655118	-1.411560	0.917570	-1.871428	3.935001	2.769752	-0.425847	0.335857	1.997513

(b)

Figure 6. (a) Data Before Normalization; (b) Data After Normalization

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

Based on Figure 9, it can be seen that each feature in the training data that has a wide feature value range (a) has been converted into a standard scale with a mean value close to zero and a standard deviation of one (b). This process ensures that all features have a balanced contribution in model learning and helps improve the stability and performance of the algorithm in the next training stage.

3.4. Resampling Methods

Based on Figure 10, the application of resampling methods was carried out to address the imbalance between the diabetes mellitus and non-diabetes mellitus classes in the training data. Each method produces different data distribution characteristics, which can affect the model's ability to learn minority class patterns.

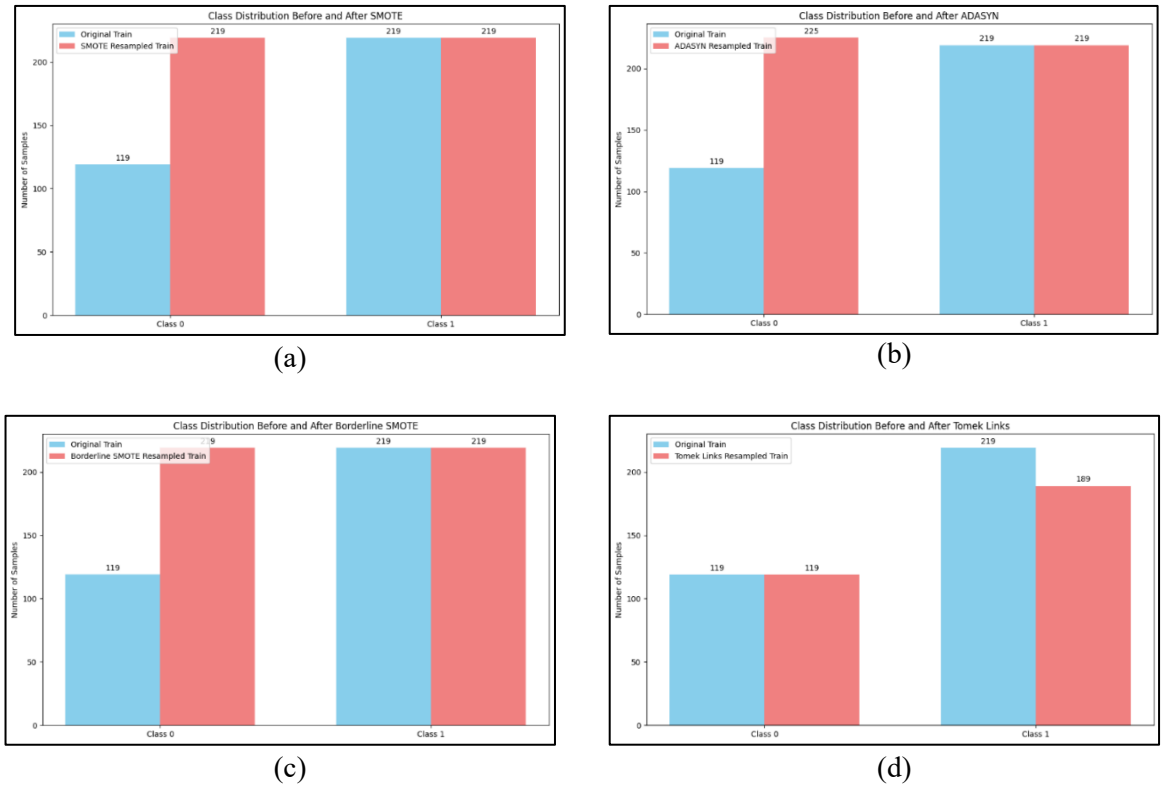

Figure 7. Resample Method Distribution (a) Smote; (b) Adasyn; (c) Borederline-Smote; (d) Tomek Links

Based on Figure 10, the application of several resampling methods shows differences in characteristics in handling data imbalance. In SMOTE, the amount of minority class data is balanced to be equivalent to the majority class through synthetic data generation, so that the distribution of both classes becomes more proportional (a). Furthermore, ADASYN produces a slightly larger amount of minority data because the data addition process is focused on areas that are difficult for the model to learn (b). Next, Borderline-SMOTE forms a balanced distribution with an emphasis on minority samples around the boundary between classes, so that data representation in critical areas becomes stronger (c). Unlike oversampling methods, Tomek Links works by reducing majority class samples around the decision boundary, so that overlap between classes can be minimized (d).

3.5. Training Model dan Hyperparamter Tuning

Model training in this study began with the use of baseline parameters for each algorithm, namely Random Forest, XGBoost, and LightGBM. The baseline configuration was used as an initial reference

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

to observe model performance without optimization, as well as a comparison to the results after hyperparameter tuning. Details of the baseline parameters used in the initial training stage are presented in Table 5.

Table 5. Initial Hyperparameter Configuration

Algoritms	Baseline Parameter Configuration
Random Forest	n_estimators=100; max_depth=5; min_samples_split=20; min_samples_leaf=10; max_features=sqrt
LightGBM	n_estimators=100; max_depth=4; learning_rate=0.05; subsample=1.0; colsample_bytree=1.0; min_child_weight=0.001; num_leaves=16; min_child_samples=50; lambda_l1=0.0; lambda_l2=0.0
XGBoost	n_estimators=100; max_depth=3; learning_rate=0.05; subsample=0.8; colsample_bytree=0.8; min_child_weight=10; gamma=0.0; reg_alpha=0.0; reg_lambda=1.0

After obtaining an initial performance overview, a hyperparameter tuning process was carried out to optimize the performance of each algorithm. Parameter search was performed by determining the search space for each important parameter, such as the number of trees, tree depth, learning rate, and regularization parameters. The range of hyperparameter values used in the optimization process is presented in Table 3. The tuning process was performed automatically using Optuna with 20 trials, where each parameter combination was evaluated using the F1-score value on cross-validation as an objective function.

Table 6. Hyperparameter Configuration with Tuned Optuna

Algoritms	Hyperparameter Configuration
Random Forest	n_estimators= 138; max_depth=6; min_samples_split=10; min_samples_leaf=35; max_features=sqrt
LightGBM	n_estimators=139; max_depth=5; learning_rate=0.023; num_leaves=8; min_child_samples=76; lambda_l1=3.35; lambda_l2=3.06
XGBoost	n_estimators= 140; max_depth=2; learning_rate=0.041; subsample=0.785; colsample_bytree=0.506; min_child_weight=7; gamma=2.46; reg_alpha=3.60; reg_lambda=0.795

The results of the optimization process show that each algorithm produces a different optimal parameter configuration, tailored to the characteristics of the data and the learning method used. A summary of the best F1-score values and the optimal hyperparameter combinations for each algorithm are shown in Table 6. These tuned parameters are then used in the model testing and evaluation stage to ensure more stable and representative performance.

3.6. Model Evaluation

Model evaluation in the initial stage is conducted to observe the algorithm's performance under baseline conditions, i.e., using default parameters without optimization. The assessment is performed on training and test data using five key metrics, including accuracy, precision, recall, F1-score, and AUC ROC. The results of this evaluation serve as a preliminary reference for understanding the model's behavior before hyperparameter tuning is performed.

Based on Table 7, a comparison of the three algorithms without resampling shows that Random Forest has the most balanced performance on the training data with an accuracy of 82.84% and an F1-score of 87.17%. However, on the test data, the accuracy value decreased to 70.55% with an F1-score of 78.39%.

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

Table 7. Baseline Model

Model	Train %					Test %				
	accuracy	**precision**	**Recall**	**F1-Score**	**AUC ROC**	**accuracy**	**precision**	**Recall**	**F1-Score**	**AUC ROC**
XGBoost	66.57	65.96	100	79.49	87.66	65.75	65.28	100	78.99	78.83
Random Forest	82.84	84.55	89.95	87.17	91.09	70.55	74.29	82.98	78.39	76.33
LightGBM	64.79	64.79	100	78.64	91.66	64.38	64.38	100	78.33	77.64

XGBoost and LightGBM showed a very high recall pattern (100%) on the test data, but this was followed by lower accuracy and precision, indicating a tendency for the model to predict most of the data as positive class.

Table 8. Evaluation of Random Forest Without Tuning

Model	**Train %**					**Test %**				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
rf_tomek _baseline	73.08	71.92	95.89	82.19	88.83	69.18	68.99	94.68	79.82	78.19
rf_tomek_ adasyn_base line	64.79	64.79	100	78.64	88.37	64.38	64.38	100	78.33	79.30
rf_tomek_b smote_basel ine	64.79	64.79	100	78.64	87.23	64.38	64.38	100	78.33	78.66
rf_b_smote _baseline	64.79	64.79	100	78.64	88.09	64.38	64.38	100	78.33	77.76
rf_tomek smote_basel ine	64.79	64.79	100	78.64	88.61	64.38	64.38	100	78.33	77.68
rf_adasyn_ baseline	64.79	64.79	100	78.64	88.85	64.38	64.38	100	78.33	77.00
rf_smote_ baseline	82.84	83.13	92.24	87.45	89.75	69.86	75.00	79.79	77.32	76.90

The results of the Random Forest baseline evaluation with various resampling techniques are shown in Table 8. The combination of Random Forest with SMOTE provides the best results among other Random Forest variations, with a test data accuracy of 69.9% and an F1-score of 77.3%. Meanwhile, combinations with ADASYN, Borderline-SMOTE, and Tomek Links tend to produce high recall, but accuracy and precision are still relatively low, ranging from 64-69%. This indicates that under untuned conditions, the effect of resampling on Random Forest is not yet significant in improving test data performance.

The evaluation of the LightGBM baseline in Table 9 shows that the combination with Tomek Links produces the best performance on the test data, with an accuracy of 74.66% and an F1-score of 81.41%. However, most other resampling combinations show recall close to 100% but with lower precision, resulting in F1-scores in the range of 78-79%. This pattern indicates that the LightGBM baseline still experiences an imbalance between the ability to detect positive classes and prediction accuracy.

Furthermore, Table 10 shows the results of the Xgboost evaluation with various resampling methods. The ADASYN combination provides test data accuracy of 71.92% with an F1-score of 78.31%, while the Tomek Links combination produces the highest F1-score of 80.00%. However, the difference in performance between combinations is relatively small, indicating that without parameter optimization, Xgboost is not yet able to utilize resampling techniques to their full potential.

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

Table 9. Evaluation of LightGBM Without Tuning

Model	Train%					Test%				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
lgbm_tomek_baseline	81.66	81.53	92.69	86.75	88.60	74.66	77.14	86.17	81.41	79.87
lgbm_smote_baseline	67.75	66.77	100	80.07	91.65	67.12	66.20	100	79.66	78.60
lgbm_tomek adasyn_baseline	64.79	64.79	100	78.64	88.47	65.75	65.28	100	78.99	80.46
lgbm_adasyn_baseline	65.68	65.37	100	79.06	90.44	65.75	65.28	100	78.99	77.74
lgbm_tomek_b_smote_baseline	65.38	65.27	99.54	78.84	88.55	65.07	64.83	100	78.66	81.28
lgbm_tomek_smote_baseline	65.38	65.18	100	78.92	89.11	64.38	64.38	100	78.33	78.56
lgbm_b_smote_baseline	65.38	65.18	100	78.92	90.72	64.38	64.38	100	78.33	77.09

Table 10. Evaluation of Xgboost Without Tuning

Model	Train %					Test %				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
xgb_tomek_baseline	78.40	78.52	91.78	84.63	85.31	72.60	75.47	85.11	80.00	78.44
xgb_b,smote_baseline	68.34	67.28	99.54	80.29	87.86	67.81	66.91	98.94	79.83	81.34
xgb_tomek_b,smote_baseline	66.27	65.77	100	79.35	85.53	67.12	66.20	100	79.66	79.99
xgb_tomek_adasyn_baseline	65.38	65.18	100	78.92	85.87	65.07	64.83	100	78.66	80.30
xgb_smote_baseline	64.79	64.79	100	78.64	87.59	64.38	64.38	100	78.33	79.11
xgb_adasyn_baseline	81.07	82.98	89.04	85.90	87.43	71.92	77.89	78.72	78.31	78.89
xgb_tomek_smote_baseline	66.86	66.36	99.09	79.49	86.94	65.07	65.25	97.87	78.30	80.26

Table 11. Model Tuning

Model	Train%					Test%				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
XGboost	72.19	73.41	89.50	80.66	78.83	70.55	72.97	86.17	79.02	78.85
LightGBM	65.09	67.12	90.41	77.04	73.84	67.12	67.69	93.62	78.57	72.19
Random forest	64.79	64.79	100	78.64	79.20	64.38	64.38	100.00	78.33	74.02

Overall, the four tables show that under baseline conditions, most models produce high recall but are not balanced with stable accuracy and precision on the test data. The difference in performance between the training data and the test data is still evident, so a further stage of hyperparameter tuning is needed to improve model generalization and optimize the use of resampling techniques.

After hyperparameter tuning, the performance of the three algorithms showed more controlled changes compared to the baseline configuration. The results of the evaluation of the tuned model without

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

resampling are shown in Table 11. At this stage, the tuned XGBoost recorded a test data accuracy of 70.55% with an F1-score of 79.02%, which was more stable than the baseline version. Meanwhile, the tuned Random Forest and LightGBM without resampling did not show significant improvements in the test data, especially in the precision and AUC metrics, indicating that tuning alone is not sufficient to overcome class imbalance.

Table 12. Evaluation of Random Forest with Tune

Model	Train%					Test%				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
rf_smote_tuned	71.89	70.81	96.35	81.62	79.21	72.60	72.13	93.62	81.48	76.74
rf_adasyn_tuned	70.71	69.48	97.72	81.21	80.10	71.23	70.31	95.74	81.08	76.94
rf_tomek_b smote_tuned	66.86	66.26	99.54	79.56	78.56	67.81	66.67	100	80.00	78.21
rf_tomek_smote_tuned	67.75	66.87	99.54	80.00	77.91	67.12	66.67	97.87	79.31	75.49
rf_tomek_tuned	70.12	70.07	94.06	80.31	77.57	69.18	69.92	91.49	79.26	75.16
rf_tomek adasyntuned	64.79	64.79	100	78.64	78.55	64.38	64.38	100	78.33	78.91
rf_b_smote_tuned	64.79	64.79	100	78.64	79.92	64.38	64.38	100	78.33	76.76

Table 13. Evaluation of LightGBM with Tuned

Model	Train%					Test%				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
lgbm_b_smote_tuned	68.05	68.20	94.98	79.39	76.47	70.55	70.40	93.62	80.37	71.99
lgbm_adasyn_tuned	67.46	67.30	96.80	79.40	76.56	68.49	68.75	93.62	79.28	74.41
lgbm_smote_tuned	66.57	67.10	94.98	78.64	76.41	68.49	69.05	92.55	79.09	72.89
lgbm_tomek_b_smote_tuned	65.98	66.67	94.98	78.34	73.75	67.12	67.69	93.62	78.57	72.64
lgbm_tomek_tuned	64.79	64.79	100	78.64	72.65	64.38	64.38	100	78.33	70.54
lgbm_tomek_adasyn_tuned	66.86	68.38	90.87	78.04	73.29	67.81	69.42	89.36	78.14	71.55
lgbm_tomek_smote_tuned	68.93	70.96	88.13	78.62	73.81	67.81	70.09	87.23	77.73	72.04

The effect of tuning becomes more apparent when combined with resampling methods. Table 12 presents the evaluation results of Random Forest after tuning with various resampling strategies. The RF-SMOTE tuned combination produced the most consistent performance, with a test data accuracy of 72.60%, precision of 72.13%, recall of 93.62%, and the highest F1-score of 81.48%. These results indicate that Random Forest responds positively to tuning when the class distribution is made more balanced. Other combinations such as RF-ADASYN and RF-Tomek also show an increase in recall, but with relatively lower precision

The results of LightGBM evaluation after tuning are shown in Table 13. In general, tuning was able to improve the balance between precision and recall compared to the initial configuration. The

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

combination of LightGBM with SMOTE and ADASYN produced a test data F1-score in the range of 79-80%, with an accuracy of 68-70%. However, the improvement in LightGBM performance is still below that of Random Forest, especially in terms of prediction stability and AUC value.

Table 14. Evaluation of Xgboost With Tuned

Model	Train					Test				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
xgb_adasyn_tuned	70.41	69.90	95.43	80.69	80.88	71.23	70.31	95.74	81.08	78.78
xgb_tomek_b_smote_tuned	68.64	67.82	98.17	80.22	79.13	70.55	69.17	97.87	81.06	78.61
xgb_b_smote_tuned	66.57	66.26	98.63	79.27	80.47	69.86	68.38	98.94	80.87	80.26
xgb_tomek_adasyn_tuned	70.12	69.67	95.43	80.54	78.96	69.18	69.29	93.62	79.64	79.32
xgb_tomek_smote_tuned	67.75	66.98	99.09	79.93	79.51	65.75	65.94	96.81	78.45	77.72
xgb_smote_tuned	64.79	64.79	100	78.64	80.31	64.38	64.38	100	78.33	79.08
xgb_tomek tuned	64.79	64.79	100	78.64	78.36	64.38	64.38	100	78.33	77.28

Furthermore, Table 14 shows the performance of XGBoost that has been tuned and combined with various resampling methods. Several configurations, such as tuned XGB-ADASYN and tuned XGB-Tomek-B-SMOTE, were able to achieve a test data F1-score above 81% with an accuracy of around 70-71%. However, the variation in performance between combinations is still quite wide, indicating that XGBoost tends to be more sensitive to changes in data distribution than Random Forest.

The evaluation results on the test data show that several combinations of models and resampling methods provide relatively similar performance based on the accuracy, precision, recall, and F1-score metrics. However, these metrics do not fully describe the model's ability to distinguish classes at various decision thresholds. Therefore, additional analysis was performed using Receiver Operating Characteristic (ROC) curves to assess class discrimination capabilities more comprehensively.

Table 15. Comparison of Best Model Performance

Model	Train					Test				
	accuracy	precision	Recall	F1-Score	AUC ROC	accuracy	precision	Recall	F1-Score	AUC ROC
rf_smote_tuned	71.89	70.81	96.35	81.62	79.21	72.60	72.13	93.62	81.48	76.74
xgb_tomek_adasyn_tuned	70.12	69.67	95.43	80.54	78.96	69.18	69.29	93.62	79.64	79.32
lgbm_smote_tuned	66.57	67.10	94.98	78.64	76.41	68.49	69.05	92.55	79.09	72.89

Table 15 presents the performance of the best configuration for each model on both training and test data. The Random Forest model with SMOTE and hyperparameter tuning achieved the highest performance on the test data, with an accuracy of 72.60 %, precision of 72.13 %, recall of 93.62 %, F1-score of 81.48 %, and AUC ROC of 76.74 %.

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

The XGBoost model with Tomek Links, ADASYN, and hyperparameter tuning obtained a test accuracy of 69.18 percent, precision of 69.29 %, recall of 93.62 %, F1-score of 79.64 %, and AUC ROC of 79.32 %. Meanwhile, the Light Gradient Boosting Machine model with SMOTE and tuning produced a test accuracy of 68.49 %, precision of 69.05 %, recall of 92.55 %, F1-score of 79.09 %, and AUC ROC of 72.89 %.

Across all models, recall values are consistently high, indicating that most diabetes cases are correctly identified. In contrast, precision values are relatively lower than recall, showing that false positive predictions are still present. A comparison between training and test results also shows that the performance differences are relatively small, suggesting that the models are able to generalize without significant overfitting.

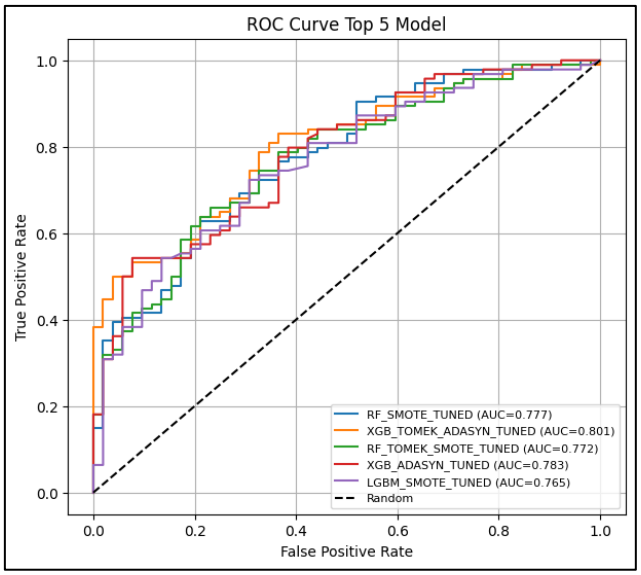

Figure 8. ROC curve

Furthermore, Figure 11 presents a comparison of the ROC curves of the five models with the best performance. All curves are above the random line, indicating that each model has a good ability to distinguish between the Diabetes Mellitus and Non-Diabetes Mellitus classes. The differences in AUC values on each curve reflect variations in model reliability, where some resampling combinations produce more stable curves that are closer to the upper left corner. These results confirm that the balance of data distribution affects the model's ability to perform consistent classification.

4. DISCUSSIONS

The results of this study confirm that data characteristics have a direct influence on the performance of the Diabetes Mellitus classification model. The dataset used was derived from real patient laboratory test data with an unbalanced class distribution, where the number of Diabetes Mellitus patients was more dominant than Non-Diabetes Mellitus patients. This condition has the potential to cause model bias towards the majority class if no special handling is done, as also reported in previous studies [13].

The application of resampling techniques has been proven to help improve class distribution in training data. Oversampling methods such as SMOTE and ADASYN are able to increase the sensitivity of the model in recognizing the Diabetes Mellitus class, as reflected in the increased recall and F1-score values in the test data. However, ADASYN shows more fluctuating performance variations due to its mechanism that focuses on areas that are difficult to classify, in line with the findings in previous studies. [36]. On the other hand, Borderline-SMOTE strengthens data representation around class boundaries,

Vol. 7, No. 3, June 2026, Page. 3025-3049
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5751

helping models form clearer decision boundaries, as also reported in similar studies. [23]. These findings are consistent with previous research showing that the application of SMOTE can significantly improve model performance, where accuracy and F1-score can reach up to 97 % when applied to Random Forest models on secondary datasets [13].

Unlike oversampling methods, Tomek Links acts as a data cleaning technique by reducing overlap between classes. The results of this study show that Tomek Links is more effective when combined with oversampling than when used alone. This pattern is consistent with previous studies which state that removing borderline samples can improve model stability, especially in medical data where there is overlap between classes [37].

From an algorithmic perspective, Random Forest demonstrated the most stable performance across various testing scenarios. Its resilience to noise and ability to handle imbalanced data distributions made this algorithm more consistent than XGBoost and LightGBM. These findings are consistent with previous research indicating that Random Forest has advantages in disease classification based on clinical and laboratory data [38]. In this study, the best model achieved an F1-score of 81.48 %and an AUC ROC of 76.74 %, which indicates a moderate performance level when compared to previous studies. The difference in performance may be influenced by the use of real laboratory data with a smaller sample size (484 records), which tends to be more heterogeneous and complex than publicly available datasets. Conversely, XGBoost and LightGBM tend to be more sensitive to changes in data distribution, so their performance is greatly affected by the combination of resampling and parameter settings.

The hyperparameter tuning process further emphasizes the importance of adjusting models to data characteristics. Without optimization, some algorithms are unable to utilize data information optimally. After tuning, performance improvements were more consistent, especially with the combination of Random Forest and SMOTE, which produced a better balance between accuracy, precision, and F1-score. This finding is also supported by previous studies showing that hyperparameter optimization techniques can improve model performance and generalization ability, particularly in handling imbalanced medical datasets. Overall, these results reinforce the conclusion that data quality, preprocessing strategies, and parameter optimization play an equally important role as algorithm selection in the development of machine learning-based Diabetes Mellitus prediction systems [13],[38].

Furthermore, this study contributes to the field of informatics by proposing a structured machine learning pipeline for handling imbalanced clinical tabular data, including preprocessing, resampling, and hyperparameter optimization strategies that can be applied to similar healthcare prediction problems.

5. CONCLUSION

This study presents a comparative analysis of three machine learning algorithms, namely Random Forest, XGBoost, and LightGBM, for predicting diabetes mellitus using resampling techniques and Optuna-based hyperparameter optimization. The results show that the application of resampling and parameter tuning improves model performance, particularly in metrics that reflect classification balance such as recall and F1-score. Among all tested combinations, Random Forest with SMOTE and hyperparameter tuning achieved the best performance, with an accuracy of 72.60 %, precision of 72.13 %, recall of 93.62 %, and F1-score of 81.48 %. These results indicate that the model is relatively effective in identifying diabetes cases while maintaining a balance between prediction accuracy and completeness.

In comparison, the XGBoost and LightGBM algorithms also produced competitive performance, with some resampling combinations achieving F1-scores above 81 percent. However, their performance tended to be more sensitive to changes in data distribution, as reflected in the variation of accuracy and

AUC-ROC across different scenarios. This suggests that model stability in boosting-based approaches is highly dependent on the choice of resampling technique and parameter configuration.

Overall, this study shows that the integration of resampling techniques and hyperparameter optimization contributes to improving classification performance on imbalanced laboratory data. From an informatics perspective, this study provides a structured machine learning pipeline for handling imbalanced clinical tabular data through preprocessing, resampling, and model tuning strategies.

However, this study has limitations, including the relatively small dataset size (484 records) and the use of data from a single healthcare facility. In addition, the model has not been validated using external or multi-center data. Therefore, although the results are promising, further validation is required before considering practical implementation in clinical decision support systems.

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest between authors or with the research subjects in this article.

ACKNOWLEDGEMENT

The author would like to thank the health care facilities that provided permission and research data, as well as all those who assisted in the data collection and processing. This research did not receive any special funding from any institution.

REFERENCES

[1] S. A. Antar *et al.*, "Diabetes mellitus: Classification, mediators, and complications; A gate to identify potential targets for the development of new effective treatments," Dec. 01, 2023, *Elsevier Masson s.r.l.* doi: 10.1016/j.biopha.2023.115734.

[2] American Diabetes Association, "Introduction and Methodology: Standards of Care in Diabetes—2024," Jan. 01, 2024, *American Diabetes Association Inc.* doi: 10.2337/dc24-SINT.

[3] Kementerian Kesehatan Republik Indonesia, Badan Kebijakan Pembangunan Kesehatan, "Survei Kesehatan Indonesia," 2023.

[4] C. Z. V. Junus, T. Tarno, and P. Kartikasari, "Klasifikasi menggunakan Metode Support Vector Machine dan Random Forest untuk Deteksi Awal Risiko Diabetes Melitus," *Jurnal Gaussian*, vol. 11, no. 3, pp. 386–396, Jan. 2023, doi: 10.14710/j.gauss.11.3.386-396.

[5] I. Permana and F. Nur Salisah, "Pengaruh Normalisasi Data Terhadap Performa Hasil Klasifikasi Algoritma Backpropagation," *Indonesian Journal of Informatic Research and Software Engineering*, vol. 2, pp. 67–72, Mar. 2022.

[6] F. Aldi, F. Hadi, N. A. Rahmi, and S. Defit, "Standardscaler's Potential In Enhancing Breast Cancer Accuracy Using Machine Learning," *Journal of Applied Engineering and Technological Science*, vol. 5, no. 1, pp. 401–413, 2023.

[7] E. Sutoyo and M. Asri Fadlurrahman, "Penerapan SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Television Advertisement Performance Rating Menggunakan Artificial Neural Network," *Jurnnal Edukasi dan Penelitian Informatika)*, vol. 6, Dec. 2020.

[8] L. Qadrini, H. Hikmah, and M. Megasari, "Oversampling, Undersampling, Smote SVM dan Random Forest pada Klasifikasi Penerima Bidikmisi Sejawa Timur Tahun 2017," *Journal of Computer System and Informatics (JoSYC)*, vol. 3, no. 4, pp. 386–391, Sep. 2022, doi: 10.47065/josyc.v3i4.2154.

[9] Y. M. Indah, R. Aristawidya, A. Fitrianto, E. Erfiani, and L. M. R. D. Jumansyah, "Comparison of Random Forest, XGBoost, and LightGBM Methods for the Human Development Index Classification," *Jambura Journal of Mathematics*, vol. 7, no. 1, pp. 14–18, Jan. 2025, doi: 10.37905/jjom.v7i1.28290.

[10] D. Cahya and P. Buani, "Deteksi Dini Penyakit Diabetes dengan Menggunakan Algoritma Random Forest," *Jurnal Sains dan Manajemen*, vol. 12, no. 1, 2024.

[11] I. Muhamad and M. Matin, "Hyperparameter Tuning menggunakan GridsearchCV pada Random Forest untuk Deteksi Malware," Politeknik Negeri Jakarta, May 2023. doi: 10.32722/multinetics.v9i1.5578.

[12] J. P. Lai, Y. L. Lin, H. C. Lin, C. Y. Shih, Y. P. Wang, and P. F. Pai, "Tree-Based Machine Learning Models with Optuna in Predicting Impedance Values for Circuit Analysis," *Micromachines (Basel).*, vol. 14, no. 2, Feb. 2023, doi: 10.3390/mi14020265.

[13] Nurussakinah, M. Faisal, and I. Budi Santoso, "Algoritma Random Forest dan Synthetic Minority Oversampling Technique (SMOTE) untuk Deteksi Diabetes," *Jurnal Informatika Sunan Kalijaga*, vol. 10, no. 2, pp. 223–234, May 2025, Accessed: Apr. 30, 2026. [Online]. Available: https://creativecommons.org/licenses/by-nc/4.0/

[14] M. A. Abubakar, M. Muliadi, A. Farmadi, R. Herteno, and R. Ramadhani, "Random Forest Dengan Random Search Terhadap Ketidakseimbangan Kelas Pada Prediksi Gagal Jantung," *Jurnal Informatika*, vol. 10, no. 1, pp. 13–18, Mar. 2023, doi: 10.31294/inf.v10i1.14531.

[15] Perkumpulan Endokrinologi Indonesia, "Pedoman Pengelolaan dan Pencegahan Diabetes Melitus Tipe 2 Dewasa Di Indonesia," 2021.

[16] M. B. Courtney, "Exploratory Data Analysis in Schools: A Logic Model to Guide Implementation," *International Journal of Education Policy and Leadership*, vol. 17, no. 4, May 2021, doi: 10.22230/ijepl.2021v17n4a1041.

[17] R. N. P. Pratama, S. Winarno, and T. N. O. Wijaya, "Thyroid Disease Prediction Using Random Forest with KNNImputer for Missing Values," *sinkron*, vol. 9, no. 1, pp. 160–166, Jan. 2025, doi: 10.33395/sinkron.v9i1.14334.

[18] R. Oktafiani, A. Hermawan, and D. Avianto, "Pengaruh Komposisi Split data Terhadap Performa Klasifikasi Penyakit Kanker Payudara Menggunakan Algoritma Machine Learning," *Jurnal Sains dan Informatika*, pp. 19–28, Jun. 2023, doi: 10.34128/jsi.v9i1.622.

[19] S. F. Kadir and A. Fairuzabadi, "Analisis Sentimen Ulasan Shopee di Google Play dengan TF-IDF dan Logistic Regression," *RIGGS: Journal of Artificial Intelligence and Digital Business*, vol. 4, no. 2, pp. 7940–57945, Jul. 2025, doi: 10.31004/riggs.v4i2.2850.

[20] A. Larasti, S. Surono, A. Thobirin, and D. A. Dewi, "Performance Analysis of Resampling Techniques for Overcoming Data Imbalance in Multiclass Classification," *Jurnal Informatika*, vol. 13, pp. 57–66, Mar. 2025.

[21] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information (Switzerland)*, vol. 14, no. 1, Jan. 2023, doi: 10.3390/info14010054.

[22] M. Tiara, T. B. Sirait, N. S. Fathonah, and M. N. Fauzan, "Pemanfaatan Algoritma ADASYN dan Support Vector Machine dalam Meningatkan Akurasi Prediksi Kanker Paru-paru," *Jurnal Mahasiswa Teknik Informatika*, vol. 8, no. 5, Oct. 2024.

[23] S. Alwaliyanto, G. Kurnia, I. Afrianty, and F. Syafria, "BULLETIN OF COMPUTER SCIENCE RESEARCH Penerapan Metode ADASYN Dalam Mengatasi Imbalanced Data Untuk Klasifikasi Penyakit Stroke Menggunakan Support Vector Machine," *Media Online)*, vol. 5, no. 4, pp. 532–541, 2025, doi: 10.47065/bulletincsr.v5i4.612.

[24] Y. Sun *et al.*, "Borderline SMOTE Algorithm and Feature Selection-Based Network Anomalies Detection Strategy," *Energies (Basel).*, vol. 15, no. 13, Jul. 2022, doi: 10.3390/en15134751.

[25] X. H. Le, S. Eu, C. Choi, D. H. Nguyen, M. Yeon, and G. Lee, "Machine learning for high-resolution landslide susceptibility mapping: case study in Inje County, South Korea," *Front. Earth Sci. (Lausanne).*, vol. 11, 2023, doi: 10.3389/feart.2023.1268501.

[26] Suci Amaliah, M. Nusrang, and A. Aswi, "Penerapan Metode Random Forest Untuk Klasifikasi Varian Minuman Kopi di Kedai Kopi Konijiwa Bantaeng," *VARIANSI: Journal of Statistics and Its application on Teaching and Research*, vol. 4, no. 3, pp. 121–127, Dec. 2022, doi: 10.35580/variansiunm31.

[27] A. F. Anjani, D. Anggraeni, and I. M. Tirta, "Implementasi Random Forest Menggunakan SMOTE untuk Analisis Sentimen Ulasan Aplikasi Sister for Students UNEJ," *Jurnal Nasional Teknologi dan Sistem Informasi*, vol. 9, no. 2, pp. 163–172, Sep. 2023, doi: 10.25077/teknosi.v9i2.2023.163-172.

[28] H. A. Salman, A. Kalakech, and A. Steiti, "Random Forest Algorithm Overview," *Babylonian Journal of Machine Learning*, vol. 2024, pp. 69–79, Jun. 2024, doi: 10.58496/bjml/2024/007.
[29] A. Brahmandjati, A. Mizwar, A. Rahim, and F. Asharudin, "Optimasi Prediksi Diabetes Dengan Algoritma XGBoost Dan Teknik Preprocessing Data," *Jurnal Ilmu Komputer dan Pendidikan*, vol. 3, pp. 116–125, Dec. 2024.
[30] A. A. Nafea *et al.*, "A Machine Learning Technique for Early Detection of Gestational Diabetes Mellitus Using SMOTE and Optimized Light Gradient Boosting Machine," *Journal of Artificial Intelligence in Medical Applications (JAIMA)*, vol. 1, no. 1, pp. 55–63, 2025, Accessed: Apr. 30, 2026. [Online]. Available: https://jaima.uoanbar.edu.iq/index.php/jaima
[31] A. Ainul Yaqin *et al.*, "Implementation of the Random Forest Algorithm with Optuna Optimization in Lung Cancer Classification," vol. 14, no. 2, Jan. 2025, [Online]. Available: http://sistemasi.ftik.unisi.ac.id
[32] A. Tikaningsih, P. Lestari, A. Nurhopipah, I. Tahyudin, E. Winarto, and N. Hassa, "Optuna Based Hyperparameter Tuning for Improving the Performance Prediction Mortality and Hospital Length of Stay for Stroke Patients," *Telematika*, vol. 17, no. 1, pp. 1–16, Feb. 2024, doi: 10.35671/telematika.v17i1.2816.
[33] Ž. Vujović, "Classification Model Evaluation Metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021, doi: 10.14569/IJACSA.2021.0120670.
[34] W. A. Firmansyach, U. Hayati, and Y. A. Wijaya, "Analisis Terjadinya Overfitting dan Underfitting pada Algoritma Naive Bayes dan Decision Tree dengan Teknik Cross Validation," *Jurnal Mahasiswa Teknik Informatika*, vol. 7, no. 1, 2023.
[35] Jude Chukwura Obi, "A comparative study of several classification metrics and their performances on data," *World Journal of Advanced Engineering Technology and Sciences*, vol. 8, no. 1, pp. 308–314, Feb. 2023, doi: 10.30574/wjaets.2023.8.1.0054.
[36] A. Cahyana, E. R. Susanto, and Parjito, "Penerapan Algoritma XGBoost untuk Prediksi Diabetes: Analisis Confusion Matrix dan ROC Curve," *Fountain of Informatics Journal*, vol. 10, no. 1, pp. 40–50, May 2025, doi: 10.21111/fij.v10i1.14311.
[37] H. Kurniawan, A. Dwi Akbar, N. Svensons, Y. Jaya Antonio, S. Karnila, and E. Safitri, "Evaluasi Performa Random Forest, XGBoost, dan LightGBM dalam Diagnosis Dini Diabetes Mellitus," *Jurnal JUPITER*, vol. 17, no. 93, pp. 835–844, May 2025.
[38] V. L. Anjani, S. Novalina Turnip, U. N. Uzlifah, and R. Kusumastuti, "Sistem Deteksi Dini Penyakit Diabetes Menggunakan Algoritma Random Forest," no. Vol. 1 No. 01 (2025): JUSINFO: Jurnal Sistem Informasi, pp. 1–9, Nov. 2025.