International Journal of Engineering and Computer Science Applications (IJECSA)
Vol. 5, No. 1, March 2026, pp. 9∼22
ISSN: 2828-5611
DOI: 10.30812/IJECSA.v5i1.6080

Analysis of Preprocessing Technique Combinations and Hyperparameter Tuning for Building a Reliable Random Forest–Based Stroke Prediction Model

Aidina Ristyawan, Arie Nugroho
Universitas Nusantara PGRI, Kediri, Indonesia

Article Info

Article history:

Received January 04, 2026
Revised January 27, 2026
Accepted February 06, 2026

Keywords:

Hyperparameter-Tuning Combinations;
Model Fitting Analysis;
Preprocessing Combinations;
Random Forest;
Stroke Prediction.

ABSTRACT

Stroke is a major health threat that can result in permanent disability or death, yet its risks can be mitigated through accurate early detection. Although the Random Forest algorithm is frequently used for stroke prediction, prior studies have often neglected model reliability, specifically the stability of performance between training and testing. This research aims to develop a reliable stroke prediction model by applying the CRISP-DM methodology to a public dataset comprising 5,110 data points. The proposed methodology involves a comprehensive evaluation of 48 preprocessing technique combinations—addressing missing values in the BMI attribute, categorical transformation, feature scaling, and class balancing—followed by a two-stage hyperparameter optimization strategy: Randomized Search for broad exploration and Grid Search Refine for local refinement to ensure optimal stability. Model performance was evaluated using accuracy, precision, recall, and F1-score metrics. The results demonstrate that hyperparameter tuning successfully enhanced model performance by up to 38.80%. Additionally, the hybrid balancing technique (SMOTETomek) did not consistently yield the most stable models in this case. The optimal model (Model No. 8) achieved a training accuracy of 0.925 and a testing accuracy of 0.877. With a minimal performance gap of 0.047 (below the 0.05 threshold), this model is classified as ”good fitting,” indicating superior generalization. Consequently, this model is highly recommended for implementation as a robust and trustworthy early-warning decision-support system for medical professionals.

Corresponding Author:

Aidina Ristyawan,
Universitas Nusantara PGRI Kediri, Kediri, Indonesia,
Email: aidinaristi@unpkediri.ac.id

How to Cite: A. Ristyawana and A. Nugroho, “Analysis of Preprocessing Technique Combinations and Hyperparameter Tuning for Building a Reliable Random Forest–Based Stroke Prediction Model”, *International Journal of Engineering and Computer Science Applications (IJECSA)*, vol. 5, no. 1, pp. 9–22, Mar. 2026. doi: 10.30812/ijecsa.v2i2.3406.

1. INTRODUCTION

Stroke is one of the most dangerous and life-threatening diseases; however, its severe impact can be prevented if it is detected at an early stage [1]. Early detection of stroke symptoms is crucial for predicting and preventing long-term consequences, as it can save lives by identifying transient ischemic attacks (mini-strokes) before a major stroke occurs [1, 2]. Technically, a stroke is a serious neurological disorder that occurs when blood flow to the brain suddenly stops due to blockage or rupture of blood vessels [2]. This condition leads to the death of brain nerve cells and can result in permanent disability or death. Several factors contribute to stroke risk, including age, heart disease, diabetes mellitus, hypertension, and body mass index (BMI) [3]. Recent studies have explored various machine learning techniques to optimize stroke prediction. For instance, stroke detection models using Logistic Regression combined with SMOTE reached an accuracy of 86% [4]. Other research utilizing Support Vector Machine (SVM) and SMOTE achieved accuracies between 85.24% and 85.45% [5]. Comparative experiments using various algorithms, such as KNN, Naive Bayes, Decision Trees, and Neural Networks, demonstrated that Random Forest consistently yielded the highest accuracy at 94% [6]. As an ensemble learning method, Random Forest is highly suitable for complex medical datasets because it reduces overfitting and improves predictive performance [7, 8]. However, this algorithm still faces challenges with high-cardinality categorical variables, imbalanced datasets, and model interpretability [9].

Efforts to enhance Random Forest performance have been extensive. Previous studies improved stroke prediction precision through balanced data handling and cross-validation [6, 10]. The application of attribute normalization on imbalanced datasets achieved up to 99% accuracy [11]. Furthermore, integrating hybrid sampling (SMOTE-Tomek) and KNNImputer successfully increased stroke prediction accuracy to between 95% and 96% [12, 13]. Feature selection methods like Forward and Backward Selection have also reached accuracies up to 99.03% in related medical contexts [14]. Despite these high-performance metrics, a significant research gap remains. Most existing studies focus primarily on standard evaluation metrics, as shown in Table 1, such as accuracy, precision, and recall, without conducting a rigorous analysis of model reliability through fitting analysis [15]. A model with high accuracy that suffers from overfitting due to excessive complexity will fail to generalize on new data, thereby reducing the quality and trustworthiness of medical decision support systems [15, 16]. Implementing a data-driven approach to balance the trade-off between overfitting and underfitting is essential to ensure that the model remains accurate and generalizable without excessive complexity [17]. Previous methodologies have largely overlooked the systematic exploration of how performance stability is affected by various combinations of preprocessing techniques.

To address this gap, this study proposes a comprehensive model-fitting analysis for stroke prediction using the Random Forest algorithm. The research design follows the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework, which provides a standard, iterative, and interactive process for discovering knowledge from data [18]. The novelty of this research lies in the systematic evaluation of 48 preprocessing technique combinations, followed by a two-stage hyperparameter optimization using Randomized Search and Grid Search Cross-Validation [19, 20]. This approach aims to identify the most stable technical configuration that yields a ”good fitting” model, ensuring reliability for implementation as an early warning system for healthcare providers [21, 22].

Table 1. Model Performance

Case Study	Algorithm	Precission	Recall	F1-Score	Accuracy
Stroke Detection [2]	Logistic Regression + SMOTE	0.88	0.87	0.87	0.87
Stroke Detection [3]	SVM + SMOTE	-	-	-	0.85
Stroke Detection [4]	Random Forest	0.94	0.94	0.94	0.95
Diabetes Detection [8]	Random Forest + SMOTE	0.97(0) 0.98(1)	0.97(0) 0.98(1)	0.97(0) 0.97(1)	0.97
Car Evaluation [9]	Random Forest + Attribute Normalization	-	-	-	0.99
Stroke Detection [10]	ADABoost + Random Forest + KNNImputer + SMOTETomek	0.97 (0) 0.96 (1)	0.96 (0) 0.96 (1)	0.96 (0) 0.96 (1)	0.96
Diabetes Detection [12]	Random Forest + Feature Selection	-	-	-	0.99
Stroke Detection [11]	Random Forest + SMOTETomek	0.97 (0) 0.94 (1)	0.94 (0) 0.97 (1)	0.96 (0) 0.96 (1)	0.96

Based on the literature review, Random Forest consistently demonstrates superior performance for stroke classification tasks. Therefore, this study aims to evaluate the performance of a stroke risk prediction model to ensure its reliability and trustworthiness. The study seeks to answer the following research questions: How does the performance of a stroke risk prediction model using Random Forest and various preprocessing techniques compare? Which preprocessing combination provides the best performance and model stability? Can the resulting model be implemented as an efficient and trustworthy early warning decision support system for stroke? To address these questions, this study evaluates the performance of Random Forests for stroke risk prediction and conducts model-fitting analyses using cross-validation and hyperparameter tuning. To prevent overfitting or underfitting, data preparation optimization and hyperparameter tuning using Randomized Search and Grid Search Cross-Validation are applied. The findings are expected to support the integration of artificial intelligence into the development of early warning systems for stroke risk, ultimately reducing mortality and preventing stroke occurrence.

2. RESEARCH METHOD

To address the research questions, this study proposes a model-fitting analysis for stroke prediction models based on the Random Forest algorithm. The research design follows the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework, an iterative and interactive model consisting of six stages aimed at discovering knowledge from data [16, 18]. The research design is visually illustrated in Figure 1.

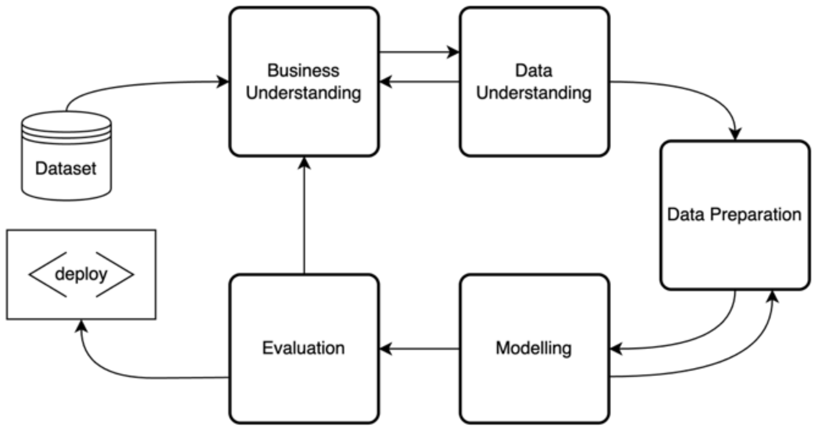


Figure 1. CRISP-DM

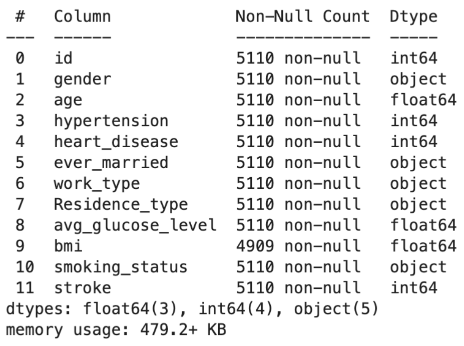
2.1. Business Understanding

The Business Understanding phase focuses on defining project objectives from a business perspective and translating them into data mining problems and initial project plans [16]. Accordingly, the main objective of this study is to identify the optimal model by optimizing preprocessing and hyperparameter tuning for stroke prediction. This model not only achieves high accuracy but also builds a Stroke Early Warning Decision Support System that is highly trustworthy. This is crucial because overfitting in medical models can lead to misdiagnoses in new patients, with fatal consequences for clinical management.

2.2. Data Understanding

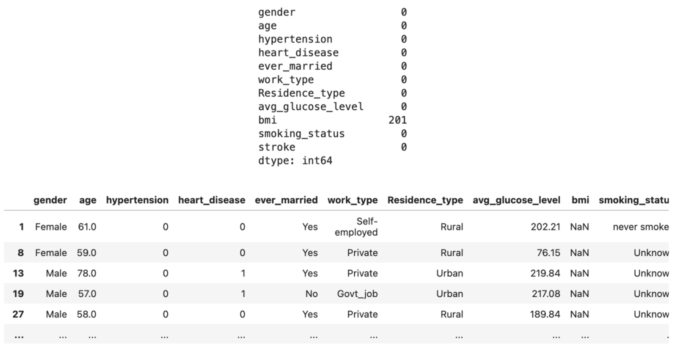
The Data Understanding phase begins with data collection and continues with activities to explore data characteristics, identify data quality issues, and gain initial insights [16]. This phase corresponds to Exploratory Data Analysis (EDA), which helps understand data in depth, detect errors, identify variables, and analyze correlations without statistical inference or modeling [18, 19]. This study uses a public dataset, as public datasets provide standardized benchmarks for machine learning model evaluation and comparison [20]. The dataset used is the "Stroke Prediction Dataset." [21], which has also been used in previous studies [2–4, 10, 11] and is accessible via Kaggle (https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/data).

The dataset consists of 5,110 records and 12 attributes, including three float64 attributes, four int64 attributes, and five object-type attributes, as shown in Figure 2. The 'bmi' attribute contains 201 missing values, which can negatively affect prediction performance (Figure 3). The dataset is also highly imbalanced, with 4,861 instances in class 0 (non-stroke) and 249 instances in class 1 (stroke), as shown in Figure 4. So that at this stage, several conditions were found in the dataset in the form of: (1) There is a missing value in the 'BMI' Feature as many as 201 data, (2) There is an extreme imbalance between stroke and non-stroke classes, (3) Different value ranges for each numeric feature, and (4) There are categorical features that cannot be processed directly by the random forest



```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

Figure 2. Stroke Dataset Description



```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                201
smoking_status       0
stroke               0
dtype: int64
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_statu
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoke
8	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknow
13	Male	78.0	0	1	Yes	Private	Urban	219.84	NaN	Unknow
19	Male	57.0	0	1	No	Govt_job	Urban	217.08	NaN	Unknow
27	Male	58.0	0	0	Yes	Private	Rural	189.84	NaN	Unknow
...	...	...	...	...	...	...	...	...	...	.

Figure 3. Identifying Missing Value

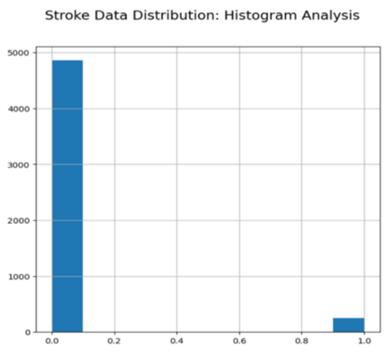


Figure 4. Imbalanced data

2.3. Data Preparation/Preprocessing

Based on the previously conducted data analysis, it is necessary to preprocess the data to identify which parameters best determine the optimal model. The overall data preparation workflow is illustrated in Figure 5. Broadly speaking, the preprocessing data flow begins with the existing dataset, then removes features that are not present in it; the omitted feature is 'ID'. Next, determine what features are included in the independent variable (X) or dependent variable (y). X consists of features: 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'avg_glucose_level', 'bmi', 'smoking_status'. While y consists of: 'stroke'. The next step is to split the data to mitigate the risk of data leakage; a detailed explanation is provided in subchapter 2.3.5. After the data are split, preprocessing is carried out as described in subsections 2.3.1-2.3.3. Subsequently, the data balancing described in subchapter 2.3.4 is performed. In the data transformation section, missing value imputation, scaling, and balancing data, a combination is carried out for each type of parameter, which is explained in sub-chapter 2.3.6. The output of each preprocessing stage is a preprocessed training dataset, a test dataset, and a preprocessing model file (*.pkl).

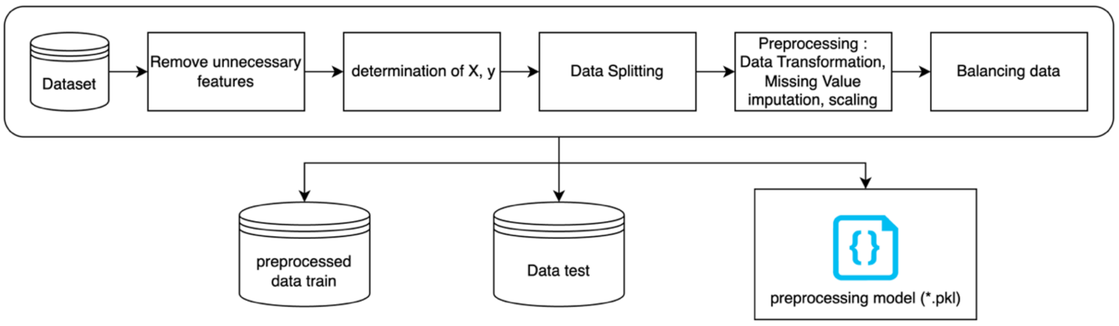

Dataset
Remove unnecessary features
determination of X, y
Data Splitting
Preprocessing : Data Transformation, Missing Value imputation, scaling
Balancing data
preprocessed data train
Data test
preprocessing model (*.pkl)

Figure 5. Data preparation schema

2.3.1 Missing Value handling on 'BMI' feature

The experiment on handling missing values in the 'BMI' feature was carried out by filling in the blank values [11] by using simple imputation techniques in the form of mean and median [23]. In addition, it will also be compared with the K-Nearest Neighbor-based imputation technique in the form of KNNImputer, which is proven to provide more accurate imputation and improve model accuracy performancel [10] and able to reduce the standard deviation of the Random Forest model, thereby making the model more stable and consistent in its predictions.

2.3.2 Handling value range differences in numeric features

Handling of this range of value differences is necessary because, in order to prevent the dominance of a feature, the model will tend to follow features that have a large number or range [9, 23]. In addition, it can also avoid data inconsistencies [9]. To address this difference in value ranges, a comparative experiment will be conducted between the MinMaxScaler technique and handlers using the StandardScaler technique. The use of the StandardScaler Technique because it is able to map each feature data point into a normal distribution [23]. Whereas MinMaxScaler projects features into a specific range. It is empirically proven to improve the accuracy level of the Random Forest model, helping the algorithm to perform the learning process better, resulting in a more stable model and having a strong generalization ability to new data [9]. As well as conformity with the characteristics of the dataset.

2.3.3 Categorical feature handling (data transformation)

The Random Forest algorithm cannot process data in nominal format directly, so it must be converted into numeric form [9, 11, 12]. Beyond mathematical needs, this also helps avoid inconsistencies (noise). In addition, proper encoding techniques help reduce the computational load that arises due to the number of non-linearly growing splits on high-cardinality categorical variables [7]. The handling of this categorical feature will be evaluated using OneHotEncoder and OrdinalEncoder.

2.3.4 Extreme imbalances between stroke and non-stroke classes handling

The experiment on handling class imbalances was carried out using SMOTE, TomekLinks, SMOTETomek, and Undersampling Based on Clustering techniques. The SMOTE technique works by generating new synthetic data in a feature space that is among existing minority class data, rather than simply randomly duplicating the data [10], thus improving the performance of the algorithm in the minority class and reducing the risk of overfitting that often occurs in the majority class [8]. Tomeklinks is one of the undersampling techniques that works by removing samples from the majority class that are very close to the minority class [10], So that it has the ability as a data cleaning method that clarifies the boundary between two classes (Decision Boundary) [11]. SMOTETomek is a hybrid approach that combines oversampling (SMOTE) to increase the number of minority classes and undersampling (Tomek) to clean up overlapping samples [10, 11]. The advantage is that it produces a balanced dataset with well-defined class clusters, thereby significantly improving the model's accuracy. Undersampling Based on Clustering (ClusterCentroid) works by replacing a set of data instances from the majority class that have certain similarities to their centroid clusters [23]. It has an advantage over other random undersampling techniques in that it reduces the data volume without losing important structural information, because the retrieved instances still represent the characteristics of the original data group.

2.3.5 Avoiding the risk of data leakage

To reduce the risk of data leakage, where the model is trained using information that should not be available during testing [23]. If the same data is used for training and testing, the evaluation results will be biased and do not reflect the model's performance in the real world [3, 20]. The preprocessing steps included removing irrelevant features, such as the ID attribute, and splitting the dataset into training (80%) and test (20%) subsets. Separation with an 80:20 ratio is seen as the optimal balance to provide the algorithm with a sufficient amount of data for valid statistical testing [3, 6]. Subsequently, combinations of data transformation techniques, missing-value handling methods, and scaling techniques were applied, followed by imbalanced-data handling applied only to the training data. This strategy was implemented to prevent any overlap or intersection between the training and testing datasets, thereby ensuring the integrity of the evaluation process [23]. Each type of data transformation was paired with each missing-value handling method and each imbalanced-data handling technique.

2.3.6 Combination of preprocessing types

Because each feature handler employs various techniques, this study proposes combining preprocessing techniques by pairing each handling technique from one feature with all handling techniques on other features. The techniques used at each preprocessing stage are presented in Table 2 and illustrated in Figure 6. As a result, 48 distinct preprocessing combinations were generated. Examples of these combinations are shown in Table 3. Consequently, 48 preprocessed training datasets were produced and prepared for model training, with examples shown in Table 4. Additionally, 48 preprocessing models were saved in *.pkl format and later used to preprocess the test data during the testing phase.

Table 2. Types of preprocessing handling

Preprocessing type	Options
Data Tranformation	One-Hot Encoding, Ordinal Encoding
Missing Value	Mean, median, KNNImputer
Scaling	StandardScaler, MinMaxScaler
Imbalanced Data	SMOTE, TomekLinks, ClusterCentroid, SMOTETomek

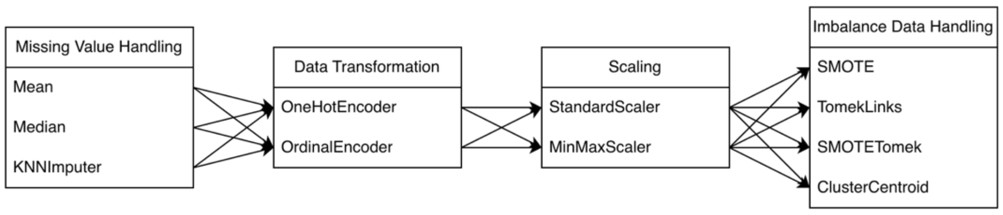


Figure 6. Preprocessing handler combinations

Table 3. Combination of Handling Types

Data Transfromation	Missing Value	Scaling	Imbalanced data
One-Hot Encoding	Mean	StandardScaler	oversampling (SMOTE)
One-Hot Encoding	Mean	MinMaxScaler	Undersampling (TomekLinks)
Ordinal Encode	Mean	MinMaxScaler	Undersampling Based on Clustering
Ordinal Encode	Median	MinMaxScaler	Hybrid sampling (SMOTETomek)
...	...	...	...

Table 4. Example of Preprocessed data train

Age	Gender_male	Gender_female	BMI	...	smoking_status_never_smoked	smoking_status_formerly_smoked
0.584960	0.0	0.0	0.0646129687788	...	0.0	1.0
0.353027	0.0	0.0	0.1337982634398	...	0.0	1.0
0.42626	0.0	0.0	0.2964160354701	...	0.0	1.0
...	...	...	...	...	...	...

2.4. Modeling

The modeling phase in this study focused on the use of the Random Forest algorithm, as it has been demonstrated in the majority of previous studies that Random Forest achieves high accuracy in stroke prediction tasks [4, 10, 11], making it a promising candidate for further evaluation. Hyperparameter tuning using Randomized Search and Grid Search Cross-Validation was performed on this algorithm to obtain optimal parameter values [16]. The preprocessed training datasets generated during the data preparation stage were used to train the Random Forest algorithm using two hyperparameter tuning methods: Randomized Search and Grid Search. The application of these two hyperparameter tuning methods aimed to enhance the search for optimal parameter configurations while maintaining computational efficiency. RandomizedSearch is well-suited for broad exploration to efficiently identify the best parameter regions, whereas GridSearch, with the refine_range function, is used to refine the best value by searching within $\pm 25\%$ of the best value identified by RandomizedSearch. The steps involved in training the model are illustrated in Figure 7.

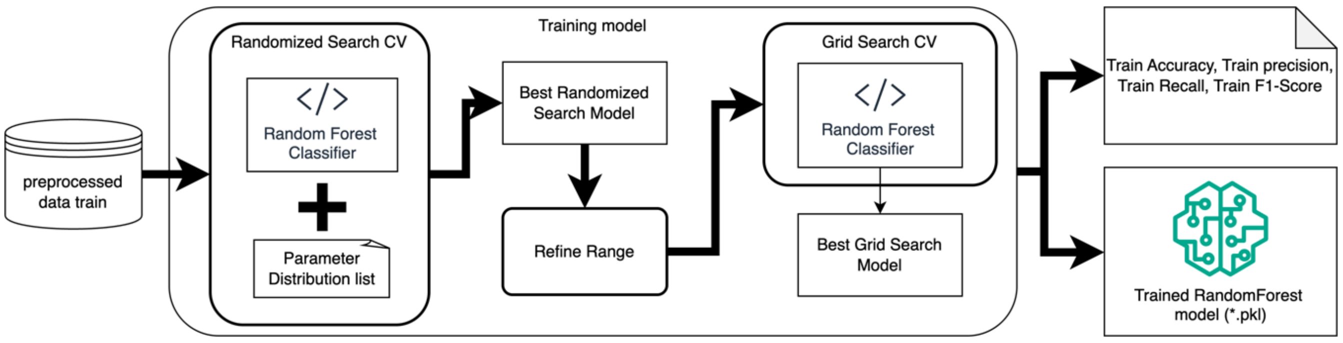


Figure 7. Training model steps

From each preprocessed training dataset generated in the preprocessing stage, a RandomForestClassifier model is trained using the distribution list parameter (Table 5) via the RandomizedSearch hyperparameter optimization method to produce the best model, called 'Best Randomized Search Model', as described in subchapter 2.4.1. Next, we refine the parameters using the 'refine_range' function and retrain using grid search, as described in sub-chapter 2.4.2. This stage produces the best models called 'Trained RandomForest models' with a *.pkl file format, as well as a matrix of several performance measures: train accuracy, train precision, train recall, and train F1-Score.

2.4.1 Train model using RandomizedSearch

In the first model training using RandomizedSearch, several parameters were set, such as: 'n_estimators' to determine the increase in the number of trees in the forest in order to improve the stability and predictive power of the model [7], however, theoretically, the level of generalization error in the Random Forest will converge to a certain extent as the number of trees increases, so a range of up to 900 needs to be tested to ensure the model reaches that convergence point [7]; The 'max_depth' parameter determines the depth of the tree, it determines the complexity of the model. Trees that are too deep are at risk of overfitting because they capture data noise, while trees that are too shallow are at risk of underfitting [15]. The 'None' value allows the tree to grow to the entire pure leaves, while a limit of up to 50 is used to control model variants [7, 15]. The 'min_samples_split' and 'min_samples_leaf' parameters are pre-pruning techniques to prevent overly specific tree formation. Determining the minimum sample count on a leaf helps balance the model's performance and its ability to be interpreted, and ensures that each decision rule is supported by a sufficient portion of data [15]; The parameter 'max_features', by default, the value used is 'sqrt', which is the square root of the total number of predictors. The use of 'sqrt' and 'log2' helps to create variation between trees (reducing correlations between individual trees), which is a key advantage of Random Forest in improving the accuracy of ensemble methods [7]. Given the unbalanced class, the 'class_weight' parameter with a value of 'balanced' serves to give greater weight to the minority class (stroke patients), thereby minimizing the risk of model bias towards the majority class and improving the ability to detect early detection in rare medical cases [6, 7], in this study it will be compared between 'None' and 'balanced'. The hyperparameters involved in the tuning process include n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, and class_weight. The parameter settings used specifically during the Randomized Search process are presented in Table 5.

Table 5. Parameter settings

Parameter	Sets
‘n_estimators’	[100, 200, 300, 400, . . ., 800, 900]
’max_depth’	[None, 10, 15, 20, 25, 30, 40, 50]
’min_samples_split’	[2, 5, 10, 20]
’min_samples_leaf’	[1, 2, 4, 8]
’max_features’	[’sqrt’, ’log2’, None]
’class_weight’	[’balanced’, None]

2.4.2 Train model using GridSearch

For the Grid Search process, a ‘refine_range’ function was employed to narrow the search range for numerical parameters around the best values obtained from the Randomized Search results. This function computes the lower and upper bounds by taking $\pm 25\%$ of the best value from the Randomized Search. This approach was shown to improve the Random Forest algorithm’s training performance, as indicated by an increase in the Area Under the Curve (AUC), with partial results presented in Table 6. At this stage, training performance metrics, including training accuracy, training precision, training recall, and training F1-score were obtained, along with trained Random Forest model files. A summary of the AUC improvement achieved through hyperparameter tuning is presented in Table 7.

Table 6. Example of AUC score improvement

Model No	Model file	random_best_auc	grid_best_auc	auc_improvement
1	RF_best_data-balanced-knn_minmax_onehot-cluster_centroids.joblib	0.9158397436	0.9170705128	0.0012307692
2	RF_best_data-balanced-knn_minmax_onehot-over.joblib	0.9912547484	0.9913559706	0.0001012223
3	RF_best_data-balanced-knn_minmax_onehot-smote_tomek.joblib	0.9930573110	0.9930705654	0.0000132543
4	RF_best_data-balanced-knn_minmax_onehot-under.joblib	0.8443204789	0.8452901574	0.0009696785
5	RF_best_data-balanced-knn_minmax_ordinal-cluster_centroids.joblib	0.9120336538	0.9120961538	0.0000625000
6	RF_best_data-balanced-knn_minmax_ordinal-over.joblib	0.9890154415	0.9890786611	0.0000632195
. . .	. . .	. . .	. . .	. . .

Table 7. Recapitulation of the AUC score improvement

Model No	% improvement	Model No	% improvement	Model No	% improvement
1	13.44%	17	0.00%	33	23.44%
2	1.02%	18	0.10%	34	0.66%
3	0.13%	19	0.48%	35	0.41%
4	11.48%	20	3.97%	36	0.00%
5	0.69%	21	27.56%	37	22.23%
6	0.64%	22	0.78%	38	1.16%
7	0.60%	23	0.20%	39	0.91%
8	10.99%	24	8.80%	40	24.98%
9	0.00%	25	3.96%	41	5.31%
10	0.11%	26	0.00%	42	0.29%
11	0.00%	27	0.00%	43	0.29%
12	6.23%	28	38.80%	44	15.49%
13	0.00%	29	9.30%	45	1.43%
14	0.35%	30	0.06%	46	0.32%
15	0.34%	31	0.04%	47	0.35%
16	17.09%	32	16.77%	48	17.94%

2.5. Evaluation

The trained models were evaluated by predicting the classes of the test data, which were assumed to represent new, previously unseen cases. To perform this evaluation, the same preprocessing steps used during training were required. Therefore, the previously saved preprocessing models (*.pkl) were utilized to preprocess the testing data. Next, class predictions for the testing data were generated using the trained Random Forest models (*.pkl). After obtaining the prediction results, evaluation metrics including accuracy, precision, recall, and F1-score were calculated. Based on these performance metrics, the best Random Forest-based model could be identified. The evaluation workflow is clearly illustrated in Figure 8.

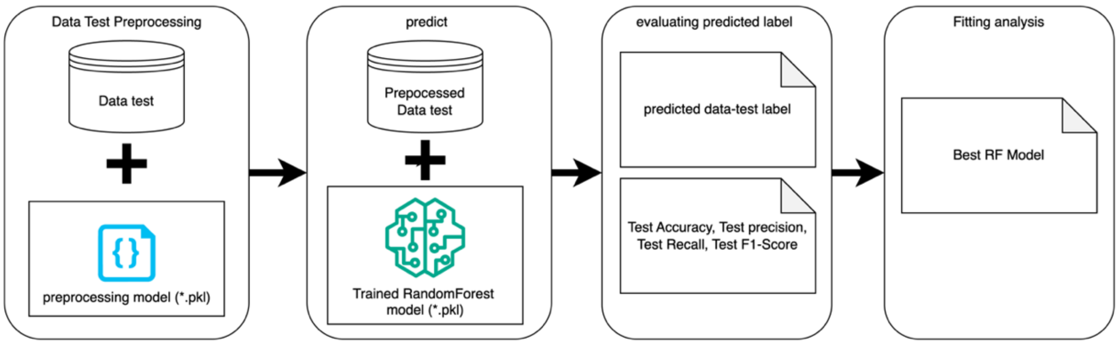

Data Test Preprocessing
Data test
preprocessing model (*.pkl)
predict
Prepocessed Data test
Trained RandomForest model (*.pkl)
evaluating predicted label
predicted data-test label
Test Accuracy, Test precision, Test Recall, Test F1-Score
Fitting analysis
Best RF Model

Figure 8. Model evaluation steps

Using the test data from the preprocessing stage (Chapter 2.3), the preprocessing model (*.pkl) is applied. So that from the preprocessed data test, the class can be predicted using the 'Trained RandomForest model'. This stage produces a 'predicted data-test label' along with a matrix of several performance measures: Test accuracy, Test Precision, Test Recall, and Test F1-Score. Based on the performance matrix obtained, 'Fitting Analysis' was continued to identify the best Random Forest model.

3. RESULT AND ANALYSIS

The results of experiments on 48 combinations of preprocessing techniques showed that model stability was strongly influenced by the method used to handle data imbalance. For example, in Table 8, aggressive oversampling in models 2 and 3 resulted in perfect accuracy (1.0) during training, but performed poorly on the test set, indicating severe overfitting. This study also achieved an accuracy of 95%-96% by integrating two-stage parameter optimization methods (Randomized and Grid Search), which narrowed the search range by ±25% to ensure convergence, with only a small difference between training and testing performance. Performance scores, including accuracy, precision, recall, and F1-score, were obtained. Subsequently, model-fitting analysis was conducted to assess whether the developed models exhibited good fit and strong generalization capability. Three types of model fitting were defined in this study: Overfitting, where training performance is high but testing performance is low; Underfitting, where both training and testing performance are low; and Good Fitting, where training and testing performance are balanced or only slightly different.

To classify the fitting type, several rules were established. If the evaluation performance was lower than the training performance with a gap of less than 0.05, the model was categorized as a Good Fit. If the performance gap was between 0.05 and 0.15, the model was classified as Moderate Fit (mild overfitting). If the gap exceeded 0.15, the model was considered Overfitting. Additionally, if the training and/or evaluation performance was below 0.7, the model was classified as Underfitting. Of the 48 preprocessing combinations generated, each was used to train a Random Forest model with the optimal parameters obtained via two hyperparameter tuning methods (Randomized Search and Grid Search), and the resulting models were subsequently evaluated. As a result, this study produced 48 Random Forest-based models, ready for model fitting. Model-fitting analysis was conducted for each performance metric (accuracy, precision, recall, and F1-score), yielding 15 models that exhibited good fit across at least one metric. These results are presented in Table 10. The detailed training and testing performance metrics for accuracy, precision, recall, and F1-score are presented in Table 9. Meanwhile, the results of the fitting-type classification are presented in Table 10.

Table 8. Accuracy, precision, recall, and F1-Score results

Model No	Train / Testing	Accuracy	Precision	Recall	F1
1	Train	0.9221105527638191	0.9242424242424242	0.9195979899497487	0.9219143576826196
	Test	0.5557729941291585	0.08775510204081632	0.86	0.15925925925925924
2	Train	1	1	1	1
	Test	0.9090019569471625	0.1791044776119403	0.24	0.20512820512820515
3	Train	1	1	1	1
	Test	0.9041095890410958	0.14705882352941177	0.2	0.16949152542372883
4	Train	0.8577532043226942	0.2510176390773406	0.9296482412060302	0.39529914529914534
	Test	0.8346379647749511	0.19487179487179487	0.76	0.3102040816326531
...	Train	...	...	...	...
	Test	...	...	...	...

Model No	Train / Testing	Accuracy	Precision	Recall	F1
48	Train	0.8828105395232121	0.28996865203761757	0.9296482412060302	0.4420549581839905
	Test	0.8463796477495108	0.1907514450867052	0.66	0.2959641255605381

Table 9. Good Fit Accuracy, precision, recall, and F1-Score results

Model No	Train / Testing	Accuracy	Precision	Recall	F1
4	Train	0.8577532043226942	0.2510176390773406	0.9296482412060302	0.39529914529914534
	Test	0.8346379647749511	0.19487179487179487	0.76	0.3102040816326531
8	Train	0.9256094496104549	0.39789473684210525	0.949748743718593	0.5608308605341246
	Test	0.8776908023483366	0.2047244094488189	0.52	0.2937853107344633
12	Train	0.8663490471414242	0.2621082621082621	0.9246231155778895	0.4084350721420643
	Test	0.8317025440313112	0.17894736842105263	0.68	0.2833333333333333
16	Train	0.9286250939143501	0.40772532188841204	0.9547738693467337	0.5714285714285715
	Test	0.8806262230919765	0.20967741935483872	0.52	0.29885057471264365
20	Train	0.9499748617395676	0.0	0.0	0.0
	Test	0.9510763209393346	0.0	0.0	0.0
24	Train	0.8891402714932126	0.3016393442622951	0.9246231155778895	0.45488257107540175
	Test	0.8542074363992173	0.19631901840490798	0.64	0.3004694835680751
28	Train	0.8696634856855852	0.26878612716763006	0.9346733668341709	0.4175084175084176
	Test	0.8365949119373777	0.18716577540106952	0.7	0.29535864978902954
32	Train	0.8846539618856569	0.29448818897637796	0.9396984924623115	0.44844124700239807
	Test	0.8493150684931506	0.19411764705882353	0.66	0.3
40	Train	0.8826338275948731	0.290625	0.9346733668341709	0.4433849821215733
	Test	0.8522504892367906	0.19760479041916168	0.66	0.30414746543778803
44	Train	0.8694451418528747	0.2663755458515284	0.9195979899497487	0.41309255079006774
	Test	0.8365949119373777	0.18032786885245902	0.66	0.2832618025751073
48	Train	0.8828105395232121	0.28996865203761757	0.9296482412060302	0.4420549581839905
	Test	0.8463796477495108	0.1907514450867052	0.66	0.2959641255605381
33	Train	0.9422110552763819	0.9583333333333334	0.9246231155778895	0.9411764705882352
	Test	0.5342465753424658	0.0872093023255814	0.9	0.15901060070671377
26	Train	1.0	1.0	1.0	1.0
	Test	0.9315068493150684	0.16666666666666666	0.1	125
43	Train	1.0	1.0	1.0	1.0
	Test	0.9315068493150684	0.16666666666666666	0.1	125
10	Train	1.0	1.0	1.0	1.0
	Test	0.9315068493150684	0.16666666666666666	0.1	125

Table 10. Good-fitting model results

Model No	Acc gap	Prec gap	Recall gap	F1 gap
4	0,02311524	0,05614584	0,16964824	0,08509506
	Good Fit	Moderate Fit	Overfitting	Moderate Fit
8	0,04791865	0,19317033	0,42974874	0,26704555
	Good Fit	Overfitting	Overfitting	Overfitting
12	0,0346465	0,08316089	0,24462312	0,12510174
	Good Fit	Moderate Fit	Overfitting	Moderate Fit
16	0,04799887	0,1980479	0,43477387	0,272578
	Good Fit	Overfitting	Overfitting	Overfitting
20	0,0011015	0	0	0
	Good Fit	Good Fit	Good Fit	Good Fit
24	0,03493284	0,10532033	0,28462312	0,15441309
	Good Fit	Moderate Fit	Overfitting	Overfitting
28	0,03306857	0,08162035	0,23467337	0,12214977
	Good Fit	Moderate Fit	Overfitting	Moderate Fit
32	0,03533889	0,10037054	0,27969849	0,14844125
	Good Fit	Moderate Fit	Overfitting	Moderate Fit
40	0,03038334	0,09302021	0,27467337	0,13923752
	Good Fit	Moderate Fit	Overfitting	Moderate Fit

Model No	Acc gap	Prec gap	Recall gap	F1 gap
44	0,03285023	0,08604768	0,25959799	0,12983075
	Good Fit	Moderate Fit	Overfitting	Moderate Fit
48	0,03643089	0,09921721	0,26964824	0,14609083
	Good Fit	Moderate Fit	Overfitting	Moderate Fit
33	0,40796448	0,87112403	0,02462312	0,78216587
	Overfitting	Overfitting	Good Fit	Overfitting
26	0,06849315	0,83333333	0,9	-124
	Overfitting	Overfitting	Good Fit	Overfitting
43	0,06849315	0,83333333	0,9	-124
	Moderate Fit	Overfitting	Overfitting	Good Fit
10	0,068493151	0,833333333	0,9	-124
	Moderate Fit	Overfitting	Overfitting	Good Fit

Table 11. Combination of preprocessing and hyperparameter tuning

Model No	Preprocessing Handling Method				Grid Search Best Parameter
	Transformation	Missing Value	Scallling	Balancing data	
4	OneHot Encoder	KNNImputer	MinMax Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 11, ’max_features’: ’sqrt’, ’min_samples_leaf’: 12, ’min_samples_split’: 2, ’n_estimators’: 400}
8	Ordinal Encoder	KNNImputer	MinMax Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 22, ’max_features’: ’log2’, ’min_samples_leaf’: 8, ’min_samples_split’: 5, ’n_estimators’: 300}
12	OneHot Encoder	KNNImputer	Standart Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 30, ’max_features’: ’sqrt’, ’min_samples_leaf’: 12, ’min_samples_split’: 10, ’n_estimators’: 600}
16	Ordinal Encoder	KNNImputer	Standart Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 15, ’max_features’: ’sqrt’, ’min_samples_leaf’: 8, ’min_samples_split’: 2, ’n_estimators’: 300}
24	Ordinal Encoder	Mean	MinMax Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 30, ’max_features’: ’sqrt’, ’min_samples_leaf’: 8, ’min_samples_split’: 30, ’n_estimators’: 600}
28	OneHot Encoder	Mean	Standart Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 18, ’max_features’: ’sqrt’, ’min_samples_leaf’: 12, ’min_samples_split’: 2, ’n_estimators’: 300}
32	OneHot Encoder	Median	MinMax Scaler	Cluster Centroids	{’class_weight’: ’balanced’, ’max_depth’: 22, ’max_features’: ’log2’, ’min_samples_leaf’: 12, ’min_samples_split’: 5, ’n_estimators’: 400}
40	Ordinal Encoder	Median	MinMax Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: 30, ’max_features’: ’sqrt’, ’min_samples_leaf’: 12, ’min_samples_split’: 10, ’n_estimators’: 1000}
44	OneHot Encoder	Median	Standart Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: None, ’max_features’: ’sqrt’, ’min_samples_leaf’: 12, ’min_samples_split’: 5, ’n_estimators’: 625}
48	Ordinal Encoder	Median	Standart Scaler	TomekLinks	{’class_weight’: ’balanced’, ’max_depth’: None, ’max_features’: ’sqrt’, ’min_samples_leaf’: 12, ’min_samples_split’: 5, ’n_estimators’: 375}
33	OneHot Encoder	Median	MinMax Scaler	Cluster Centroids	{’class_weight’: ’balanced’, ’max_depth’: 7, ’max_features’: ’log2’, ’min_samples_leaf’: 1, ’min_samples_split’: 10, ’n_estimators’: 600}

4. CONCLUSION

Based on 48 experimental trials across different combinations of preprocessing techniques, 15 models met the criterion for good fit. Across these 48 models, performance was improved by applying a combination of Randomized Search and Grid Search for hyperparameter tuning. The three highest performance improvements were observed for model numbers 28 (38.80%), 21 (27.56%), and 40 (24.98%). The best model was selected based on the highest accuracy and the satisfaction of the goodness-of-fit criteria for at least one performance metric, namely accuracy, precision, recall, or F1-score. However, based on the results presented in Tables

9 and 10, several models exhibited anomalous behavior. For instance, model number 20 showed zero values for precision, recall, and F1-score in both training and testing phases. In addition, models 26, 43, and 10 produced F1-score values exceeding 1.0 (up to 125) during testing, indicating invalid evaluation results. Anomalies of F1-score values above 1.0 (as in models 26, 43, and 10) are identified as evaluation validity failures. Technically, this occurs when standard scaling is combined with highly imbalanced data, where the precision or recall for the minority class is close to zero but the denominator in the system's evaluation algorithm is unstable. Consequently, these models (models 20, 26, 43, and 10) were excluded from the selection of the best model.

From these findings, it can be concluded that the preprocessing technique with the greatest impact on the Random Forest algorithm's prediction of stroke risk is the handling of imbalanced data. As shown in Tables 10 and 11, the use of SMOTE-Tomek (hybrid sampling) did not yield models that met the good-fitting criteria. The best-performing models (characterized by high accuracy and good fitting) were identified as model number 33, which achieved the highest training accuracy of 0.942 with a testing accuracy of 0.534, and model number 8, which achieved a training accuracy of 0.925 and a testing accuracy of 0.877, representing the best testing accuracy performance. The analysis showed that KNNImputer yields more accurate estimates for complex medical datasets because it uses the k-nearest neighbors approach, rather than simply averaging values. The use of TomekLinks as an undersampling technique clarifies the decision boundary between classes by removing overlapping majority samples, thereby minimizing the performance gap to 0.047 (Good Fit).

Therefore, model 8 was selected as the most suitable for implementation as a reliable and efficient early-warning decision support system for stroke prediction, as it showed the smallest difference between training and testing accuracy (0.04791865). According to Table 8, Model 8 also met the good-fitting criteria based on accuracy. This is consistent with the study's purpose of assessing reliability: models with a test accuracy of 0.877 (Model 8) are considered more feasible than models with a training accuracy of 1.0, despite the latter's lower overfitting. This research not only produces high-accuracy models but also emphasizes reliability through goodness-of-fit analysis. In a medical context, stable models (such as Model 8 with a gap of 0.047) are more valuable to clinicians because they provide assurance that the system does not produce misleading diagnostic results when handling new patients (unseen data), making them feasible to integrate into hospital early warning systems. A limitation of this study is that the selection of the best model was based solely on accuracy metrics and goodness-of-fit analysis. For future research, alternative model selection methods may be considered to identify models that better align with specific business or organizational requirements, particularly with respect to interpretability in the medical field. In addition, the use of larger and more diverse datasets should be considered to further improve model robustness and generalizability. These limitations could become suggestions for further study.

ACKNOWLEDGEMENTS

The authors would like to express their sincere gratitude to the Institute for Research and Community Service (LPPM) of Universitas Nusantara PGRI Kediri for the 2025 research grant, which supported the successful completion of this research.

REFERENCES

[1] J. Liljehult, T. Christensen, and K. B. Christensen, "Early Prediction of One-Year Mortality in Ischemic and Haemorrhagic Stroke," en, *Journal of Stroke and Cerebrovascular Diseases*, vol. 29, no. 4, p. 104 667, Apr. 2020. DOI: 10.1016/j.jstrokecerebrovasdis.2020.104667

[2] M. Guhdar, A. Ismail Melhum, and A. Luqman Ibrahim, "Optimizing Accuracy of Stroke Prediction Using Logistic Regression," *Journal of Technology and Informatics (JoTI)*, vol. 4, no. 2, pp. 41–47, Jan. 19, 2023. DOI: 10.37802/joti.v4i2.278

[3] E. Wulandari and A. Witanti, "The Classification of Stroke Prediction Using the Support Vector Machine (SVM) Method," en-US, *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 11, no. 3, pp. 17–29, Sep. 18, 2024. DOI: 10.35957/jatisi.v11i3.8044

[4] T. K. Amarya et al., "Analisa Perbandingan Algoritma Classification Berdasarkan Komposisi Label," en, *Prosiding SEMNAS INOTEK (Seminar Nasional Inovasi Teknologi)*, vol. 8, no. 1, pp. 32–40, Jul. 27, 2024. DOI: 10.29407/inotek.v8i1.4906

[5] W. Wang et al., "A systematic review of machine learning models for predicting outcomes of stroke with structured data," en, *PLOS ONE*, vol. 15, no. 6, O. Beiki, Ed., e0234722, Jun. 12, 2020. DOI: 10.1371/journal.pone.0234722

[6] X. Huang et al., "Novel Insights on Establishing Machine Learning-Based Stroke Prediction Models Among Hypertensive Adults," *Frontiers in Cardiovascular Medicine*, vol. 9, p. 901 240, May 6, 2022. DOI: 10.3389/fcvm.2022.901240

[7] T. Zhu, “Analysis on the Applicability of the Random Forest,” *Journal of Physics: Conference Series*, vol. 1607, no. 1, p. 012 123, Aug. 1, 2020. DOI: 10.1088/1742-6596/1607/1/012123

[8] A. Nugroho and D. Harini, “Teknik Random Forest untuk Meningkatan Akurasi Data Tidak Seimbang,” *JSITIK: Jurnal Sistem Informasi dan Teknologi Informasi Komputer*, vol. 2, no. 2, pp. 128–140, Jun. 1, 2024. DOI: 10.53624/jsitik.v2i2.379

[9] A. Nugroho and A. Husin, “Performance Analysis of Random Forest Using Attribute Normalization,” *Sistemasi*, vol. 11, no. 1, p. 186, Jan. 9, 2022. DOI: 10.32520/stmsi.v11i1.1681

[10] A. Ristyawan, A. Nugroho, and T. K. Amarya, “Optimasi Preprocessing Model Random Forest untuk Prediksi Stroke,” *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 12, no. 1, Mar. 10, 2025. DOI: 10.35957/jatisi.v12i1.9587

[11] T. K. Amarya, A. Ristyawan, and R. Firliana, “Optimization of Random Forest Algorithm Performance for Early Detection of Stroke Disease Using Medical Record Data,” *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 10, no. 3, pp. 832–840, Jul. 4, 2025. DOI: 10.30591/jpit.v10i3.8424

[12] A. Nugroho, A. Z. Fanani, and G. F. Shidik, “Evaluation of Feature Selection Using Wrapper For Numeric Dataset With Random Forest Algorithm,” in *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarang, Indonesia: IEEE, Sep. 18, 2021, pp. 179–183. DOI: 10.1109/iSemantic52711.2021.9573249

[13] H. Li et al., “Keeping Deep Learning Models in Check: A History-Based Approach to Mitigate Overfitting,” *IEEE Access*, vol. 12, pp. 70 676–70 689, 2024. DOI: 10.1109/ACCESS.2024.3402543

[14] C. Xu, P. Coen-Pirani, and X. Jiang, “Empirical Study of Overfitting in Deep Learning for Predicting Breast Cancer Metastasis,” en, *Cancers*, vol. 15, no. 7, p. 1969, Mar. 25, 2023. DOI: 10.3390/cancers15071969

[15] M. Zlobin and V. Bazylevych, “A Data-Driven Approach for Balancing Overfitting and Underfitting in Decision Tree Models,” *Central Ukrainian Scientific Bulletin. Technical Sciences*, vol. 1, pp. 14–26, 11(42) Mar. 31, 2025. DOI: 10.32515/2664-262X.2025.11(42).1.14-26

[16] R. Wirth and J. Hipp, “CRISP-DM: Towards a Standard Process Model for Data Mining,” en, in *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, 2000, pp. 29–39. [Online]. Available: https://www.researchgate.net/publication/239585378_CRISP-DM_Towards_a_standard_process_model_for_data_mining

[17] M. Kaur et al., “Early Stroke Prediction Methods for Prevention of Strokes,” en, *Behavioural Neurology*, vol. 2022, S. Satapathy, Ed., pp. 1–9, Apr. 11, 2022. DOI: 10.1155/2022/7725597

[18] P. N. Kumar and K. V. Kumar, “Systematic Approach to Perform Task Centric Exploratory Data Analysis with Case study,” *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 3, pp. 1920–1927, Jun. 7, 2021. DOI: 10.30534/ijatcse/2021/601032021

[19] Uma K and M. Hanumanthappa, “Heart Disease Data Analysis Using Exploratory Data Analysis Method,” *International Journal of Engineering Applied Sciences and Technology*, vol. 7, no. 4, pp. 121–126, Aug. 1, 2022. DOI: 10.33564/IJEAST.2022.v07i04.016

[20] C. J. Harrison and C. J. Sidey-Gibbons, “Machine learning in medicine: A practical introduction to natural language processing,” en, *BMC Medical Research Methodology*, vol. 21, no. 1, p. 158, Dec. 2021. DOI: 10.1186/s12874-021-01347-1

[21] Fedesoriano, *Stroke Prediction Dataset*, en, 2021. [Online]. Available: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

[22] U. Shafique and H. Qaiser, “A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA),” English, *International Journal of Innovation and Scientific Research*, vol. 12, no. 1, pp. 217–222, Nov. 20, 2014. [Online]. Available: http://www.ijisr.issr-journals.org/abstract.php?article=IJISR-14-281-04

[23] A. Apicella, F. Isgrò, and R. Prevete, “Don’t push the button! Exploring data leakage risks in machine learning and transfer learning,” en, *Artificial Intelligence Review*, vol. 58, no. 11, p. 339, Aug. 20, 2025. DOI: 10.1007/s10462-025-11326-3

[This page intentionally left blank.]