IJASST International Journal of Applied Sciences and Smart Technologies

Volume 07
Issue 01

p-ISSN 2655-8564, e-ISSN 2685-943, pp. 129-144, 2025

Effect of Hyperparameter Tuning on Performance on Classification model

Muhammad Sholeh*[1], Uning Lestari[1], Dina Andayati[2]

[1] *Informatics Study Program, Faculty of Science and Information Technology, Universitas AKPRIND Indonesia*
[2] *Retail Management Study Program, Faculty of Communication and Business, Universitas AKPRIND Indonesia*
**Corresponding Author: muhash@akprind.ac.id*

(Received 14-02-2025; Revised 05-03-2025; Accepted 06-03-2025)

Abstract

This research aims to analyze the effect of hyperparameter tuning on the performance of Logistic Regression, K-Nearest Neighbours, Support Vector Machine, Decision Tree, Random Forest, Random Forest Classifier, Naive Bayes algorithms. These six algorithms were tested both using hyperparameter tuning and not using hyperparameter tuning. The dataset used in this research is a public dataset, namely the heart datasheet. This datasheet contains information about features related to the diagnosis of heart disease. Hyperparameter tuning is performed using a grid search technique to determine the best combination of hyperparameter values that can improve model accuracy. Performance comparison is done by measuring the accuracy, precision, recall, and F1-score of each algorithm before and after tuning. The research method follows the stages in the Knowledge Discovery in Databases (KDD) methodology. The KDD methodology consists of several stages of data collection, data cleaning to remove errors, data integration from various sources, and data selection and transformation to be ready for analysis. Next, data mining is performed to find patterns or relationships in the data and evaluation and interpretation of the results to ensure their validity. The results show that hyperparameter tuning applied to the six algorithms does not necessarily improve performance. In the algorithm. SVM and decision tree algorithms, the performance results before hyperparameter tuning actually have a higher accuracy value. The performance of algorithms that experienced an increase after hyperparameter tuning was logistic regression and K-Nearest neighbours. The same performance results are generated in the Random Forest and Naive Bayes algorithms. Based on testing the six algorithms and using the heart datasheet, the hyperparameter tuning process does not always result in a better performance value.

Keywords: Algorithm, Datasheet Hyperparameter tuning, Grid search

1 Introduction

The data science process aims to find the best performance of the model used. This data science process is expected to produce the most accurate and effective model in predicting or classifying data from the datasheet processed. In the context of classification, the best performance is the ability of the model to make correct predictions with a minimal error rate, and to generalize well to data that has never been seen before. The goals in finding the best performance include maximizing the benefits that can be obtained from the available data, by minimizing the bias and variance of the model. This process includes selecting the right algorithm, setting hyperparameters, and selecting relevant features [1],[2]. Classification models can be used to make predictions from a datasheet, [3] created a classification model to determine nutritional status and [4] created a classification model to predict whether patients have lung and colon cancer using the CNN (Convolutional Neural Network) algorithm. By finding the best performing model, the data science process can ensure that the results of data analysis provide more accurate and reliable insights for data-driven decision making, and optimize the resources available in the application of the model.

One of the processes used to improve model performance is Hyperparameter tuning [5]. The goal of finding the best performance in the data science process is to produce a model that is most accurate and effective in predicting or classifying the given data. In the context of classification, the best performance is the ability of the model to make correct predictions with a minimal error rate, and to generalize well to data that has never been seen before. One of the goals of finding the best performance is to maximize the benefits that can be obtained from the available data, by minimizing the bias and variance of the model. This process includes selecting the right algorithm, setting hyperparameters, and selecting relevant features [6]. By finding the best performing model, the data science process can ensure that the results of data analysis provide more accurate and reliable insights for data-driven decision making, as well as optimize the resources available in the application of the model.

Some previous studies have shown that hyperparameter tuning can improve the performance of classification algorithms [7], [8]. Research by Ilemobayo [9], applied hyperparameter tuning using the grid search method to improve accuracy. The results showed that with tuning, the accuracy of the model increased significantly compared to without tuning. [10], used hyperparameter tuning in Machine Learning to Predict Student Academic Achievement. The results showed that the comparison of the four hyperparameter techniques had almost the same MAE value. The range of MAE values between techniques is not significant. GridSearchCV is a technique that has the lowest MAE value, but to achieve this value requires a large estimator value and depth and a very small learning_rate.

[11], This research focuses on analyzing the classification model using the Support Vector Machine (SVM) algorithm, with the aim of finding the optimal parameter value through two hyperparameter tuning techniques, namely Grid Search and Random Search. The research was conducted on seven different datasets, with an evaluation based on the accuracy value, memory usage, and the time required to test the validity of the best model from the two techniques.

Similar research has been conducted by [12]. which uses GridSearch to optimize the Random Forest model. In that study, optimization was performed using the GridSearchCV method to improve model accuracy. The evaluation results showed an increase in model performance, with accuracy reaching 89.61%, recall of 90.15%, precision of 88.72%, and F1-Score of 89.43%. This finding confirms that optimization with GridSearchCV can significantly improve model performance, not only in terms of accuracy but also in other evaluation metrics such as recall, precision, and F1-Score.

Research [13], using the Hyperparameter Tuning method implemented using the CRISP-DM framework. The result obtained is a stance detection model using the Support Vector Machine algorithm which has a micro f1-score value of 78.4%. The hyperparameters of the model are C of 10, max_features parameter from the feature extraction process using TF-IDF method of 10000, and using unigram features. It is concluded that the model has better performance with a micro f1-score value that is 6.6% greater than the stance detection model in previous research.

Research [14] uses the AdaBoost method to process science data with a classification model. The process to get a good accuracy value is done using Hyperparameter Optimization. AdaBoost has hyperparameters needed in the classification process, namely learning rate and n_estimator. The hyperparameter process in this study uses RandomSearchCV. RandomSearchCV is a hyperparameter search method for machine learning models that works by randomly trying hyperparameter combinations from a predefined search space. Unlike GridSearchCV, which tries all possible hyperparameter combinations in the search space, RandomSearchCV randomly selects a certain number of combinations to test. The method is applied using cross-validation techniques to ensure the model's performance on various subsets of data.

Another study using RandomSearchCV on the Random Forest algorithm was conducted [15],, the main purpose of hyperparameter tuning conducted in this study was to improve accuracy in the Covid-19 diagnosis process. The process of finding the optimal hyperparameter value was carried out using two methods, namely Grid Search and Random Search. The results show that the Random Forest algorithm can be used to diagnose Covid-19 with an accuracy rate of 94%. Furthermore, the application of hyperparameter tuning proved effective in increasing the accuracy of this algorithm by 2%, thus providing more optimal results to support disease diagnosis.

In addition, research by [16] also shows that SVM with hyperparameter tuning produces much better performance in cancer classification compared to other algorithms such as Decision Tree and KNN. This study revealed that selecting the right kernel and setting the C and gamma parameters can provide a significant increase in performance in data classification. The average accuracy result obtained is 0.91 and the accuracy can be obtained in a short iteration. This research shows that the use of the PSO algorithm in SVM hyperparameter optimization in imbalanced digital image classification works effectively and efficiently in improving the accuracy obtained. Other research that uses the SVM algorithm and the hyperparameter process is carried out [17] with the aim of improving the classification accuracy of heart disease diagnosis, [18] using hyperparameter tuning to Optimize Stunting with Classification models and using KNN, SVM, and Naïve Bayes algorithms.

Meanwhile, in the KNN algorithm, hyperparameter tuning by selecting the right number of nearest neighbors can also improve prediction accuracy. A study by [19], showed that selecting the optimal number of neighbors gave a clear improvement to the classification results. [20], development of K Hyperparameter Optimization in K-Nearest Neighbor Using Particle Swarm Optimization (PSO). The purpose of PSO is to find the best solution by utilizing collective behavior and iteration between particles. [21], Research conducted using the K-Nearest Neighbor (KNN) algorithm to classify breast cancer, with the number of nearest neighbors (k) parameter determined as 7. To improve the performance of the algorithm, Hyperparameter Tuning is performed using the Grid

Search Cross Validation method. The evaluated performance includes accuracy, precision, and recall. The results showed that the KNN classification model achieved an accuracy rate of 83%, precision of 73%, and recall of 89%

From the literature review, it can be concluded that hyperparameter tuning is an important step in improving the performance of classification algorithms. By performing proper tuning, it can achieve better and more accurate performance in real-world applications, including in the classification of cancer datasets.

The research aims to analyze the effect of hyperparameter tuning on the performance of Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, Random Forest Classifier and Naive Bayes algorithms in the classification of heart datasets using grid search techniques. |The research also aims to identify algorithms that have the best performance after hyperparameter tuning, as well as compare the performance of each algorithm before and after hyperparameter tuning.

2 Material and Methods

Data Analysis Technique

The research method used is Knowledge Discovery in Databases (KDD), which is a systematic process for identifying patterns, information, and useful knowledge from large data sets. The KDD stage begins with data selection, where data relevant to the research objectives are collected from various sources. Next, data cleaning is performed to remove noise, duplication, and inconsistencies that may affect the analysis results. The third stage is data transformation, where the data is converted into a format suitable for further analysis processes. After that, data mining is performed, which is the core process of KDD that involves applying algorithms and statistical techniques to extract hidden patterns or information. The final stage is evaluation and interpretation, where the data mining results are assessed to ensure their accuracy and relevance to the research objectives. [22]. The stages of the KDD methodology are presented in Fig. 1

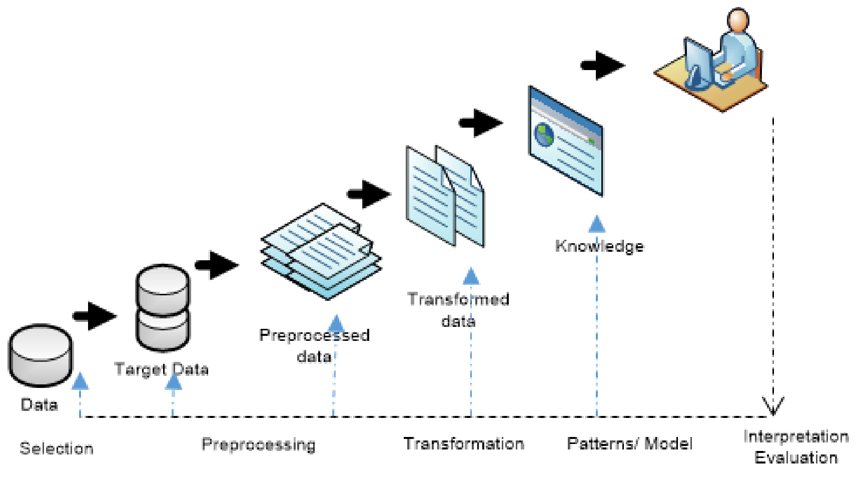


Figure 1. Stages in the Knowledge Discovery method [22]

Datasheet

The processed datasheet is a datasheet obtained from https://archive.ics.uci.edu. The datasheet used is a datasheet that contains data related to heart disease and is stored in a CSV file. The heart disease datasheet can be downloaded at https://archive.ics.uci.edu/dataset/45/heart+disease. The datasheet consists of 309 data and consists of 14 columns.

Hyperparameter Tuning with Grid Search

Hyperparameter tuning is the process of optimizing parameters that are not learned directly by the machine learning model during training, but are determined before the training process begins. The goal of hyperparameter tuning is to find the hyperparameter combination that produces the best model performance.

The hyperparameter tuning method used in this research is Grid Search. Grid Search is a systematic exploration technique where all possible hyperparameter combinations are tested within a predefined range. The process works by creating a "grid" of all combinations of hyperparameter values to be tested, then training the model for each of those combinations and evaluating its performance using metrics such as accuracy, F1-score. .

The main advantage of Grid Search is its ability to find the optimal hyperparameter combination thoroughly, as all possible combinations are tested. However, the disadvantage is that it is computationally expensive, especially when the number of hyperparameters and their value ranges are large, as the number of combinations to be tested increases exponentially. The process of creating a gridsearchCV object is

```
Grid_search = GridSearchCV(
estimator=model, # Model to be tuned
param_grid=param_grid, # Grid hyperparameter c
v=5, # Cross-validation with 5 fold
scoring='accuracy', # Evaluation metric
verbose=1, # Display progress
n_jobs=-1 # Use all CPU cores
)
```

The required parameters are in GridSearchCV are:

- cv=5: Uses 5-fold cross-validation to evaluate each hyperparameter combination.
- scoring='accuracy': Uses accuracy as the evaluation metric.
- verbose=1: Displays the progress during the tuning process.
- n_jobs=-1: Uses all CPU cores to speed up computation.

3 Results and Discussions

Data Selection Stages

In the data selection stage, the dataset used is the Heart Disease Dataset which contains 303 samples with 14 attributes. These attributes include age, gender, chest pain type, blood pressure, cholesterol level, electrocardiogram results, maximum heart rate, and others. This data was chosen because it is relevant for predicting the presence of heart disease based on patient characteristics. The dataset was obtained from the UCI Machine Learning Repository public data repository.

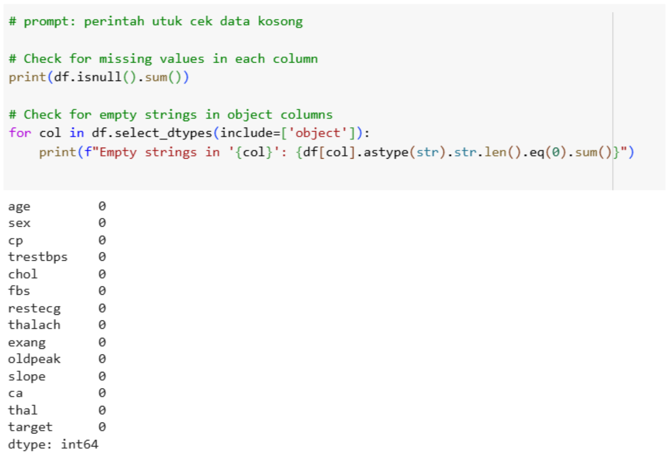
Data Cleaning Stages

Data cleaning stage is performed to remove noise, missing values, and inconsistencies.

(1) Checking for empty data. This process is to ensure that the datasheet has no empty data. Fig. 2, the process for checking empty data. The result of the datasheet checking process is that there is no empty data.

(2) Checking twin data. This process is carried out to check the datasheet of the same data. If there is the same data, delete it. Fig. 3, the process of deleting the same.

(3) Checking outlier data. Outliers are data that have values that are significantly different from the majority of other data in a dataset. The presence of outliers can affect the statistical analysis and performance of data science models, so it is important to detect and handle outliers appropriately. Fig. 4, the process of checking outliers in the datasheet.



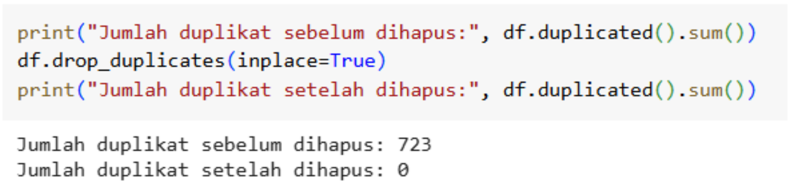
```
# prompt: perintah utuk cek data kosong

# Check for missing values in each column
print(df.isnull().sum())

# Check for empty strings in object columns
for col in df.select_dtypes(include=['object']):
    print(f"Empty strings in '{col}': {df[col].astype(str).str.len().eq(0).sum()}")
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

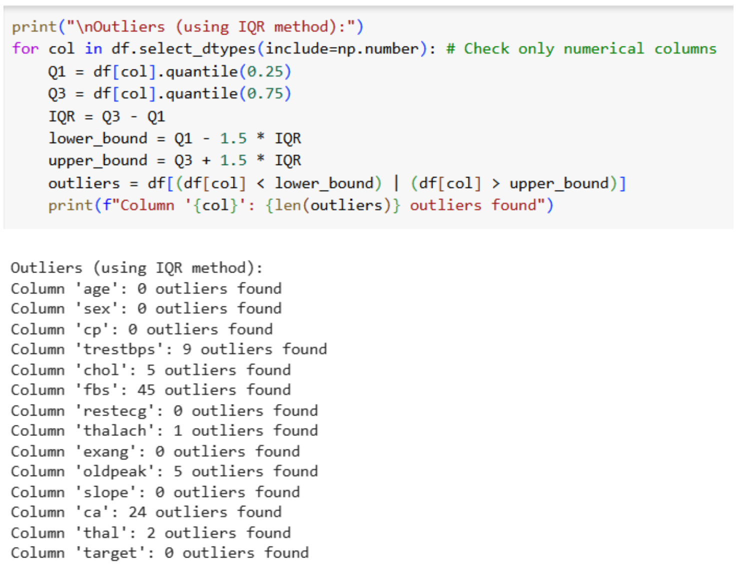
Figure 2. Process for viewing empty data information



```
print("Jumlah duplikat sebelum dihapus:", df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("Jumlah duplikat setelah dihapus:", df.duplicated().sum())
```

```
Jumlah duplikat sebelum dihapus: 723
Jumlah duplikat setelah dihapus: 0
```

Figure 3. The process for viewing empty data information



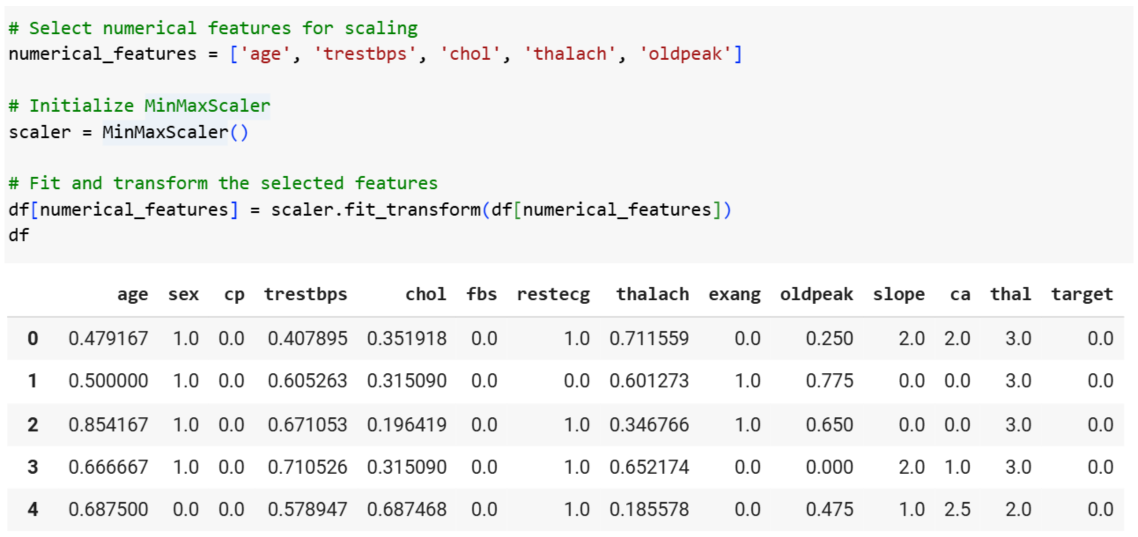
```
print("\nOutliers (using IQR method):")
for col in df.select_dtypes(include=np.number): # Check only numerical columns
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"Column '{col}': {len(outliers)} outliers found")
```

```
Outliers (using IQR method):
Column 'age': 0 outliers found
Column 'sex': 0 outliers found
Column 'cp': 0 outliers found
Column 'trestbps': 9 outliers found
Column 'chol': 5 outliers found
Column 'fbs': 45 outliers found
Column 'restecg': 0 outliers found
Column 'thalach': 1 outliers found
Column 'exang': 0 outliers found
Column 'oldpeak': 5 outliers found
Column 'slope': 0 outliers found
Column 'ca': 24 outliers found
Column 'thal': 2 outliers found
Column 'target': 0 outliers found
```

Figure 4. The process of checking outlier data

Data TransformationStage

In the data transformation stage, several steps are taken to prepare the data for further processing. The process is to normalize numeric attributes such as age, blood pressure (trestbps), cholesterol (chol), thalach and oldpeak using the Min-Max Scaling method so that all attributes have a uniform scale. Fig. 5, Results of the data transformation process on attributes age, trestbps, chol, thalach and oldpeak.



```
# Select numerical features for scaling
numerical_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the selected features
df[numerical_features] = scaler.fit_transform(df[numerical_features])
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0.479167	1.0	0.0	0.407895	0.351918	0.0	1.0	0.711559	0.0	0.250	2.0	2.0	3.0	0.0
1	0.500000	1.0	0.0	0.605263	0.315090	0.0	0.0	0.601273	1.0	0.775	0.0	0.0	3.0	0.0
2	0.854167	1.0	0.0	0.671053	0.196419	0.0	1.0	0.346766	1.0	0.650	0.0	0.0	3.0	0.0
3	0.666667	1.0	0.0	0.710526	0.315090	0.0	1.0	0.652174	0.0	0.000	2.0	1.0	3.0	0.0
4	0.687500	0.0	0.0	0.578947	0.687468	0.0	1.0	0.185578	0.0	0.475	1.0	2.5	2.0	0.0

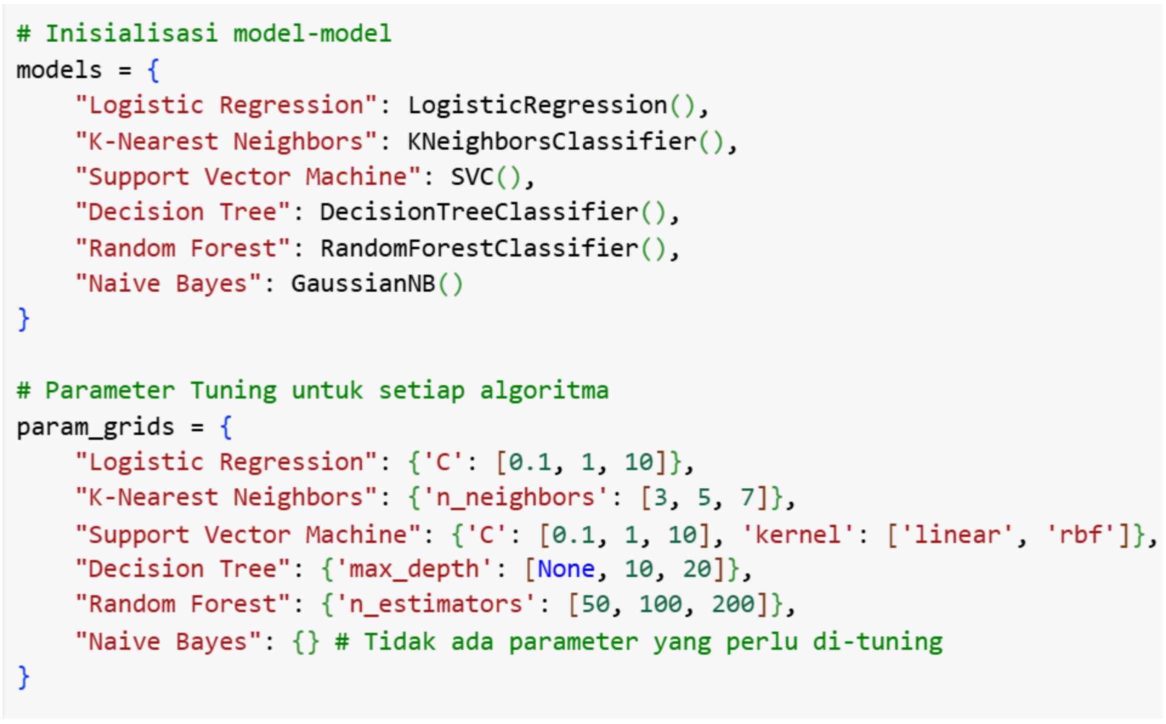
Figure 5. Results of the data transformation process on attributes age, trestbps, chol, thalach and oldpeak.

Stages of model building

The model selection process begins with selecting an appropriate algorithm based on the type of problem and data characteristics. In this research, the algorithms used are Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, Random Forest Classifier, Naive Bayes can. These six algorithms were tested both using hypertuning and not using hypertuning. Once the model is selected, the model training process is performed by optimizing the hyperparameters using techniques such as Grid Search or Random Search. The data is divided into training data (70%) and test data (30%), and validation methods such as 5-fold cross-validation are used to ensure the model is not overfitting. Grid Search will test all combinations of predefined hyperparameters, such as n_estimators (10, 50, 100) and max_depth (None, 10, 20), while Random Search will test combinations randomly. Each hyperparameter combination is tested by training the model on the training data and evaluating its performance using metrics such as accuracy or F1-score. The hyperparameter combination that gives the best performance is selected as the final configuration of the model. Thus, this process ensures that the resulting model has optimal performance and is ready to be used for prediction on new data. The model parameter settings are presented in Fig. 6.

Evaluation and interpretation stage

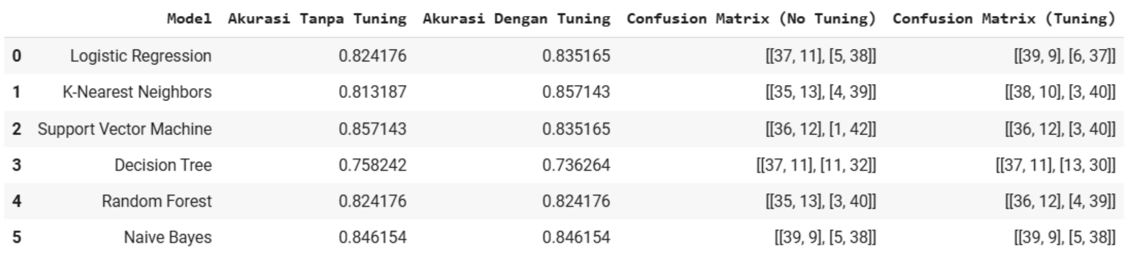
In the evaluation stage, the model that has been built is tested to measure its performance using metrics appropriate to the type of problem. The evaluation results are presented in Fig. 7.



```
# Inisialisasi model-model
models = {
    "Logistic Regression": LogisticRegression(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Machine": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Naive Bayes": GaussianNB()
}

# Parameter Tuning untuk setiap algoritma
param_grids = {
    "Logistic Regression": {'C': [0.1, 1, 10]},
    "K-Nearest Neighbors": {'n_neighbors': [3, 5, 7]},
    "Support Vector Machine": {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']},
    "Decision Tree": {'max_depth': [None, 10, 20]},
    "Random Forest": {'n_estimators': [50, 100, 200]},
    "Naive Bayes": {} # Tidak ada parameter yang perlu di-tuning
}
```

Figure 6. Model parameter settings



	Model	Akurasi Tanpa Tuning	Akurasi Dengan Tuning	Confusion Matrix (No Tuning)	Confusion Matrix (Tuning)
0	Logistic Regression	0.824176	0.835165	[[37, 11], [5, 38]]	[[39, 9], [6, 37]]
1	K-Nearest Neighbors	0.813187	0.857143	[[35, 13], [4, 39]]	[[38, 10], [3, 40]]
2	Support Vector Machine	0.857143	0.835165	[[36, 12], [1, 42]]	[[36, 12], [3, 40]]
3	Decision Tree	0.758242	0.736264	[[37, 11], [11, 32]]	[[37, 11], [13, 30]]
4	Random Forest	0.824176	0.824176	[[35, 13], [3, 40]]	[[36, 12], [4, 39]]
5	Naive Bayes	0.846154	0.846154	[[39, 9], [5, 38]]	[[39, 9], [5, 38]]

Figure 7. Model evaluation results

The figure Based on the Fig. 7, the results of analyzing the performance of six classification algorithms before and after hyperparameter tuning, several research results were obtained

1. Logistic Regression experienced a slight increase in accuracy from 0.824 to 0.835 after tuning. The change in confusion matrix showed improvement in classifying positive samples with a slight increase in the number of correct predictions.

2. K-Nearest Neighbors (KNN) showed a significant improvement from 0.813 to 0.857, indicating that tuning successfully found the optimal combination of the number of neighbors (k-value) that improved classification performance.
3. Support Vector Machine (SVM) experienced a slight decrease in accuracy from 0.857 to 0.835, possibly due to overfitting or inappropriate kernel parameter selection.
4. Decision Tree showed a decrease in performance after tuning, with accuracy dropping from 0.758 to 0.736. This indicates that tuning may have caused the model to become more complex without improved generalization.
5. Random Forest had a stable performance, with accuracy remaining at 0.824 before and after tuning, indicating that tuning did not have a significant impact on this model.
6. Naïve Bayes also had no change in performance, with a fixed accuracy of 0.846, indicating that this method is not significantly affected by hyperparameter tuning.

From these results it can be concluded that hyperparameter tuning does not always improve model accuracy. Algorithms such as Logistic Regression and KNN benefit from tuning, while SVM and Decision Tree experience performance degradation, and Random Forest and Naïve Bayes remain stable. This confirms that the effectiveness of tuning is highly dependent on the characteristics of the algorithm and the dataset used.

4 Conclusions

Based on this research, it can be concluded that hyperparameter tuning does not always significantly improve model performance. Tests were conducted on six classification algorithms, namely Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, Random Forest, and Naïve Bayes, using the heart disease dataset. Evaluation is done by measuring accuracy, precision, recall, and F1-score before and after tuning using grid search technique to find the best hyperparameter combination.

The results show that the effectiveness of hyperparameter tuning is highly dependent on the algorithm and dataset characteristics. In Logistic Regression and KNN, tuning successfully improves model performance by optimizing parameters such as the number of neighbors in KNN and regulation in Logistic Regression. In SVM and Decision Tree, tuning actually caused a decrease in accuracy and in Random Forest and Naïve Bayes, tuning did not make a significant change to model performance.

This study confirms that hyperparameter tuning does not always provide better results, and its application should be tailored to the characteristics of the dataset and the specific needs of each algorithm. In the process of data processing and modeling, a deep understanding of the algorithms used and directed experiments are needed to ensure that tuning really provides benefits in improving model performance.

References

[1] D. Cielen, A. D. B. Meysman, and M. Ali, *Introducing Data Science*. 2016.

[2] M. Arhami and M. Nasir, *Data Mining - Algoritma dan Implementasi*. Yogyakarta: Penerbit Andi, 2020.

[3] K. Pinaryanto, R. A. Nugroho, and Y. Basilius, “Classification of Toddler Nutrition Using C4 . 5 Decision Tree Method,” *Int. J. Appl. Sci. Smart Technol.*, vol. 3, no. 1, pp. 131–142, 2021.

[4] B. L. Qasthari, E. Susanti, and M. Sholeh, “Classification of Lung and Colon Cancer Histopathological Images Using Convolutional Neural Network (CNN) Method on a Pre-Trained Models,” *Int. J. Appl. Sci. Smart Technol.*, vol. 5, no. 1, pp. 133–142, 2023.

[5] M. Barlow, *Learning to Love Data Science*. Gravenstein Highway North, Sebastopol: O’Reilly Media, Inc, 2015.

[6] D. Sarkar, R. Bali, and T. Sharma, *Practical Machine Learning with Python*. Bangalore, Karnataka, India: Apress, 2018.

[7] D. N. Ardelia, H. D. Arifin, S. Daniswara, and A. P. Sari, “Klasifikasi Harga Ponsel Menggunakan Algoritma Logistic Regression,” *J. INFORMATICS Electron. Eng.*, vol. 04, no. 01, pp. 37–43, 2024.

[8] R. A. Putri and N. S. Fatonah, “Perbandingan Metode Klasifikasi serta Analisis Faktor Akademis Pola Kelulusan Mahasiswa di Perguruan Tinggi,” *J. Inform. J. Pengemb. IT*, vol. 7, no. 2, pp. 109–117, 2022, doi: 10.30591/jpit.v7i2.3082.

[9] J. A Ilemobayo *et al.*, “Hyperparameter Tuning in Machine Learning: A

Comprehensive Review," *J. Eng. Res. Reports*, vol. 26, no. 6, pp. 388–395, 2024, doi: 10.9734/jerr/2024/v26i61188.

[10] M. Arifin and S. Adiyono, "Hyperparameter Tuning in Machine Learning to Predicting Student Academic Achievement," *Int. J. Artif. Intelegence Res.*, vol. 8, no. 1, pp. 1–8, 2024.

[11] M. Fajri and A. Primajaya, "Komparasi Teknik Hyperparameter Optimization pada SVM untuk Permasalahan Klasifikasi dengan Menggunakan Grid Search dan Random Search," *J. Appl. Informatics Comput.*, vol. 7, no. 1, pp. 14–19, 2023, doi: 10.30871/jaic.v7i1.5004.

[12] N. Amalia and Asmunin, "Optimasi Algoritma Random Forest dengan Hyperparameter Tuning Menggunakan GridSearchCV untuk Prediksi Nasabah Churn pada Industri Perbankan," *Manaj. Inf.*, vol. 16, no. 1, pp. 1–9, 2024.

[13] A. Hamied Nababan and M. Y. Hutagalung, "Hyperparameter Tuning Pada Model Stance Detection Menggunakan GridSearchCV," *J. Sains dan Teknol.*, vol. 5, no. 1, pp. 205–209, 2023, [Online]. Available: https://doi.org/10.55338/saintek.v5i1.1505

[14] T. A. E. Putri, T. Widiharih, and R. Santoso, "Penerapan Tuning Hyperparameter Randomsearchcv Pada Adaptive Boosting Untuk Prediksi Kelangsungan Hidup Pasien Gagal Jantung," *J. Gaussian*, vol. 11, no. 3, pp. 397–406, 2023, doi: 10.14710/j.gauss.11.3.397-406.

[15] A. Baita, I. A. Prasetyo, and N. Cahyono, "Hyperparameter Tuning on Random Forest for Diagnose Covid-19," *JIKO (Jurnal Inform. dan Komputer)*, vol. 6, no. 2, pp. 138–143, 2023, doi: 10.33387/jiko.v6i2.6389.

[16] S. W. Lubis, P. P. Adikara, and B. Darma, "Optimasi Hyperparameter Support Vector Machine Dengan Particle Swarm Optimization Terhadap Klasifikasi Citra Digital Imbalanced," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. Vol 8 No 3, 2024, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/13508

[17] G. F. Fahrudin, S. Suroso, and S. Soim, "Pengembangan Model Support Vector Machine untuk Meningkatkan Akurasi Klasifikasi Diagnosis Penyakit Jantung," *J. Teknol. Sist. Inf. dan Apl.*, vol. 7, no. 3, pp. 1418–1428, 2024, doi: 10.32493/jtsi.v7i3.42254.

[18] W. Firgiawan, D. Yustianisa, and N. A. Nur, "Hyperparameter Tuning for Optimizing Stunting Classification with KNN , SVM , and Naïve Bayes Algorithms," *J. TEKNO KOMPAK*, vol. 19, no. 1, pp. 92–104, 2025.

[19] P. J. Deo and Y. B. D. Setianto, "K-Nearest Neighbor (KNN) Algorithm Performance In Diabetes Case Study," *PROXIES*, vol. 6, no. 1, pp. 93–102, 2022.

[20] M. Rizki, A. Hermawan, and D. Avianto, “Optimization of Hyperparameter K in K-Nearest Neighbor Using Particle Swarm Optimization,” *JUITA J. Inform.*, vol. 12, no. 1, p. 71, 2024, doi: 10.30595/juita.v12i1.20688.

[21] N. Hendradinata, I. Gede, and S. Astawa, “Hyperparameter Tuning Algoritma KNN Untuk Klasifikasi Kanker Payudara Dengan Grid Search CV,” *JNATIA Vol.*, vol. 1, no. November, pp. 397–402, 2022.

[22] S. Suraya, M. Sholeh, and D. Andayati, “Penerapan Metode Clustering Dengan Algoritma K-Means Pada Pengelompokan Indeks Prestasi Akademik Mahasiswa,” *Skanika*, vol. 6, no. 1, pp. 51–60, 2023, doi: 10.36080/skanika.v6i1.2982.

This page intentionally left