Jurnal Teknik Informatika Vol. 18 No. 1, April 2025 (111-121)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i1.42260

JURNAL TEKNIK INFORMATIKA

Homepage : http://journal.uinjkt.ac.id/index.php/ti

Comparison of Hyperparameter Tuning Methods for Optimizing K-Nearest Neighbor Performance in Predicting Hypertension Risk

Dimas Trianda[1], Dedy Hartama[2], Solikhun[3]

[1,3]Informatics Engineering Study Program, STIKOM Tunas Bangsa
[2]Information Systems Study Program, STIKOM Tunas Bangsa
[1,2,3]Jl. Sudirman, Pematangsiantar, Indonesia

Article:
Accepted: January 10, 2025
Revised: August 18, 2024
Issued: April 30, 2025

***Correspondence Address:**
trianda28721@gmail.com

ABSTRACT

Hypertension is a major cause of cardiovascular disease, making early risk prediction essential. According to WHO, hypertension cases are estimated to reach 1.28 billion by 2023. This study aims to optimize the K-Nearest Neighbor (KNN) algorithm for predicting hypertension risk through hyperparameter tuning. Three methods *Grid SearchCV*, *Bayes SearchCV*, and *Random SearchCV* are compared to determine the best parameter configuration. The dataset, obtained from *Kaggle*, consists of 520 balanced samples (260 positive and 260 negative) with 18 health-related features such as age, gender, blood pressure, cholesterol, glucose, and others. After preprocessing, the KNN model is tuned using each method by testing combinations of neighbors (k), weight types, and distance metrics. Results show *Bayes SearchCV* achieved the highest accuracy of 92%, outperforming the baseline KNN model, which had 85% accuracy. The ROC AUC score of 0.96191 also indicates excellent classification performance. In conclusion, *Bayes SearchCV* significantly improves KNN's predictive ability in hypertension risk classification.

Keywords : *comparison; hypertension; machine learning; KNN; hyperparameter tuning.*

Jurnal Teknik Informatika Vol. 18 No. 1, April 2025 (111-121)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i1.42260

1. INTRODUCTION

Hypertension, it is a long-term condition characterized by increased arterial pressure, which plays a significant role in causing cardiovascular diseases like heart attacks and strokes [1], [2]. Worldwide, more than one billion people suffer from hypertension and its prevalence is increasing, particularly in countries with low and middle incomes [3], [4]. The increasing burden of hypertension underscores the need for effective early prediction and detection to enable timely medical interventions and improve patient health outcomes [5], [6]. With the growing reliance on data-driven approaches in healthcare, predictive models have become essential tools in managing chronic diseases like hypertension [7].

One such data driven approach is data mining, which has proven effective in analyzing large datasets and uncovering hidden patterns [8], [9]. In the healthcare sector, data mining techniques are applied to predict, classify, and analyze disease outcomes, providing valuable insights that can guide clinical decision making [10], [11]. Machine learning, an integral part of data mining, is particularly useful for disease prediction, including hypertension, by processing patient data to identify key risk factors [12], [13]. Among the various *machine learning* algorithms, *K-Nearest Neighbor* (KNN) is widely appreciated for its simplicity and effectiveness in classification tasks [14]. KNN's ability to handle complex multidimensional data makes it a suitable choice for predicting hypertension risk based on factors such as age, gender, and cholesterol levels [15], [16].

However, despite its effectiveness, one limitation of KNN lies in determining the optimal hyperparameters, such as the number of neighbors or the distance metric, which greatly affects the model's performance [17], [18]. This is why *hyperparameter tuning* is crucial. *Hyperparameter tuning* involves selecting the best set of parameters to enhance the accuracy, precision, overall model performance [19]. Traditional manual tuning methods are often inefficient and prone to errors, especially when dealing with large data sets and complex models. To address these challenges, automated techniques have been developed to expedite this tuning process.

Some of the most popular techniques for hyperparameter optimization are *Grid Search Cross-Validation (Grid SearchCV), Bayesian Optimization Search Cross-Validation (Bayes SearchCV),* and *Random Search Cross-Validation (Random SearchCV)* [20], [21]. *Grid SearchCV* systematically searches through a predefined set of hyperparameter combinations, evaluating each combination using cross-validation to minimize overfitting, ensuring that the model generalizes well to new data [22], [23]. By assessing model performance across multiple data subsets, *Grid SearchCV* identifies the optimal hyperparameter combination that maximizes metrics such as accuracy [24]. This comprehensive search process makes it highly effective for optimizing machine learning algorithms such as KNN, particularly in healthcare applications [25]. Meanwhile, *Bayes SearchCV* employs a probability based approach to select more optimal hyperparameter combinations with fewer iterations compared to *Grid SearchCV*. This technique leverages Bayesian Optimization to update the probability distribution of the best hyperparameters based on previous evaluations, improving search efficiency and reducing computational time [26], [27]. In comparison, *Random SearchCV* selects hyperparameter combinations randomly from a predefined search space [28], [29]. Although this method does not conduct an exhaustive search like *Grid SearchCV,* research has shown that with a sufficient number of iterations, *Random SearchCV* can often find near-optimal results with lower computational costs. Each of these three methods has its own advantages and limitations in optimizing KNN hyperparameters. Therefore, this study focuses on comparing *Grid SearchCV, Bayes SearchCV,* and *Random SearchCV* to enhance KNN performance in predicting hypertension risk.

Previous studies that serve as references for this research include the study by Sudriyanto, which focused on neural network optimization for hypertension prediction using the Particle Swarm Optimization (PSO) algorithm. This research combined Neural Networks (NN) with PSO to optimize weights and biases, resulting in improved prediction performance. The dataset used included hypertension data with variables such as age, gender, blood pressure, cholesterol levels, and others. Experimental results showed that Neural

Networks with PSO produced a lower Root Mean Squared Error (RMSE) value (0.170) compared to Neural Networks without PSO (0.197), indicating improved accuracy in predicting hypertension risk [30].

Another study by Ongkosianbhadra and Lestari developed a hypertension risk predictive model using the Gradient-Boosting Decision Tree (GBDT) algorithm optimized with various *hyperparameter tuning* methods, including *Tree Parzen Estimation*, which achieved the highest accuracy of 74.43%. The dataset used comprised 70,693 rows of information provided by the Centers for Disease Control and Prevention (CDC), with 17 attributes covering behavior, medical history, and health status. The results demonstrated that hyperparameter optimization with *Tree Parzen Estimation* improved model performance, producing more accurate hypertension risk predictions compared to other tuning methods such as Grid Search and *Bayesian Optimization* [31].

The selection of the KNN algorithm is based on previous studies comparing various classification methods. Widodo et al. analyzed the performance of KNN, Bagging, and Random Forest in predicting breast cancer using a dataset of 286 instances. The results showed that KNN outperformed the other methods with an accuracy of 74.37%, compared to Bagging (73.29%) and Random Forest (72.92%) [32]. Another study by Amien et al. compared KNN and Naïve Bayes for diabetes classification, where KNN (K=5) achieved the highest accuracy of 90%, while Naïve Bayes reached only 80% [33]. These findings highlight the superiority of KNN in handling complex datasets, although optimizing the K value is essential for achieving the best performance.

Unlike previous studies, this research utilizes the K-Nearest Neighbor (KNN) algorithm optimized with three hyperparameter tuning techniques: *Grid SearchCV*, *Bayes SearchCV*, and *Random SearchCV*.

In this study, a hypertension prediction model was developed using the K-Nearest Neighbor (KNN) algorithm optimized with Grid SearchCV, Bayes SearchCV, and Random SearchCV. The model building process begins with data collection and preparation, including handling empty values, normalization, and encoding categorical variables to ensure data quality. Next, model selection is carried out by considering the suitability of KNN in handling classification-based health data. To increase accuracy, hyperparameter optimization was carried out using three tuning methods compared in this study. The optimized model is then evaluated based on confusion metrics such as accuracy, precision, recall and f1-score to determine the best tuning method to improve KNN performance in predicting hypertension risk. This approach aims to compare the effectiveness of these three methods in enhancing KNN performance for hypertension data classification.

This study focuses on a comparative analysis of *Grid SearchCV*, *Bayes SearchCV*, and *Random SearchCV* in optimizing KNN hyperparameters for predicting hypertension risk. By evaluating the performance of each method based on metrics such as accuracy, precision, and computational efficiency, this research aims to identify the most optimal hyperparameter tuning technique for improving KNN's ability to classify individuals based on risk factors. By combining the strengths of KNN with various hyperparameter optimization strategies, this study aims to develop a more reliable predictive model. This model is expected to assist healthcare professionals in identifying high-risk individuals more accurately, enabling earlier interventions and reducing the health burden associated with hypertension.

2. METHODS

The dataset is secondary and is taken online from the *kaggle* platform which can be accessed through the following link: https://www.kaggle.com/datasets/frederickfelix/hipertensin-arterial-mxico/data

The dataset consists of 520 data samples with 18 health attributes, including age, gender, uric acid value, HDL cholesterol, LDL cholesterol, total cholesterol, creatinine value, glucose result, triglyceride value, average glucose result, *glucosylated* hemoglobin value, weight, height, blood pressure, sleep hours, BMI, total activity, and risk. Data samples were taken using simple random sampling techniques.

Before analysis, the dataset underwent a cleaning and preprocessing stage, where each entry was examined to ensure no missing or incomplete data. The data was then split into two subsets: a training set and a testing set, ensuring that the model could generalize well and provide accurate predictions for unseen

Jurnal Teknik Informatika Vol. 18 No. 1, April 2025 (111-121)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i1.42260

data. This process guarantees that the dataset used is reliable and representative for hypertension prediction. All analyses were conducted using *Jupyter Notebook*. The research methodology is further detailed through the steps illustrated in Figure 1.

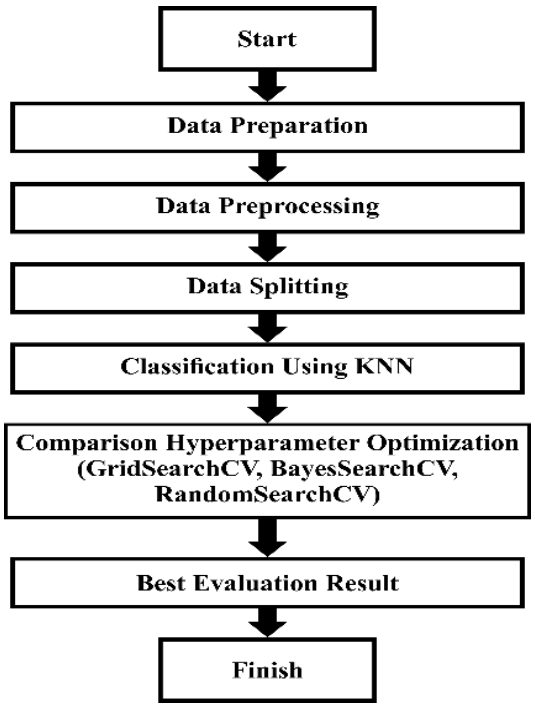

Figure 1. *Research stages*

The model in this diagram follows the Knowledge Discovery in Databases (KDD) framework because it includes the main stages in data exploration. Starting from Data Preparation, where hypertension data is selected and prepared, then Data Preprocessing to clean and normalize the data. After that, Data Splitting divides the data into train and test before carrying out Classification Using KNN as the core Data Mining process. To improve model performance, Hyperparameter Optimization was carried out using Grid SearchCV, Bayes SearchCV, and Random SearchCV. The best results are then evaluated in Best Evaluation Result before the model is completed. This process reflects the KDD stages from selection, transformation, pattern exploration, to evaluation, thus ensuring the model works optimally in predicting hypertension. For more details, see the flow methodology in Figure 2.

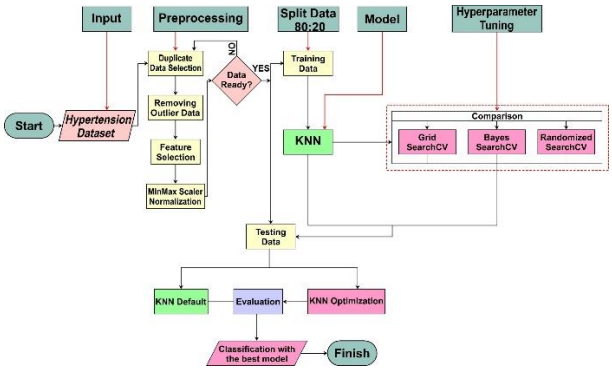

Figure 2. *Flow methodology*

Figure 2 illustrates a more detailed methodological flow for building a classification model using K-Nearest Neighbors (KNN) on the hypertension dataset. The first step is preprocessing, which involves data cleaning by selecting duplicate data, removing outliers, and applying *MinMax Scaler* normalization to ensure that feature values fall within the same range. Once the data is prepared, it is split into two parts: 80% training data for model training and 20% testing data for model evaluation. Next, the KNN model is initially trained using default parameters to obtain a baseline result. In the following step, hyperparameter tuning is performed using three optimization methods *Grid SearchCV*, *Bayes SearchCV*, and *Randomized SearchCV*. The results from each tuning method are compared to determine the optimal parameters. Once the optimized KNN model is obtained, its performance is evaluated using the testing data. Finally, the best-performing KNN model is used for classification on the hypertension dataset, producing an optimized classification model as the final outcome.

2.1. Data Preparation

This stage involves collecting a hypertension dataset that includes various features related to patient health metrics, such as gender, age, BMI, cholesterol levels, blood pressure, and more. The data was obtained from open sources, such as Kaggle. This dataset forms the foundation of the research, making it essential to ensure that the data used is relevant and representative.

2.2. Data *Preprocessing*

Data *preprocessing* is crucial for enhancing model accuracy and ensuring consistent results. At this stage, missing values in the dataset are handled using *SimpleImputer*, which can fill gaps with statistical measures such as the mean. Additionally, *MinMaxScaler*

is used to scale features so that all variables have a uniform range (usually between 0 and 1), which allows the model to process data more efficiently without being affected by large numbers.

2.3. Split Data

After the preprocessing stage, the dataset was divided into two parts: 80% for training data and 20% for testing data using the *train_test_split* function. This division was based on a previous study on the performance of the K-Nearest Neighbor (KNN) and Cross-Validation on Cardiovascular Disease Data, which compared different data split ratios (20:80, 50:50, and 80:20) [34]. The study found that the 80:20 ratio achieved the highest accuracy of 91%, making it the optimal choice for this research to ensure the best model performance. The training data was used to build the KNN model, while the testing data was reserved for evaluating the model's performance. This division is crucial to ensure that the model generalizes well to new and unseen data.

2.4. Classification Using KNN

At this stage, the K-Nearest Neighbor (KNN) algorithm is used to classify patients as either at risk or not at risk of hypertension. A pipeline is created to streamline the classification process, integrating data preprocessing steps and the KNN algorithm into a single workflow. This pipeline simplifies the process and ensures that all steps are applied consistently during both the training and testing phases. In the standard KNN scenario, the number of neighbors (K) is set to 5, using Euclidean distance as the default metric.

2.5. Hyperparameter Optimization *Comparison*

Hyperparameter tuning is performed using *Grid SearchCV, Bayes SearchCV,* and *Random SearchCV* to enhance the performance of the KNN model. This process involves testing various values for key parameters, such as the number of neighbors (K) ranging from 1 to 30, weight types (uniform and distance), and distance metrics (Manhattan and Euclidean), with cross-validation (CV) set to 5, to identify the combination that provides the best accuracy. By systematically exploring these parameters, the model is optimized to be more effective and capable of delivering more accurate predictions for the given dataset.

2.6. Validation and Evaluation

To evaluate the performance of the applied model, validation is conducted using k-fold cross-validation. In this study, k-fold cross-validation is implemented with a training-to-testing data ratio of 8:2, with variations in the value of k ranging from 1 to 30. The model will be evaluated by assessing the performance of an optimized KNN model using *Grid SearchCV, Bayes SearchCV* and *Random SearchCV* for hyperparameter tuning.

3. RESULTS AND DISCUSSION

The study comprises seven main stages: data preparation, preprocessing, data splitting into training and testing sets, classification using the KNN method, comparison of hyperparameter optimization using *Grid SearchCV, Bayes SearchCV* and *Random SearchCV* for KNN models, comparing KNN performance before and after hyperparameter optimization, and finally, obtaining the best classification results based on evaluation metrics such as accuracy.

2.1. Preparing Data For Classification

This research utilizes a 2024 dataset sourced from Kaggle. The dataset contains several main features for predicting hypertension which can be seen in table 1.

Table 1. *Description of each predicted data variable*

No.	Variable	Information
1.	*Gender* (X1)	Patient gender (1 for male, 2 for female)
2.	*Age* (X2)	Patient age.
3.	*Uric_Acid_Value* (X3)	Value of uric acid levels in the blood.
4.	*HDL_Cholesterol_Value* (X4)	The value of HDL (High-Density Lipoprotein) cholesterol in the blood, which is often called "good cholesterol".
5.	*LDL_Cholesterol_Value* (X5)	The value of LDL (Low-Density Lipoprotein) cholesterol in the blood, which is often called "bad cholesterol".
6.	*Total_Cholesterol_Value* (X6)	Total cholesterol value in the blood, including HDL, LDL and other cholesterol.
7.	*Creatinine_Value* (X7)	Assess the level of creatinine in the blood, which indicates kidney function.
8.	*Glucose_Result* (X8)	Results of blood glucose levels, usually from a fasting test or random test.
9.	*Triglycerides_Value* (X9)	Assess blood levels of triglycerides, a type of fat found in the blood.
10.	*Average_Glucose_Result* (X10)	Average blood glucose levels over a certain time period.

Jurnal Teknik Informatika Vol. 18 No. 1, April 2025 (111-121)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i1.42260

Table 1 continued…

No.	Variable	Information
11.	*Glycosylated_Hemoglobin_Value* (X11)	The glycated hemoglobin value shows the average blood glucose level over the last 2-3 months.
12.	*Weight* (X12)	The patient's weight is in kilograms (kg).
13.	*Height* (X13)	The patient's height is in centimeters (cm).
14.	*Blood_Pressure* (X14)	The patient's blood pressure usually consists of systolic and diastolic pressure.
15.	*Sleep_in_Hours* (X15)	The patient's sleep duration in hours per day.
16.	*BMI* (X16)	The patient's body mass index is calculated from body weight (kg) divided by height squared (m²).
17.	*Total_Activity* (X17)	The patient's total physical activity in a certain unit of time.
18.	*Hypertension_Risk* (Y)	The patient's risk of hypertension (1 for at risk, 0 for not at risk)

The dataset consists of 18 features with a total of 520 data entries. Of the 18 features, one feature functions as the target variable, while the other 17 features function as input. The feature targeted is "hypertension risk".

Before creating a machine learning model, the dataset must go through preprocessing steps and ensure that the classes are balanced, making it ready for further analysis, The results of data balancing can be seen in Figure 3.

Figure 3. *Data after balancing*

The data used is presented in table 2 below.

Table 2. *Dataset used*

No.	X1	X2	X3	X4	X5	…	X17	Risk
1.	1	56	4.8	34	86	…	570	1
2.	1	40	4.8	34	86	…	355	0
3.	2	55	4.8	34	86	…	480	1
4.	2	56	4.8	34	86	…	180	1
5.	1	50	4.8	34	86	…	280	1
6.	2	53	5.5	51	116	…	380	1
7.	2	55	6	44	86	…	270	1
8.	2	55	4.8	34	86	…	240	1
9.	2	38	4.8	34	86	…	420	1
10.	2	29	4.8	34	86	…	270	1
11.	2	38	4.1	40	71	…	280	1
12.	2	30	3.4	22	47	…	165	1
13.	1	44	6	31	114	…	490	1
14.	2	26	4.8	34	86	…	1020	1
15.	1	56	4.8	34	86	…	720	1
16.	1	40	4.8	34	86	…	80	1
17.	2	46	4.8	34	86	…	480	0
18.	2	26	4.8	34	86	…	70	1
…	…	…	…	…	…	…	…	…
520.	1	64	4.8	34	86	…	480	0

2.2. Data Splitting

The data is divided into two parts: the training set and the test set. This division is made so that the model can learn from the training data and then be tested with the test data to evaluate its performance. The data split is shown in Figure 4.

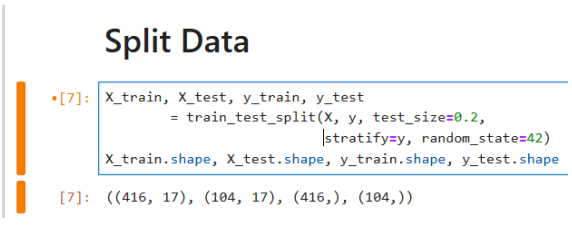

Figure 4. *Data Split Process*

Jurnal Teknik Informatika Vol. 18 No. 1, April 2025 (111-121)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i1.42260

In the code above, the input variables are stored in X, while the target variable is stored in y. The data is split use an 80:20 ratio, 80% of the data for training and 20% for testing, using the train_test_split method in the sklearn.model_selection library to split the data randomly, with the parameter stratify = y to ensure that the class distribution remains balanced between the training and test sets.

A total of 416 data points are used for training, and 104 data points are used for testing.

2.3. Building the KNN Model Without *Optimation Methods*

Before applying the Hyperparameter Tuning method for hyperparameter optimization, an initial model is built using the KNN algorithm with K = 5. At this stage, the model is trained using the dataset, which has been split into 80% training data and 20% test data. The choice of K = 5 as the default value in KNN is based on standard practices in the literature, where this value provides a good balance between bias and variance. If K is too small (e.g., K = 1), the model tends to overfit, while if K is too large, the model may underfit and lose its ability to capture local patterns in the data. Additionally, in the initial model without Hyperparameter Tuning, the distance metric used is Euclidean distance, which is the standard method in KNN. Euclidean distance measures the straight-line distance between two points in a multidimensional space, making it a commonly used approach due to its simplicity and effectiveness in determining sample proximity within the dataset.

Next, in the confusion matrix for KNN without Hyperparameter Tuning method, as shown in Figure 5, the results from the confusion matrix indicate that out of the total 104 test data, there are 43 true positives, 45 true negatives, 9 false positives, and 7 false negatives.

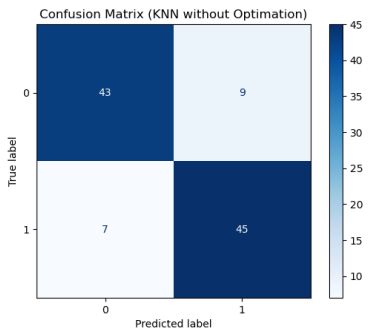

Figure 5. *Confusion matrix KNN without optimation*

Next are the prediction results and evaluation generated from the KNN model without Hyperparameter Tuning.

Predict and Evaluate

```
[31]: y_pred_knn = knn_model.predict(X_test_scaled)
      print("Classification Report (KNN without Optimation):\n",
            classification_report(y_test, y_pred_knn))
```

```
Classification Report (KNN without Optimation):
              precision    recall  f1-score   support

           0       0.86      0.83      0.84        52
           1       0.83      0.87      0.85        52

    accuracy                           0.85       104
   macro avg       0.85      0.85      0.85       104
weighted avg       0.85      0.85      0.85       104
```

Figure 6. *Classification report KNN without optimation*

The training and testing scores for the KNN model without Hyperparameter Tuning can be seen in Figure 7.

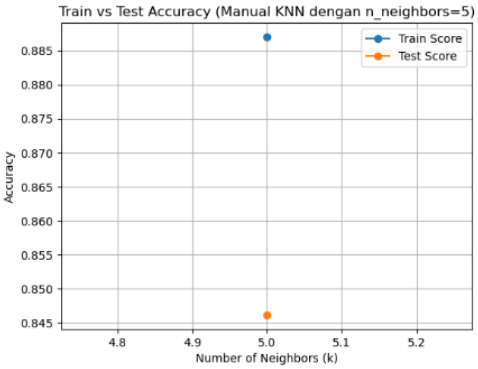

Figure 7. *Plot training vs testing accuracy KNN without optimation*

As shown in Figure 7, the accuracy obtained from the training data reached 89%, indicating that the model can recognize patterns in the training data well. However, when the model was tested using the test data, the accuracy dropped to 85%. This difference in accuracy suggests the presence of overfitting, where the model is too fitted to the training data and struggles to generalize to new, unseen data.

Overfitting occurs when the model learns too many details and noise from the training data, leading to decreased performance on the test data. Therefore, the next step is to compare hyperparameter optimization using *Grid SearchCV*, *Bayes SearchCV*, and *Random SearchCV* to find the optimal value for K, in order to improve accuracy and reduce the risk of overfitting with the help of K-Fold Cross Validation.

Jurnal Teknik Informatika Vol. 18 No. 1, April 2025 (111-121)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i1.42260

2.4. Comparison of Hyperparameter Tuning Methods

At this stage, a comparison of hyperparameter tuning methods *Grid SearchCV*, *Bayes SearchCV*, and *Random SearchCV* was conducted to determine the best parameter combination for the KNN model. The range of explored hyperparameter values included K values from 1 to 30, uniform and distance-based weighting options, and p-values tested as 1 (Manhattan) and 2 (Euclidean). This process was performed using cross-validation with five different scenarios.

The best hyperparameter model was achieved using KNN with Bayes SearchCV. The results of the comparison among the three hyperparameter tuning methods are presented in Table 3 based on figure 8, 9, and 10.

Table 3. *The accuracy of the models developed in the study and their changes due to the optimization process*

	Standard Models	Optimized Models		
	KNN	GS	BS	RS
Train	89 %	87 %	87 %	87 %
Test	85 %	89 %	92 %	91 %

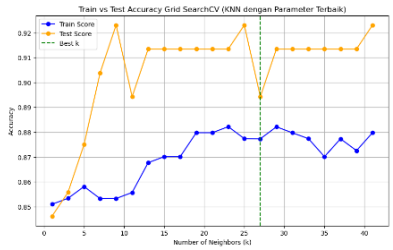

Figure 8. *Plot training vs testing accuracy KNN with grid searchCV*

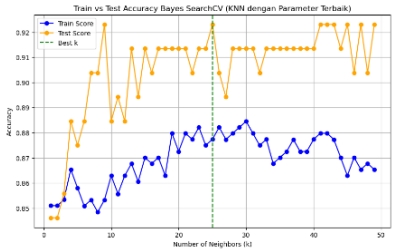

Figure 9. *Plot training vs testing accuracy KNN with bayes searchcv*

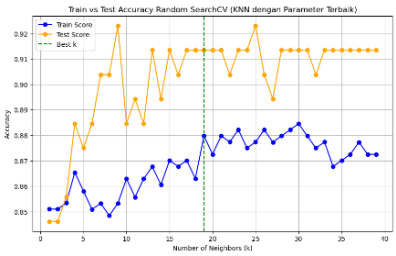

Figure 10. *Plot training vs testing accuracy KNN with random searchCV*

The following are the prediction results and evaluation produced by the KNN model with Bayes SearchCV.

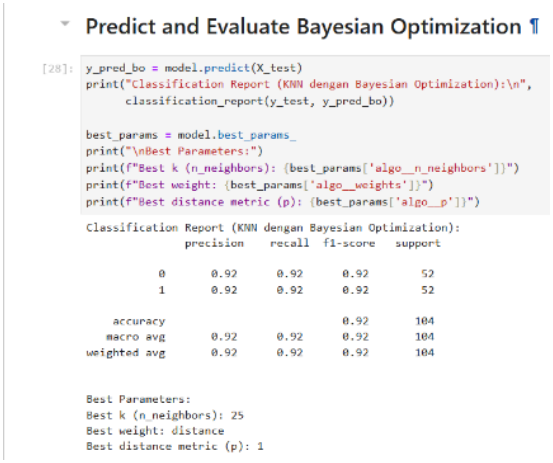

Figure 11. *Classification report KNN with bayes searchCV*

The tuning results indicate that the best parameters for the model are K value of 25, p of 1 (Manhattan), and distance weights. The accuracy achieved by this model is 92% for the training data and 87% for the testing data and precision of 92%, recall of 92%, and f1-score of 92%. indicating that there is no *overfitting* as the training and testing accuracies are fairly balanced.

Next, in the confusion matrix of KNN with *Bayes SearchCV*, as shown in Figure 12, the results indicate that out of a total of 104 test data, there are 48 true positives, 48 true negatives, 4 false positives, and 4 false negatives.

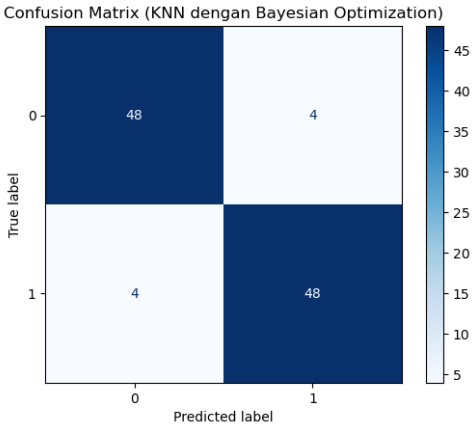

Figure 12. *Confusion matrix KNN with bayes searchCV*

The calculation process for accuracy, precision, recall, and F1-score based on the confusion matrix is carried out as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} = \frac{48+48}{48+4+4+48} = 0,92$$

$$Precission = \frac{TP}{TP+FP} = \frac{48}{48+4} = 0,92$$

$$Recall = \frac{TP}{TP+FN} = \frac{48}{48+4} = 0,92$$

Next is to calculate the F1 Score which is a comparison of the weighted average precision and recall.

$$F1\ Score = 2 * \frac{(Recall * Precission)}{(Recall + Precission)}$$

$$F1 - Score = 2 * \frac{(92\% * 92\%)}{(92\% + 92\%)} = 92\%$$

The model also achieved an *ROC AUC Score* of 0.96191, which demonstrates good performance in classification. The *ROC Curve* can be seen in Figure 13.

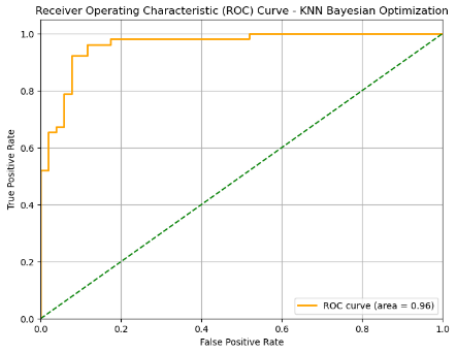

Figure 13. *ROC curve bayes searchCV*

2.5. Exploratory Data Analysis (EDA)

Next, the researcher analyzed the relationship between condition attributes and the target using exploratory data analysis (EDA) to identify the conditions that contribute to the highest risk of hypertension.

From the EDA process, the data identified the highest-risk factors as follows:

Table 4. *The results of the EDA*

***Exploratory Data Analysis (EDA)* Plot Results**	
Factors	***Result***
Gender	Woman
Age	29 – 59
Uric Acid Value	4.8
HDL Cholesterol Value	34
LDL Cholesterol Value	86
Total Cholesterol Value	139
Creatinine Value	0.58
Glucose Result	92
Triglycerides Value	123
Average Glucose Result	103
Glycosylated_Hemoglobin_Value	5.2
Weight	Unknown
Height	Unknown
Blood_Pressure	Unknown
Sleep in Hours	< 6 Hours
BMI	> 25 or higher
Total_Activity	Unknown

CONCLUSION

This study successfully optimized the *K-Nearest Neighbor (KNN)* algorithm for predicting hypertension risk, with *Bayes SearchCV* outperforming *Grid SearchCV* and *Random SearchCV*. The optimized model achieved 92% training accuracy and 87% testing accuracy, along with a ROC AUC score of 0.96191, indicating excellent classification performance and strong generalization. Exploratory Data Analysis revealed that women aged 29–59 were the most vulnerable group in the dataset. Key contributing factors included uric acid, HDL/LDL cholesterol, total cholesterol, triglycerides, average glucose level, HbA1c value, BMI over 25, and short sleep duration. However, missing variables such as weight, height, blood pressure, and activity level may have limited the depth of the analysis. This model has the potential to assist healthcare professionals in the early detection of hypertension risk. Future research should evaluate the model using larger and more diverse datasets, explore additional hyperparameter tuning methods, and incorporate more detailed patient attributes to improve prediction accuracy and support more comprehensive preventive strategies.

REFERENCES

[1] W. Frąk, A. Wojtasińska, W. Lisińska, E. Młynarska, B. Franczyk, and J. Rysz, "Pathophysiology of Cardiovascular Diseases: New Insights into Molecular Mechanisms of Atherosclerosis, Arterial Hypertension, and Coronary Artery Disease," *Biomedicines*, vol. 10, no. 8, 2022, doi: 10.3390/biomedicines10081938.

[2] K. Gadó, A. Szabo, D. Markovics, and A. Virág, "Most common cardiovascular diseases of the elderly – A review article," *Dev. Heal. Sci.*, vol. 4, no. 2, pp. 27–32, 2022, doi: 10.1556/2066.2021.00048.

[3] M. H. Elnaem *et al.*, "Disparities in Prevalence and Barriers to Hypertension Control: A Systematic Review," *Int. J. Environ. Res. Public Health*, vol. 19, no. 21, pp. 1–16, 2022, doi: 10.3390/ijerph192114571.

[4] A. Mohammed Nawi *et al.*, "The Prevalence and Risk Factors of

Hypertension among the Urban Population in Southeast Asian Countries: A Systematic Review and Meta-Analysis," *Int. J. Hypertens.*, vol. 2021, 2021, doi: 10.1155/2021/6657003.

[5] O. K. A, "International Journal of Research Publication and Reviews Machine Learning in Predictive Modelling : Addressing Chronic Disease Management through Optimized Healthcare Processes," *Int. J. Res. Publ. Rev.*, vol. 6, no. 1, pp. 1525–1539, 2025.

[6] M. A. Gadjiev *et al.*, "Innovations in arterial blood pressure management: enhancing public health through effective prevention methods," *Rev. Latinoam. Hipertens.*, vol. 19, no. 6, pp. 277–282, 2024, doi: 10.5281/zenodo.12674070.

[7] Ibrahim Adedeji Adeniran, Christianah Pelumi Efunniyi, Olajide Soji Osundare, and Angela Omozele Abhulimen, "Data-driven decision-making in healthcare: Improving patient outcomes through predictive modeling," *Int. J. Sch. Res. Multidiscip. Stud.*, vol. 5, no. 1, pp. 059–067, 2024, doi: 10.56781/ijsrms.2024.5.1.0040.

[8] X. Shu and Y. Ye, "Knowledge Discovery: Methods from data mining and machine learning," *Soc. Sci. Res.*, vol. 110, no. April 2022, p. 102817, 2023, doi: 10.1016/j.ssresearch.2022.102817.

[9] Salumanda and Christian, "Investigating Data Mining Methods For Pattern And Relationship Detection In Large Datasets," *Int. J. Data Sci. Eng.*, vol. 1, no. 1, pp. 1–9, 2023.

[10] S. M. D. A. C. Jayatilake and G. U. Ganegoda, "Involvement of Machine Learning Tools in Healthcare Decision Making," *J. Healthc. Eng.*, vol. 2021, 2021, doi: 10.1155/2021/6679512.

[11] W. T. Wu *et al.*, "Data mining in clinical big data: the frequently used databases, steps, and methodological models," *Mil. Med. Res.*, vol. 8, no. 1, pp. 1–12, 2021, doi: 10.1186/s40779-021-00338-z.

[12] E. Dritsas and M. Trigka, "Efficient Data-Driven Machine Learning Models for Cardiovascular Diseases Risk Prediction," *Sensors (Switzerland)*, vol. 50, 2023, doi: 10.5937/mckg50-11761.

[13] A. Rahim, Y. Rasheed, F. Azam, M. W. Anwar, M. A. Rahim, and A. W. Muzaffar, "An Integrated Machine Learning Framework for Effective Prediction of Cardiovascular Diseases," *IEEE Access*, vol. 9, pp. 106575–106588, 2021, doi: 10.1109/ACCESS.2021.3098688.

[14] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat, "Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications," *J. Big Data*, vol. 11, no. 1, 2024, doi: 10.1186/s40537-024-00973-y.

[15] P. K. Bhowmik *et al.*, "Advancing Heart Disease Prediction through Machine Learning: Techniques and Insights for Improved Cardiovascular Health," *Br. J. Nurs. Stud.*, no. 2022, pp. 35–49, 2024, doi: 10.32996/bjns.

[16] M. A. Naser, A. A. Majeed, M. Alsabah, T. R. Al-Shaikhli, and K. M. Kaky, "A Review of Machine Learning's Role in Cardiovascular Disease Prediction: Recent Advances and Future Challenges," *Algorithms*, vol. 17, no. 2, pp. 1–33, 2024, doi: 10.3390/a17020078.

[17] C. Zhang, P. Zhong, M. Liu, Q. Song, Z. Liang, and X. Wang, "Hybrid Metric K-Nearest Neighbor Algorithm and Applications," *Math. Probl. Eng.*, vol. 2022, 2022, doi: 10.1155/2022/8212546.

[18] S. V. Razavi-Termeh, A. Sadeghi-Niaraki, S. Razavi, and S. M. Choi, "Enhancing flood-prone area mapping: fine-tuning the K-nearest neighbors (KNN) algorithm for spatial modelling," *Int. J. Digit. Earth*, vol. 17, no. 1, pp. 1–29, 2024, doi: 10.1080/17538947.2024.2311325.

[19] Y. A. Ali, E. M. Awwad, M. Al-Razgan, and A. Maarouf, "Hyperparameter Search for Machine Learning Algorithms for Optimizing the Computational Complexity," *Processes*, 2023, doi: https:// doi.org/10.3390/pr11020349.

[20] M. Sahu and S. Soni, "A Predictive Approach to Employee Turnover Through Machine Learning," *Int. J. Mod. Eng. Manag. Res.*, vol. 12, no. 3, pp. 5–18, 2024.

[21] A. H. Fristiana, S. A. I. Alfarozi, A. E. Permanasari, M. Pratama, and S. Wibirama, "A Survey on Hyperparameters Optimization of Deep

Learning for Time Series Classification," *IEEE Access*, vol. 12, no. November, pp. 191162–191198, 2024, doi: 10.1109/ACCESS.2024.3516198.

[22] P. K. Sahu and T. Fatma, "Optimized Breast Cancer Classification Using PCA-LASSO Feature Selection and Ensemble Learning Strategies with Optuna Optimization," *IEEE Access*, vol. 11, p. 1, 2025, doi: 10.1109/ACCESS.2025.3539746.

[23] A. A. Albishri and M. M. Dessouky, "A Comparative Analysis of Machine Learning Techniques for URL Phishing Detection," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 6, pp. 18495–18501, 2024, doi: https://doi.org/10.48084/etasr.8920.

[24] K. Alemerien, S. Alsarayreh, and E. Altarawneh, "Diagnosing Cardiovascular Diseases using Optimized Machine Learning Algorithms with GridSearchCV," *J. Appl. Data Sci.*, vol. 5, no. 4, pp. 1539–1552, 2024, doi: 10.47738/jads.v5i4.280.

[25] H. K. Bharadwaj *et al.*, "A Review on the Role of Machine Learning in Enabling IoT Based Healthcare Applications," *IEEE Access*, vol. 9, pp. 38859–38890, 2021, doi: 10.1109/ACCESS.2021.3059858.

[26] Y. Zhao, W. Zhang, and X. Liu, "Grid search with a weighted error function: Hyper-parameter optimization for financial time series forecasting," *Appl. Soft Comput.*, vol. 154, no. February, p. 111362, 2024, doi: 10.1016/j.asoc.2024.111362.

[27] Y. Han and I. Joe, "Enhancing Machine Learning Models Through PCA, SMOTE-ENN, and Stochastic Weighted Averaging," *Appl. Sci.*, vol. 14, no. 21, 2024, doi: 10.3390/app14219772.

[28] M. Ridwan and E. Utami, "Optimized Hyperparameter Tuning for Improved Hate Speech Detection with Multilayer Perceptron," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 8, no. 4, pp. 525–534, 2024, doi: https://doi.org/10.29207/resti.v8i4.5949.

[29] M. Muntasir Nishat *et al.*, "A Comprehensive Investigation of the Performances of Different Machine Learning Classifiers with SMOTE-ENN Oversampling Technique and Hyperparameter Optimization for Imbalanced Heart Failure Dataset," *Sci. Program.*, vol. 2022, no. Cvd, 2022, doi: 10.1155/2022/3649406.

[30] S. Sudriyanto, "Optimizing Neural Networks Using Particle Swarm Optimization (PSO) Algorithm for Hypertension Disease Prediction," *JEECOM J. Electr. Eng. Comput.*, vol. 5, no. 2, pp. 278–284, 2023, doi: 10.33650/jeecom.v5i2.6759.

[31] F. V. Ongkosianbhadra and C. C. Lestari, "Pengembangan Model Prediksi Risiko Hipertensi Menggunakan Algoritma Gradient Boosting Decision Tree yang Dioptimalkan dengan Hyperparameter Tuning Tree Parzer Estimation," *J. Inform. dan Sist. Inf.*, vol. 9, no. 2, pp. 35–44, 2023, doi: 10.37715/juisi.v9i2.4403.

[32] A. M. Widodo, N. Anwar, B. Irawan, L. Meria, and A. Wisnujati, "Komparasi Performansi Algoritma Pengklasifikasi KNN , Bagging, Dan Random Forest Untuk Prediksi Kanker Payudara," *Konf. Nas. Ilmu Komput.*, pp. 367–372, 2021.

[33] I. L. F. Amien, W. Astuti, and K. M. Lhaksamana, "Perbandingan Metode Naïve Bayes dan KNN (K-Nearest Neighbor) dalam Klasifikasi Penyakit Diabetes," *e-Proceeding Eng.*, vol. 10, no. 2, pp. 1911–1920, 2023.

[34] I. P. Putri, "Analisis Performa Metode K-Nearest Neighbor (KNN) dan Crossvalidation pada Data Penyakit Cardiovascular," *Indones. J. Data Sci.*, vol. 2, no. 1, pp. 21–28, 2021, doi: 10.33096/ijodas.v2i1.25.