EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

Optimizing mushroom classification through machine learning and hyperparameter tuning

Hamidah Maulida Khasanah[1)], Afrig Aminuddin*[2)], Ferian Fauzi Abdulloh[1)], Majid Rahardi[1)], Hairani Hairani[3)] and Bima Pramudya Asaddulloh[4)]

[1)]Department of Informatics, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia
[2)]Department of Information System, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia
[3)]Department of Computer Science, Faculty of Engineering, Bumigora University, Mataram, 83127, Indonesia
[4)]Department of Informatics, Postgraduate Program, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia

Received 21 May 2024
Revised 15 August 2024
Accepted 11 September 2024

Abstract

This research explores the application of machine learning in the classification of mushrooms as poisonous or edible, emphasizing the importance of optimal model performance to ensure food safety. This study compares four classification algorithms-Random Forest, Logistic Regression, Decision Tree, and Naive Bayes-before optimizing the two best models through Hyperparameter Tuning using Grid Search. The proposed method involves Exploratory Data Analysis (EDA), Data Preprocessing, Classification Modeling, Performance Evaluation, and Hyperparameter Tuning. The dataset used is Mushroom Classification data, and the results show that the Random Forest algorithm performs better with ROC values close to 100%, high recall, and good F1-Score. Hyperparameter tuning further improved the ROC and recall of the Random Forest model, emphasizing its adaptability to the nature of the dataset. This research emphasizes the importance of robust data processing and model optimization to achieve accurate and reliable predictions in mushroom classification, contributing to food safety endeavors.

Keywords: Machine learning, Mushroom classification, Hyperparameter tuning

1. Introduction

Mushrooms are heterotrophic eukaryotic organisms that require organic compounds for growth and can be unicellular or multicellular. Mushrooms are generally classified into three main categories, namely, edible, poisonous, and unknown status [1]. Mushrooms with an unknown status are often categorized as poisonous to avoid potential health risks. Mushrooms have benefits in various fields, including food, agriculture, health, and economy [2]. In the food sector, mushrooms are a cheap and readily available functional food ingredient and substitute, which is protein-rich and a source of vitamins, carbohydrates, fiber, minerals, essential amino acids, unsaturated fatty acids, and saturated fats [3]. Furthermore, mushrooms also contain active ingredients that are beneficial for health, such as polysaccharides (glucans), triterpenoids, nucleotides, mannitol, and alkaloids [4]. However, some species can cause severe infections and poisoning, such as nausea and vomiting, and organ damage, such as liver necrosis and even death.

Accurate mushroom identification is crucial to prevent severe consequences of poisonous mushroom consumption [5]. Mushrooms can be recognized by their size, color, cap, and stem shape. Manual methods are often inadequate due to wide morphological variations and similarities between harmful and harmless species [6]. Of the many species of mushrooms, only 50-100 species are edible [7]. In other words, most mushrooms are inedible [8]. Some poisonous species have similarities with edible mushrooms, making identification based solely on physical traits complex.

Reducing the risks associated with mushroom consumption requires an effective method of determining which mushrooms are safe and which are poisonous [9]. The application of machine learning in mushroom classification is highly relevant in this context. This technology utilizes artificial intelligence techniques to analyze morphological data automatically and accurately, simplifying the identification process and reducing the chance of errors [10]. Using more extensive and complex data, machine learning develops classification models that effectively distinguish mushrooms that are safe for consumption from those that are poisonous [11]. This approach aids public health by reducing the risk of poisonous mushroom consumption and ensuring food safety.

Researchers enhanced the mushroom classification evaluation metrics with Hyperparameter Tuning using the Grid Search method. Grid Search is a practical approach to systematically search for the best hyperparameter combination to improve model performance [12]. Hyperparameters are settings that affect how the model is built, and in this study, researchers focused on determining specific values for hyperparameters, such as learning rate. The researcher specifies a range of values for each hyperparameter, and Grid Search tests all combinations of those values [13]. The initial evaluation compared four classification methods to identify the two lowest-performing models, which were then optimized with Hyperparameter Tuning. This technique accelerated the search for the best parameters and improved model performance, ensuring only safe mushrooms were classified as edible.

*Corresponding author.
Email address: afrig@amikom.ac.id
doi: 10.14456/easr.2024.61

This research used four classification algorithms to distinguish between safe and poisonous mushrooms: Random Forest, Logistic Regression, Decision Tree, and Naive Bayes. Random Forest was chosen for its ability to reduce overfitting by combining multiple decision trees [14]. Logistic Regression provided transparent and interpretable classification probabilities [15]. Decision Tree provides an intuitive structure for making decisions, although it is prone to overfitting without tuning [16]. Naive Bayes effectively handles extensive data, assuming independence between features offering good speed and performance [17]. Each algorithm has its strengths and weaknesses, and this selection aims to find the most accurate and efficient mushroom classification model.

The structure of this paper is organized as follows. Section I presents an introduction to the problems and proposed solutions in the application of machine learning. Section II presents works related to the application of machine learning. Section III details the dataset used, followed by the proposed steps, including Exploratory Data Analysis (EDA), Data Preprocessing, Classification Modeling, Performance Evaluation, and Hyperparameter Tuning. Section IV presents the visual analysis of EDA and performance evaluation on models such as Random Forest, Logistic Regression, Decision Tree, Naive Bayes, and Hyperparameter Tuning. The study is concluded in Section V.

2. Related works

Paudel and Bhatta [18] evaluated the performance of Random Forest and REP Tree algorithms for mushroom classification using the UCI Machine Learning Repository dataset, with Random Forest achieving 100% in all metrics. While this study highlights the effectiveness of Random Forest, it has notable limitations. The reliance on a single dataset raises concerns about the generalizability of the results. Additionally, the study's focus on only two algorithms—Random Forest and REP Tree—without exploring other classifiers or incorporating hyperparameter tuning might limit the comprehensiveness of the findings. The lack of comparison with other advanced classification techniques and the absence of model optimization through techniques such as Grid Search suggests that the potential for improving model performance and achieving more nuanced insights was not fully explored. Thus, while the study demonstrates Random Forest's potential, it leaves room for further investigation into diverse algorithms and advanced tuning methods to address these limitations and enhance classification accuracy.

Wang [19] explores the application of Vision Transformer (ViT) models for mushroom species classification, comparing them with Convolutional Neural Network (CNN) architectures such as VGG, ResNet, and Xception. The study demonstrates that the ViT-L/32 model significantly outperforms the CNN models, achieving an accuracy of 95.97% and an area under the curve (AUC) of 99.01%. This improvement addresses the limitations of CNNs in distinguishing mushroom species with similar morphologies. The research also highlights challenges in mushroom classification, such as morphological variability and the importance of data visualization for interpretability. Despite these advancements, the study notes limitations regarding data quality and model generalization, suggesting areas for further research to enhance model accuracy and interpretability.

Essa and Dhanalakshmi [20] investigated the classification of edible and poisonous mushrooms using various machine learning algorithms, including CART, SVM, and K-Nearest Neighbors. Their study aimed to evaluate the performance of these models through metrics such as accuracy, precision, recall, and F1 score. The results demonstrated that the Random Forest algorithm achieved the highest accuracy of 99.63%, outperforming other models like Decision Trees and K-Nearest Neighbors, which had 97.60% and 98.33%, respectively. The study highlighted the importance of feature selection and data preprocessing for improving classification accuracy and noted limitations such as dataset constraints that could impact the model's generalization. This research contributes to understanding the effectiveness of machine learning algorithms in mushroom classification and suggests further exploration in this field.

Ortiz-Letechipia et al. [21] examined the performance of mushroom classification methods using the UCL Machine Learning Mushroom dataset, utilizing features selected through LASSO and GALGO. The study evaluated several models, including XGBoost, KNN, and Logistic Regression. With features selected by LASSO, the XGBoost model achieved an AUC of 0.999, a sensitivity of 0.989, and a specificity of 0.977. KNN and Logistic Regression models also showed competitive results, with AUCs of 0.976 and 0.960, respectively, and notable sensitivity and specificity. Conversely, with features selected by GALGO, XGBoost attained an AUC of 0.999, sensitivity of 0.998, and specificity of 0.736. KNN and Logistic Regression had AUCs of 0.998 and 0.736, with sensitivities of 0.989 and 0.702 for KNN and 0.987 and 0.573 for Logistic Regression. Wang identified that while XGBoost demonstrated superior performance in terms of AUC, sensitivity, and specificity, other models showed limitations, particularly in sensitivity and specificity. These limitations might impact the applicability of the models in real-world scenarios that require a balance between sensitivity and specificity.

Ketwongsa et al. [22] focused on developing a deep-learning model for classifying poisonous and edible mushrooms using an enhanced AlexNet architecture. Their proposed model achieved a high accuracy of 98.50% on the mushroom dataset and demonstrated significantly faster testing times compared to other pre-trained models such as ResNet-50 and GoogLeNet. This model also effectively reduced training and testing durations while maintaining high performance, making it a valuable tool for mushroom classification. The study compared the model with various CNN architectures, highlighting its efficiency in both speed and accuracy. The authors recommended expanding the dataset to include more mushroom varieties and diverse backgrounds to enhance the model's coverage and applicability. Identified limitations include the small dataset size and limited species focus, which could impact the model's generalization to real-world scenarios. This research contributes significantly to addressing the challenges of identifying poisonous and edible mushrooms and provides a foundation for further research in mushroom classification.

3. Proposed method

The method proposed in this research consists of five stages: Exploratory Data Analysis (EDA), Data Preprocessing, Classification Modeling, Performance Evaluation, and Hyperparameter Tuning.

3.1 Dataset

The dataset used in this research is Mushroom Classification data, accessed at https://www.kaggle.com/datasets/uciml/mushroom-classification. There are 23 attributes, 22 of which are feature attributes, and one is a class attribute. The class consists of 2 labels: edible and poisonous. The dataset is distributed with 4,208 records labeled as edible and 3,916 records labeled as poisonous. The attributes included in the dataset are class, cap shape, cap surface, cap color, bruises, odor, gill attachment, gill spacing, gill size, gill

color, stalk shape, stalk root, stalk surface above ring, stalk surface below ring, stalk color above ring, stalk color below ring, veil type, veil color, ring number, ring type, spore print color, population, and habitat. The sample of the dataset is presented in Table 1.

Table 1 Mushroom classification data sample

No	Class	Cap shape	Cap surface	…..	Spore print color	Population	Habitat
1	poisonous	convex	smooth	….	black	scattered	urban
2	edible	convex	smooth	….	brown	numerous	grasses
3	edible	bell	smooth	….	brown	numerous	meadows
4	poisonous	convex	scaly	….	black	scattered	urban
5	edible	convex	smooth	….	brown	abundant	grasses
6	edible	convex	scaly	….	black	numerous	grasses
7	edible	bell	smooth	….	black	numerous	meadows
8	edible	bell	scaly	….	brown	solitary	meadows
9	poisonous	convex	scaly	….	black	several	grasses
10	edible	bell	smooth	….	black	solitary	meadows
	….	….	….	….	….	….	….
8115	poisonous	flat	scaly	….	white	clustered	woods
8116	edible	convex	smooth	….	orange	numerous	leaves
8117	poisonous	knobbed	scaly	….	white	numerous	meadows
8118	poisonous	knobbed	smooth	….	white	numerous	woods
8119	poisonous	knobbed	scaly	….	white	numerous	woods
8120	edible	knobbed	smooth	….	buff	clustered	leaves
8121	edible	convex	smooth	….	buff	clustered	leaves
8122	edible	fibrous	smooth	….	buff	clustered	leaves
8123	poisonous	knobbed	scaly	….	white	numerous	leaves
8124	edible	convex	smooth	….	orange	clustered	leaves

This dataset is reasonably representative of various mushrooms as it contains physical descriptions of 23 species of mushrooms from the Agaricus and Lepiota families, as described in The Audubon Society Field Guide to North American Mushrooms (1981). Although no column specifically lists species names, the data provides various characteristics supporting accurate classification. Each species in the dataset is identified as an edible mushroom, a poisonous mushroom, or a mushroom that is not known to be edible and not recommended. This last category is combined with the poisonous mushroom category in the analysis.

The sample dataset shows some mushroom data that affects the mushroom classification. Some of the data shows several things about the mushroom, such as the shape of the cap, whether it is shaped like a cone, bell, concave, or flat. The data also show whether the surface cap is smooth, scaly, or fibrous. These data will be used to predict the mushroom so that the model can classify whether the mushroom is edible or poisonous. The research flow diagram is visualized in Figure 1.

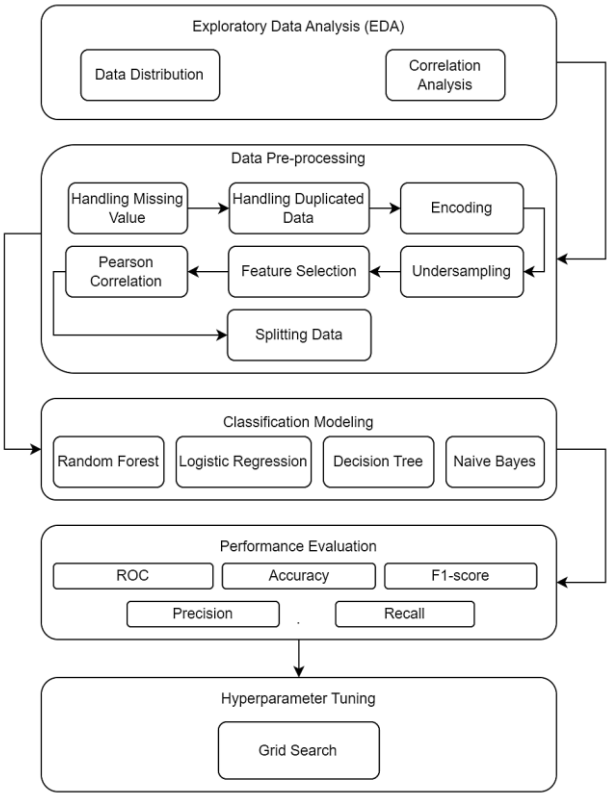

Figure 1 The research flow diagram

The research begins with visual analysis at the Exploratory Data Analysis (EDA) stage, and then the data is further processed at the data preprocessing stage. The processed data is then trained using several machine-learning algorithms. Researchers used ROC, accuracy, F1-score, precision, recall, and support metrics to evaluate the model's performance.

3.2 Exploratory data analysis (EDA)

At this stage, researchers explore the data to get an initial dataset analysis. The analysis involved visualizing the data as needed, with researchers focusing on various aspects such as data distribution and correlation analysis. The researchers used the correlation module of the Matplotlib library to visualize the data and the `pyspark.ml.stat` module to calculate the correlation between the two variables. In addition, we utilized the Seaborn and Pyplot modules for data visualization, creating heatmap-like plots. Combining these three modules allowed the researcher to calculate and visualize the correlation matrix effectively.

The researcher obtains descriptive statistical information for each numeric column in the dataset. Numeric features are identified by selecting columns with 'int' or 'double' data types. Then, researchers use the summary method to generate descriptive statistics, such as the number of data, mean, standard deviation, minimum value, quartile, and maximum value. Next, researchers continued the analysis by conducting correlation analysis to understand the relationship between numerical variables in the dataset. The researchers used a correlation matrix that can measure the extent of the relationship between variables. This analysis provides insight into the correlation pattern, whether positive, negative, or even no correlation between the variables. This information can be an essential foundation for feature selection in model development or further decision-making strategies.

3.3 Data preprocessing

At this stage, researchers applied a series of methods to prepare the data before it was used in modeling. The process starts with handling missing values, duplicated data, encoding, undersampling, and feature selection. These five methods were applied sequentially to ensure optimal data quality. In the first step, the researchers checked for missing values in the dataset. Researchers found no missing values in this dataset, so no deletion or replacement of empty data was required. Handling missing values is done to improve the accuracy of data analysis and ensure model quality and more reliable decision-making. The correlation between attributes on the mushroom dataset is presented in Figure 2.

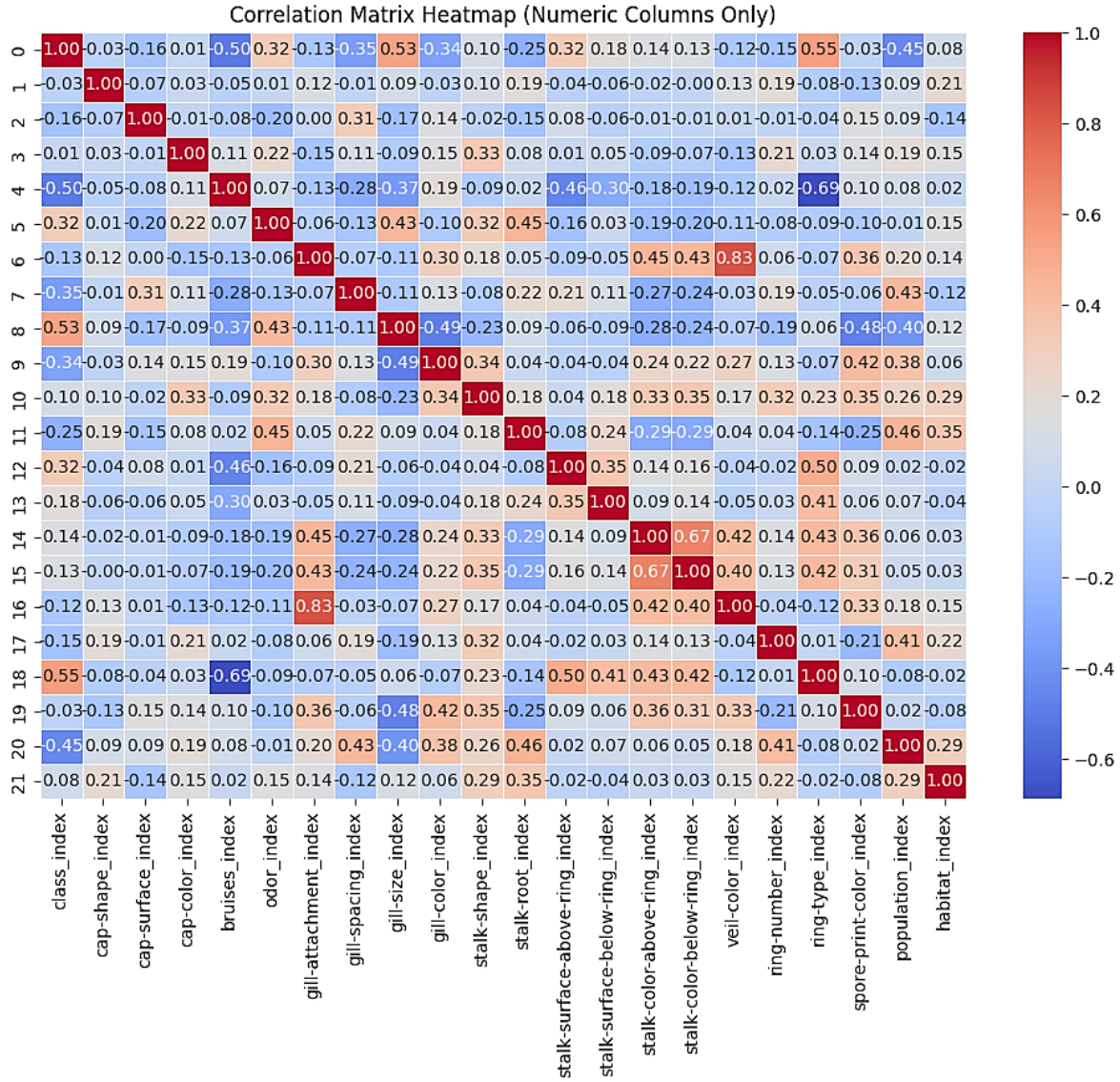

Figure 2 Correlation between attributes on the mushroom dataset

The researcher also checked for duplicate data and found no duplicate data, so there was no need to remove duplicate data. Removal of duplicate data is essential because duplicates can lead to redundancy of information in the dataset, and this can cause the model to capture the same pattern repeatedly (overfitting), affecting the quality of the model performance.

Researchers continued encoding to convert data with a string into a numeric data type. Some attributes in the dataset were identified as having string data types, so encoding labels were applied to convert them into numeric data types. Encoding techniques are significant in machine learning because machine learning algorithms and models can only process data numerically. Encoding techniques ensure that all information in the dataset can be converted to a numerical representation, allowing the model to understand and process the data effectively.

A measure used to reduce the negative impact of data imbalance is oversampling. Oversampling is a machine-learning approach that may successfully handle issues associated with class imbalance [23]. It effectively addresses the problem of class imbalance by increasing the number of samples in the minority class to balance it with the majority class. When the number of samples between the majority and minority classes is unbalanced, the model tends to over-predict the majority class, ignoring patterns that may exist in the minority class. Using oversampling techniques, particularly SMOTE or Synthetic Minority Over-sampling Technique, allows the model to learn patterns in the minority class more effectively, thereby improving prediction accuracy. By oversampling the minority class, the model becomes more balanced in handling the data and can reduce the risk of overfitting, which can occur if the model focuses too much on the majority class. SMOTE also allows the model to recognize patterns in the minority class better, resulting in more accurate and reliable model results.

The researcher can select features by identifying variables that correlate significantly with the target variable. The correlation threshold is predefined, and numerical features that have a correlation exceeding the threshold are considered relevant. The feature selection results will display the top five features that are considered relevant. The researcher moves on to the Pearson correlation stage from the feature selection results to explore the relationship between variables. Pearson correlation is calculated between numerical features and target variables. This step measures the association level or linear relationship between each feature and the target variable. Pearson correlation provides a value between -1 and 1, where a positive value indicates a positive relationship, a negative value indicates a negative relationship, and a zero value indicates no correlation. The results of this Pearson correlation calculation provide further insight into how strong or weak the relationship is between specific features and the target variable, assisting researchers in selecting the most relevant attributes for further analysis and establishing accurate classification models.

In the last stage, the data that has been prepared through feature selection and Pearson correlation calculation is divided into three subsets: training data (train), testing data (test), and validation data (validation). The data division process is performed randomly using the randomSplit function, where the data will be allocated to training, testing, and validation with proportions of 50%, 25%, and 25%, respectively. Seeds also ensure reproducibility in data division so that the results can be replicated consistently. Researchers can directly generate the amount of data in each subset, which helps understand the proportion of data used to train the model and test the model's performance. The amount of data in each subset shows how representative the overall dataset's training, testing, and validation data is. This ensures that data sharing is done correctly before the dataset is used in the model training and evaluation process.

3.4 Classification modeling

In this step, modeling is done by applying machine learning algorithms to the dataset and storing it in a variable. The process of training the model using data that has been labeled gives the model the ability to recognize patterns in the dataset. Data diversity contributes to the model's ability to make good predictions on various data types during testing. The condition of the data also has a significant effect on the results of model training. Good quality data will result in a model with optimal prediction performance. Conversely, insufficient data can result in a model that has unsatisfactory prediction performance.

Data balance is crucial to achieve optimal prediction results for each class. An imbalance in the data can cause the model to be more likely to identify a dominating class, resulting in predictions that focus more on that class. An unbalanced dataset is one with an uneven distribution of classes, which becomes a problem when the minority class is much smaller and the main class has a relatively high misclassification cost [24]. Apart from the amount of data, the numerical scale balance of the data is also a key factor. If the numerical scale of the data is unbalanced, the model may prioritize features with larger numerical scales while ignoring data with more minor numerical scales. Therefore, numerical scale imbalance in the data may affect the model's decision-making based on valuing features with larger scales.

The selection of the algorithm also has an impact on the performance of the model. Algorithms suitable for specific data types may result in superior model performance compared to other algorithms. Combining a good quality dataset and the proper algorithm selection will result in optimal model performance. In addition to data conditions and algorithm suitability, feature selection also affects model performance. Features that provide essential information to the model will contribute to more accurate classification.

3.5 Performance evaluation

Evaluation of model performance can be accomplished using various metrics. The confusion matrix provides an overview of the model's prediction results against the test data, showing the comparison between the original and predicted labels. Researchers can use the confusion matrix to analyze the number of correctly and incorrectly classified data in each class. Some of the performance evaluation metrics used in this study include recall, accuracy, precision, and F1-score are presented as follows:

$$recall = \frac{TP}{TP + FN} \tag{1}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{2}$$

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (4)$$

3.6 Hyperparameter tuning

The hyperparameter tuning stage is applied to the models used in this study using the grid search method. Researchers will build a pipeline for each model and involve the process of feature scale adjustment. The tuning process is performed by forming a parameter grid containing a combination of hyperparameter values tested for each model. This method explicitly initializes the CrossValidator using the k-Fold Cross Validation method, where test and validation data are used to evaluate model performance after hyperparameter tuning [25].

The tuning results are then evaluated using performance metrics such as ROC, accuracy, F1-score, precision, and recall. The confusion matrix, which shows the prediction distribution for all classes in one compact view [26], is used to provide a more detailed description of the model's prediction results. The evaluation aims to ensure that the model has optimal performance after hyperparameter tuning. Overall, the hyperparameter tuning method is a critical step in ensuring that the model performs optimally when handling the classification task that has been performed.

4. Results and analysis

This section presents the results and analysis of the application of the proposed method, including the researcher's data processing analysis through machine learning algorithm modeling. Elemental data analysis is necessary to understand the dataset used in modeling. In this study, some algorithms include Random Forest, Logistic Regression, Decision Tree, and Naive Bayes. Then, the researcher evaluated the performance of each model used by measuring parameters such as accuracy, precision, recall, F1-score, and ROC value. This evaluation provides deep insight into how well each algorithm can handle the given data and the extent of its ability to make predictions.

After obtaining the performance evaluation results using several algorithms, we continued the analysis by comparing the results to find the best and optimal values. Carefully, we selected four algorithms that showed the best performance to be deepened in the next stage, the hypertuning stage. The hypertuning process began with compiling a parameter grid, a list of values to be tested for each parameter. Then, we use optimization methods such as Grid Search to find the combination of parameters that gives the best results. After completing the hypertuning process, we re-evaluate the model and compare it with the results before and after the hypertuning stage. The results of this evaluation provide a more accurate picture of the model's performance on the new data, allowing researchers to quantify the improvements achieved through the hypertuning process. The comparison between the results before and after hyper-tuning provides valuable insight into the effectiveness of parameter adjustments on overall model performance.

4.1 Performance analysis

The confusion matrix provides a more in-depth look at how the model handles different types of errors. The confusion matrix displays in detail the number of True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN) generated by the model. The confusion matrix for the four classification algorithms used in this study is presented in Tables 2 and 3.

Table 2 Confusion matrix for model testing evaluation

Classification technique	TP	FN	FP	TN
Random forest	962	27	0	1055
Logistic Regression	843	105	212	884
Decision tree	1039	20	16	969
Naïve bayes	837	282	218	707

Table 3 Confusion matrix for model validation evaluation

Classification technique	TP	FN	FP	TN
Random forest	1001	23	0	941
Logistic Regression	807	90	194	874
Decision tree	991	15	10	949
Naïve bayes	799	261	202	703

Generally, there is good consistency between the testing and validation evaluation results. The superior performance of algorithms in the testing stage, such as Random Forest and Decision Tree, remained strong in the validation stage, showing good generalization ability. Conversely, algorithms such as Naïve Bayes that showed low performance in testing also showed similar weaknesses in validation. This suggests that the initial testing results are reliable and predictable, reflecting the stability of the model's performance when applied to new data.

4.2 Hyperparameter tuning

In this section, researchers focused on the models with the lowest evaluation results from the four previously used models, as summarized in Tables 4 and 5. Table 4 presents the validation evaluation metrics, showing that Logistic Regression and Naive Bayes had the lowest performance, with ROC values of 93.00 and 57.00, respectively. Table 5 shows the test evaluation metrics, where Logistic Regression and Naive Bayes had ROC values of 92.00 and 60.00, respectively. Random Forest and Decision Tree models achieved higher ROC values of 100.00 and 99.00 during validation and 100.00 and 98.00 during testing. Hyperparameter tuning aims to optimize these models further by finding the best-performing configuration through Grid Search.

Table 4 Comparison of validation evaluation between various machine learning techniques

Classification technique	Accuracy	Precision	Recall	F1-Score	ROC
Random forest	98.83	98.86	98.83	98.83	100.00
Logistic Regression	85.55	85.98	85.55	85.52	93.00
Decision tree	98.73	98.73	98.73	98.73	99.00
Naïve bayes	76.44	76.51	76.44	76.40	57.00

Table 5 Comparison of test evaluation between various machine learning techniques

Classification technique	Accuracy	Precision	Recall	F1-Score	ROC
Random forest	98.63	98.67	98.63	98.63	100.00
Logistic Regression	84.49	84.92	84.49	84.47	92.00
Decision tree	98.24	98.24	98.24	98.24	98.00
Naïve bayes	75.54	75.59	75.54	75.49	60.00

The Random Forest model has been optimized through hyperparameter tuning to achieve optimal performance. Grid search was used to test various parameter combinations, including the number of trees (numTrees) tested at 10 and 20 and the maximum depth of trees (maxDepth) tested at values of 5 and 10. In addition, the bootstrap parameter (bootstrap=True) was used to increase the variation between trees, and the feature subset strategy (featureSubsetStrategy=auto) was set to allow the model to select the best features automatically. The seed value (seed=5344069310727965689) ensures the reproducibility of the model results so that experiments can be repeated with consistency.

The Decision Tree was also tuned to the best parameters through grid search. The tested parameters include the maximum depth of the tree (maxDepth) at 5 and 10 and the minimum information gain (minInfoGain) set to 0.0. In addition, node caching was turned off to maintain data consistency during tree-building. The seed value (seed=-1559179628730338361) ensures that the model results can be consistently reproduced. This hyperparameter tuning is designed to optimize the structure and predictive ability of the Decision Tree model, improving its ability to capture complex patterns in the data.

Logistic Regression has been optimized through hyperparameter tuning, including testing parameter configurations such as regParam, elasticNetParam, maxIter, tol, and fitIntercept. The regParam parameter was tested at values of 0.01, 0.1, and 0.5, while elasticNetParam was tested at 0.0, 0.2, 0.5, and 0.8. The maximum number of iterations (maxIter) was tested at 50, 100, and 150, with rejects at 1e-06, 1e-05, and 1e-04. The fitIntercept parameter was tested for True and False values. Grid search used CrossValidator to select the best parameter combination based on model performance on validation data.

The Naive Bayes model was adjusted with smoothing and modelType parameters. The smoothing parameter was tested at values of 0.1, 0.5, 1.0, 1.5, and 2.0, while the modelType was tested at 'multinomial' and 'bernoulli'. Naive Bayes with modelType 'bernoulli' requires feature binarization for a suitable representation. CrossValidator was used to select the best parameter configuration based on model performance on validation data.

The hyperparameter tuning stage adjusts certain parameters in the model to improve its accuracy, precision, and overall performance. Through this tuning, it is expected that the model will be able to classify data better, both in terms of increasing the detection of positive cases (True Positives) and reducing classification errors, such as False Positives and False Negatives. The confusion matrix results for each algorithm after applying hyperparameter tuning, which includes results for testing and validation evaluation, are presented in Tables 6 and 7.

Table 6 Confusion matrix for model testing evaluation with hyperparameter tuning

Classification technique	TP	FN	FP	TN
Random forest	1055	11	0	978
Logistic Regression	937	106	118	883
Decision tree	1055	11	0	978
Naïve bayes	837	49	218	940

Table 7 Confusion matrix for model validation evaluation with hyperparameter tuning

Classification technique	TP	FN	FP	TN
Random forest	1001	11	0	953
Logistic Regression	890	92	111	872
Decision tree	1000	11	1	953
Naïve bayes	799	38	202	926

Overall, the application of hyperparameter tuning has improved the performance of the models, especially for Random Forest and Decision Tree, which showed consistent and optimal results in both evaluations. Logistic Regression and Naïve Bayes also slightly improved, although they still lagged compared to the other two models. These results show that parameter tuning can help improve the accuracy and generalization ability of models. However, its effectiveness may vary depending on the complexity and characteristics of the algorithms used.

Table 8 Comparison of optimal model performance with Hyperparameter Tuning (validation)

Classification technique	Accuracy	Precision	Recall	F1-Score	ROC
Random Forest (RF)	98.83	98.86	98.83	98.83	100.00
RF + hyperparameter tuning	99.44	100.00	98.86	99.43	99.98
Decision Tree (DT)	98.73	98.73	98.73	98.73	99.00
DT + Hyperparameter tuning	99.39	99.90	98.86	99.37	99.77

Table 8 compares optimal model performance before and after hyperparameter tuning using validation data. After hyperparameter tuning, high-performing algorithms, such as Random Forest and Decision Tree, show accuracy, precision, recall, and F1-score improvements. For Random Forest, accuracy increases from 98.83% to 99.44%, precision rises from 98.86% to 100.00%, recall improves from 98.83% to 98.86%, and F1-score grows from 98.83% to 99.43%. However, the ROC value shows a slight decrease from 100.00% to 99.98%, which may reflect a shift in the model's focus on classifying specific data points. Similarly, the Decision Tree shows enhanced performance with accuracy increasing from 98.73% to 99.39%, precision from 98.73% to 99.90%, recall from 98.73% to 98.86%, and F1-score from 98.73% to 99.37%, while the ROC value shows a slight decrease from 99.00% to 99.77%.

Table 9 Comparison of suboptimal model performance with Hyperparameter Tuning (validation)

Classification technique	Accuracy	Precision	Recall	F1-Score	ROC
Logistic Regression (LR)	85.55	85.98	85.55	85.52	93.00
LR + hyperparameter tuning	89.67	88.71	90.46	89.57	95.00
Naïve Bayes (NB)	76.44	76.51	76.44	76.40	57.00
NB + Hyperparameter tuning	87.79	82.09	96.06	88.53	97.00

Table 9 shows the performance of models initially exhibiting lower performance metrics, both before and after hyperparameter tuning. Logistic Regression shows notable improvements, with accuracy increasing from 85.55% to 89.67%, precision rising from 85.98% to 88.71%, recall improving from 85.55% to 90.46%, and F1-score growing from 85.52% to 89.57%. The ROC value also increases from 93.00% to 95.00%. Naïve Bayes, which starts with the lowest performance, shows substantial gains: accuracy rises from 76.44% to 87.79%, precision from 76.51% to 82.09%, recall from 76.44% to 96.06%, and F1-score from 76.40% to 88.53%. The ROC value shows a dramatic increase from 57.00% to 97.00%. These results show that hyperparameter tuning significantly enhances model performance, particularly for previously underperformed models.

Table 10 Comparison of optimal model performance with Hyperparameter Tuning (test)

Classification technique	Accuracy	Precision	Recall	F1-Score	ROC
Random Forest (RF)	98.63	98.67	98.63	98.63	100.00
RF + hyperparameter tuning	99.46	100.00	98.89	99.44	99.98
Decision Tree (DT)	98.24	98.24	98.24	98.24	98.00
DT + Hyperparameter tuning	99.46	100.00	98.89	99.44	99.76

Table 10 compares model performance before and after hyperparameter tuning on the test data. After hyperparameter tuning, optimal performance algorithms such as Random Forests and Decision Trees improve key performance metrics. In Random Forest, accuracy increased from 98.63% to 99.46%, precision from 98.67% to 100.00%, recall from 98.63% to 98.89%, and F1-score increased from 98.63% to 99.44%. However, the ROC value slightly decreased from 100.00% to 99.98%. This slight decrease may be due to the change in the model's focus on classifying the data, which may affect the model's ability to distinguish between classes. In Decision Tree, accuracy increased from 98.24% to 99.46%, precision from 98.24% to 100.00%, recall from 98.24% to 98.89%, and F1-score from 98.24% to 99.44%, with a slight decrease in ROC value from 98.00% to 99.76%. Hyperparameter tuning on both algorithms improved the model's overall performance, although there was a slight compromise on the class separation ability as measured by the ROC.

Table 11 Comparison of suboptimal model performance with Hyperparameter Tuning (test)

Classification technique	Accuracy	Precision	Recall	F1-Score	ROC
Logistic Regression (LR)	84.49	84.92	84.49	84.47	92.00
LR + hyperparameter tuning	89.04	88.21	89.28	88.74	94.00
Naïve Bayes (NB)	75.54	75.59	75.54	75.49	60.00
NB + Hyperparameter tuning	86.94	81.17	95.05	87.56	96.00

Table 11 shows the initially suboptimal performance of the model before and after hyperparameter tuning on the validation data. Logistic Regression experienced significant improvement, with accuracy rising from 84.49% to 89.04%, precision increasing from 84.92% to 88.21%, recall improving from 84.49% to 89.28%, and F1-score growing from 84.47% to 88.74%. The ROC value also increased from 92.00% to 94.00%. Naïve Bayes, which initially showed the lowest performance, experienced substantial improvement: accuracy rose from 75.54% to 86.94%, precision from 75.59% to 81.17%, recall from 75.54% to 95.05%, and F1-score from 75.49% to 87.56%. The ROC value increased drastically from 60.00% to 96.00%. Hyperparameter tuning improved the overall performance of both models, especially for Naïve Bayes, showing that parameter tuning can significantly improve prediction and classification capabilities, even for previously sub-optimal models.

Furthermore, the comparison between the proposed and existing schemes is shown in Table 12. Paudel and Bhatta [18] utilized the UCI Machine Learning Repository dataset without separate validation, which could limit generalization. Wang [19] applied Vision Transformer with a potentially non-standardized dataset. Essa and Dhanalakshmi [20] presented results from various algorithms

without detailed validation information. Ortiz-Letechipia et al. [21] used the UCL dataset with limited feature selection. Ketwongsa et al. [22] focused on deep learning models with a small dataset. In comparison, this research offers advantages by using a more representative dataset of 23 mushroom species with clear categories and implementing a systematic approach to data splitting for validation and testing, thereby enhancing the reliability and generalizability of the model compared to other studies.

Table 12 Comparison between the proposed method and the existing schemes in mushroom classifications

Scheme	Best Model	Accuracy	Precision	Recall	F1-Score	ROC
Paudel and Bhatta [18]	RF	100.00	100.00	100.00	100.00	-
Wang [19]	ViT-L/32	95.97	95.69	95.92	95.77	-
Essa and Dhanalakshmi [20]	RF	99.63	99.60	-	99.60	99.01
Ortiz-Letechipia et al. [21]	XGBoost	-	-	-	-	99.90
Ketwongsa et al. [22]	ResNet-50	96.50	-	-	-	-
Proposed Method	RF (Validation)	99.44	100.00	98.86	99.43	99.98
	RF (Test)	99.46	100.00	98.89	99.44	99.98
	DT (Validation)	99.39	99.90	98.86	99.37	99.77
	DT (Test)	99.46	100.00	98.89	99.44	99.76

Random Forest with hyperparameter tuning significantly improved both validation and test data. In validation, accuracy increased from 98.83% to 99.44%, precision rose from 98.86% to 100.00%, recall improved from 98.83% to 98.86%, and F1-score grew from 98.83% to 99.43%, with a slight ROC decrease from 100.00% to 99.98%. Similarly, on test data, accuracy increased from 98.63% to 99.46%, precision rose from 98.67% to 100.00%, recall improved from 98.63% to 98.89%, and F1-score increased from 98.63% to 99.44%, with a minor ROC decrease from 100.00% to 99.98%. These results highlight that hyperparameter tuning significantly enhanced the Random Forest model's performance, particularly in precision and overall accuracy, despite a slight trade-off in class separation capability as indicated by the ROC. Random Forest with hyperparameter tuning proved to be better in mushroom classification, especially when considering the importance of precision in classification decision-making.

In mushroom classification, accuracy, and precision are essential but serve different purposes. Accuracy measures how often the model correctly classifies a mushroom as edible or poisonous. Precision measures how often a mushroom classified as edible is edible. This is very important to prevent misclassifying poisonous mushrooms as edible, which can have serious consequences.

Based on the analysis, precision is more important than accuracy in mushroom classification. Misclassifying poisonous mushrooms as edible mushrooms has a more severe impact than misclassifying edible mushrooms as poisonous mushrooms. In this case, the Decision Tree without hyperparameter tuning shows slightly higher precision than the one with hyperparameter tuning. These results show that although the Decision Tree with hyperparameter tuning has higher accuracy, recall, F1-score, and ROC compared to the Decision Tree without hyperparameter tuning if the main goal is to minimize the risk of misclassifying poisonous mushrooms as edible mushrooms, the Decision Tree model without hyperparameter tuning may be preferred even though the other metrics slightly decreased after tuning. This emphasizes the importance of precision in critical applications such as food safety.

5. Conclusions

This research demonstrates that hyperparameter optimization through Grid Search significantly improves the performance of classification models, particularly for algorithms that are already optimal, such as Random Forest and Decision Tree. In validation data, Random Forest shows an increase in accuracy from 98.83% to 99.44%, precision from 98.86% to 100.00%, recall from 98.83% to 98.86%, and F1-score from 98.83% to 99.43%, despite a slight decrease in ROC from 100.00% to 99.98%. Similarly, Decision Tree displays clear improvements, with accuracy rising from 98.73% to 99.39%, precision from 98.73% to 99.90%, recall from 98.73% to 98.86%, and F1-score from 98.73% to 99.37%, while ROC value slightly decreases from 99.00% to 99.77%. The test data shows similar results, where Random Forest achieves an accuracy of 99.46%, precision of 100.00%, recall of 98.89%, and F1-score of 99.44%, with ROC slightly decreasing from 100.00% to 99.98%. Decision Tree records an accuracy of 99.46%, precision of 100.00%, recall of 98.89%, and F1-score of 99.44%, with ROC decreasing from 98.00% to 99.76%.

Conversely, the previously suboptimal algorithms, Logistic Regression and Naïve Bayes, show significant performance improvements after hyperparameter tuning. On validation data, Logistic Regression's accuracy increased from 84.49% to 89.04%, precision from 84.92% to 88.21%, recall from 84.49% to 89.28%, and F1-score from 84.47% to 88.74%, with ROC rising from 92.00% to 94.00%. Naïve Bayes exhibits a substantial improvement, with accuracy rising from 75.54% to 86.94%, precision from 75.59% to 81.17%, recall from 75.54% to 95.05%, and F1-score from 75.49% to 87.56%, while ROC increases dramatically from 60.00% to 97.00%. Similar improvements are observed in test data for both models. Despite the significant enhancements in Logistic Regression and Naïve Bayes, Random Forest remains the most accurate model for mushroom classification, particularly in food safety, due to its high accuracy and precision.

6. Acknowledgements

This research is supported and fully funded by Universitas Amikom Yogyakarta.

7. References

[1] Meenu M, Xu B. Application of vibrational spectroscopy for classification, authentication and quality analysis of mushroom: a concise review. Food Chem. 2019;289:545-57.

[2] Tu D, Wu F, Lei Y, Xu J, Zhuang W, Zhao Y, et al. Analysis of differences in flavor attributes of soups: a case study on shiitake mushrooms dried from different drying techniques. J Food Compos Anal. 2024;131:106228.

[3] Tongcham P, Supa P, Pornwongthong P, Prasitmeeboon P. Mushroom spawn quality classification with machine learning. Comput Electron Agric. 2020;179:105865.

[4] Brown AJP, Brown GD, Netea MG, Gow NAR. Metabolism impacts upon Candida immunogenicity and pathogenicity at multiple levels. Trends Microbiol. 2014;22(11):614-22.
[5] White J. New classification of mushrooms poisoning. Toxicol Anal Clin. 2018;30(3):157-8.
[6] Tao K, Liu J, Wang Z, Yuan J, Liu L, Liu X. ReYOLO-MSM: A novel evaluation method of mushroom stick for selective harvesting of shiitake mushroom sticks. Comput Electron Agric. 2024;225:109292.
[7] Nagulwar MM, More DR, Mandhare LL. Nutritional properties and value addition of mushroom: a review. Pharma Innov J. 2020;9(10):395-8.
[8] Tarawneh O, Tarawneh M, Sharrab Y, Husni M. Mushroom classification using machine-learning techniques. AIP Conf Proc. 2023;2979(1):030003.
[9] Farshbaf Aghajani P, Soltani Firouz M, Bohlol P. Revolutionizing mushroom identification: improving efficiency with ultrasound-assisted frozen sample analysis and deep learning techniques. J Agric Food Res. 2024;15:100946.
[10] Liu Q, Fang M, Li Y, Gao M. Deep learning based research on quality classification of shiitake mushrooms. LWT. 2022;168:113902.
[11] Özbay E, Özbay FA, Gharehchopogh FS. Visualization and classification of mushroom species with multi-feature fusion of metaheuristics-based convolutional neural network model. Appl Soft Comput. 2024;164:111936.
[12] Chun TH, Hashim UR, Ahmad S, Salahuddin L, Choon NH, Kanchymalay K. Efficacy of the image augmentation method using CNN transfer learning in identification of timber defect. Int J Adv Comput Sci Appl. 2022;13(5):107-14.
[13] Acuña-Zegarra MA, Santana-Cibrian M, Velasco-Hernandez JX. Modeling behavioral change and COVID-19 containment in Mexico: a trade-off between lockdown and compliance. Math Biosci. 2020;325:108370.
[14] Fadillah MI, Aminuddin A, Rahardi M, Abdulloh FF, Hartatik H, Asaddulloh BP. Diabetes diagnosis and prediction using data mining and machine learning techniques. 2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP); 2023 Dec 1-2; Yogyakarta, Indonesia. USA: IEEE; 2023. p. 110-5.
[15] Kusnawi K, Ipmawati J, Asadulloh BP, Aminuddin A, Abdulloh FF, Rahardi M. Leveraging various feature selection methods for churn prediction using various machine learning algorithms. Int J Informatics Vis. 2024;8(2):897-905.
[16] Akbar MI, Aminuddin A, Abdulloh FF, Rahardi M, Wahyuni SN, Asaddulloh BP. Comparison of machine learning techniques for heart disease diagnosis and prediction. 2023 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA);2023 Nov 14-15; Surabaya, Indonesia. USA: IEEE; 2023. p. 815-20.
[17] Ekatama RA, Rahardi M, Aminuddin A, Abdulloh FF. Sentiment analysis of electric vehicles in indonesia using support vector machine and naïve bayes. 2023 3rd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS); 2023 Dec 6-8; Bali, Indonesia. USA: IEEE; 2023. p. 120-5.
[18] Paudel N, Bhatta J. Mushroom classification using random forest and REP tree classifiers. Nepal J Math Sci. 2022;3(1):111-6.
[19] Wang B. Automatic mushroom species classification model for foodborne disease prevention based on vision transformer. J Food Qual. 2022;2022(1):1173102.
[20] Essa IA, Dhanalakshmi R. Machine learning-based classification of edible and poisonous mushrooms: a performance comparison. Int J Res Appl Sci Eng Technol. 2023;11:1364-70.
[21] Ortiz-Letechipia JS, Galvan-Tejada CE, Galván-Tejada JI, Soto-Murillo MA, Acosta-Cruz E, Gamboa-Rosales H, et al. Classification and selection of the main features for the identification of toxicity in Agaricus and Lepiota with machine learning algorithms. PeerJ. 2024;12:e16501.
[22] Ketwongsa W, Boonlue S, Kokaew U. A new deep learning model for the classification of poisonous and edible mushrooms based on improved AlexNet convolutional neural network. Appl Sci. 2022;12(7):3409.
[23] Mujahid M, Kına E, Rustam F, Villar MG, Alvarado ES, De La Torre Díez I. Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering. J Big Data. 2024;11:87.
[24] Vuttipittayamongkol P, Elyan E, Petrovski A. On the class overlap problem in imbalanced data classification. Knowl Based Syst. 2021;212:106631.
[25] Raikwal JS, Saxena K. Performance evaluation of SVM and K-nearest neighbor algorithm over medical data set. Int J Comput Appl. 2012;50(14):35-9.
[26] Heydarian M, Doyle TE, Samavi R. MLCM: Multi-label confusion matrix. IEEE Access. 2022;10:19083-95.