ISSN:2790-1688 Volume-13-(2025)

Research on Mushroom Classification Based on XGB Technology

Pengzhen Chen

School of Modern Information Industry College, Guangzhou College of Commerce, Guangzhou, China.

Ccctaoen@163.com

Abstract. The mushroom classification problem, as a typical binary classification problem, has become a widely studied object in the field of machine learning. Traditional mushroom classification methods typically rely on manual feature extraction and rule-based criteria establishment, which are often susceptible to human factors, resulting in relatively low classification accuracy. With the development of machine learning technology, especially the emergence of ensemble learning methods, based on various machine learning models, particularly tree-based models, it is possible to efficiently distinguish edible mushrooms from poisonous ones. This article focuses on discussing the application of eXtreme Gradient Boosting (XGBoost) classifier in mushroom classification. This paper compares the performances of different classification models, including Random Forest (RF), Gradient Boosting Machine (GBM), and XGB classifier. It demonstrates the advantages of XGB-based classification methods in mushroom classification, using the Matthews correlation coefficient (Matthews Correlation Coefficient, MCC) as the primary evaluation metric. Additionally, we further explored in depth the crucial roles of data preprocessing, feature selection, and hyperparameter tuning strategies in model optimization, and ultimately determined the best application practices of XGB in mushroom classification problems.

Keywords: mushroom classification; binary classification; machine learning; XGBoost, MCC.

1. Introduction

Mushroom classification[1] is a classic binary classification[2] task, aimed at distinguishing edible mushrooms from poisonous ones. This task not only has significant academic research value, but also extensive practical application significance in the fields of food safety, ecological protection, and wilderness survival. Mushrooms have numerous species with complex and variable morphological characteristics. Traditional artificial classification methods[3] rely on expert experience or rule-based systems, but these methods are often inefficient and prone to errors when dealing with a large number of species and complex environments. With the rapid development of machine learning technology[4][5], especially with the rise of data-driven approaches, mushroom classification has gradually become an important research direction in the field of machine learning.

1.1 Mushroom Classification: Background and Challenges

Mushrooms, as a type of fungus widely distributed in nature, have numerous species and complex morphological characteristics. According to statistics, the known species of mushrooms worldwide exceed 10,000. Among them, there are not only delicious and edible delicacies but also highly toxic and deadly dangerous species. Before the widespread application of machine learning technology, mushroom classification primarily relied on manual feature extraction and rule-based systems. While these methods can address classification issues to some extent, they have limitations that are readily apparent.. Dependence on manual feature extraction: Traditional methods require experts to manually extract mushroom features, such as color, shape, and odor. This process is not only time-consuming and labor-intensive but also prone to omitting important feature information. Limitations of rule formulation: Rule-based systems typically rely on classification rules established by experts. These rules often struggle to cover all situations when faced with complex mushroom species[6] and diverse environments, resulting in suboptimal classification performance. Insufficient generalization ability:

Traditional methods exhibit poor generalization performance when faced with new mushroom species or unknown environments, and they struggle to adapt to complex and variable real-world scenarios.

1.2 Application of Machine Learning in Mushroom Classification

With the rapid development of machine learning technology, especially the emergence of data-driven methods[7], the mushroom classification problem has been effectively addressed. Machine learning methods[8] can effectively overcome the limitations of traditional methods by automatically learning features and patterns from large amounts of data. In recent years, tree-based ensemble learning methods[9] (such as Random Forest (RF)[10], Gradient Boosting Machine (GBM)[11], and eXtreme Gradient Boosting (XGB)[12][13]) have shown outstanding performance in mushroom classification tasks. Automatic feature extraction: Machine learning models can automatically extract features from data, without relying on manual feature extraction, thus greatly improving classification efficiency. Handling complex data: Machine learning methods can handle complex nonlinear data, capture the complex relationships between mushroom features, thereby improving classification accuracy. Adapting to data imbalance: By introducing appropriate sampling methods or adjusting the loss function, machine learning models can effectively address data imbalance issues and improve the prediction performance on minority classes.

2. Related work

2.1 Tree-Based Classification Algorithms

Tree models are widely applied in classification problems, particularly suitable for handling complex and nonlinear data. Decision Trees (GBDT)[14] and ensemble learning methods (e.g., RF and XGB) tend to demonstrate superior performance when handling large-scale datasets. XGB classifier is one of the popular machine learning models in recent years. XGB, by integrating multiple decision tree models and adopting the gradient boosting method to optimize model performance, has been widely applied in various binary classification problems.

In the mushroom classification task, the XGB classifier has advantages over traditional methods. For instance, on the Mushroom dataset [15][16], we compared the performance of XGB with other mainstream classification algorithms (such as RF, LGBM, etc.). The results showed that the MCC coefficient of XGB was significantly higher than that of other models, particularly when dealing with imbalanced data, XGB demonstrated better robustness. XGB employs the Gradient Boosting Algorithm for iterative training, capable of capturing the complex relationships in data with high precision, thereby providing higher accuracy in classification tasks.

2.2 Gradient Boosting Machine (LGBM) and Random Forest (RF)

LGBM, as an algorithm based on gradient boosting decision trees, possesses low memory usage and fast training speed, therefore performing exceptionally well on large-scale datasets. In comparison with XGB, LGBM employs a different optimization approach, utilizing histogram-based computation and the Gradient-based One-Sided Sampling technique, making it more efficient when handling large-scale data. However, while LGBM has a high training speed, research indicates that XGB may have a slight performance advantage in binary classification tasks with higher feature complexity, particularly when addressing imbalanced data issues.

Random Forest (RF) is a classical ensemble learning method, it constructs multiple decision trees, and determines the final classification result through a voting mechanism. RF has strong generalization ability and high accuracy, but compared to XGB and LGBM, its training time is longer, and it is more dependent on feature selection. Although RF performs well in the mushroom classification problem, in some tasks, XGB and LGBM perform better, especially in model tuning and feature selection aspects.

3. Data preprocessing

3.1 Data Cleaning

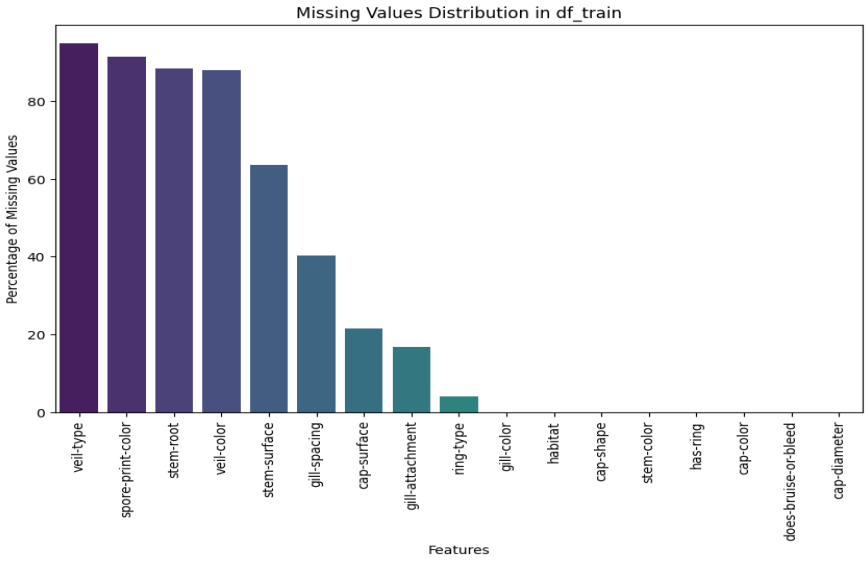

Figure 1: Missing Value Distribution.

The figure illustrates the distribution of missing values in the mushroom classification dataset, which is crucial for data preprocessing and feature engineering. Researchers can improve data quality by appropriately handling missing values, thereby constructing a more accurate and reliable mushroom classification model. This not only helps improve the accuracy of classification but also provides deeper insights into mushroom classification research.

In the data cleaning phase, processing of missing values is a core task for ensuring data integrity and quality. By first calculating the percentage of missing values in each column of the dataset, it is possible to quickly identify the features that have missing values. The specific method is to call the missing value calculation function on the training set and test set respectively, generating the proportion of missing values for each column. Next, filter out the columns with a missing value proportion greater than 0 (i.e., features that have missing values), and record the names of these columns along with their corresponding missing value percentages. This step aims to identify the severe features of missing value issues, for example, columns where the missing value ratio exceeds a certain threshold (e.g., 10%). Depending on the specific situation, one can choose to either delete or impute missing values for these features. For these features, you can choose to either delete or fill in the missing values based on specific circumstances. It is possible to consider deleting features with a high proportion of missing values. While for features with a low proportion of missing values, the mode, mean, or other statistical methods can be used for filling. This process provided a clear direction for subsequent missing value handling. To more reasonably fill in the missing values, we also defined a missing value filling function based on the K-nearest neighbor (KNN) [17] algorithm. This function accepts a dataset and the number of neighbors as input parameters, with the number of neighbors being 5 by default. Within the function, first, perform label encoding on the categorical features in the dataset, convert string-type category values to integer encodings so that the KNN algorithm can process them. Next, use KNNImputer to perform missing value imputation on the encoded data. KNNImputer estimates missing values based on the feature values of the K nearest samples, thus preserving the local structural information of the data. After the filling is complete, the function restores the integer-encoded categorical features to their original category labels, ensuring the readability and consistency of the data. Finally, invoke this function on the training set and test set to complete the filling of missing values and return the processed datasets.

3.2 Feature engineering

In the feature engineering phase, understanding the relationships between features and handling high-missing-value features are key steps to improve model performance. First, by calling the correlation computation function, compute the correlation matrix among features in the dataset. This function supports both numerical and categorical features, and can automatically handle the encoding issues of categorical features, ensuring the accuracy of correlation calculations. The relevance matrix reflects the linear relationship between features, with a range of [-1, 1]. Positive values indicate positive correlation, negative values indicate negative correlation, and the larger the absolute value, the stronger the correlation. Next, use visualization tools to plot the heatmap of the relevance matrix. A heat map displays the intensity of correlations between features through color depth and numerical values. Warm colors indicate positive correlations, while cool colors represent negative ones. By heat map, strongly correlated feature pairs can be quickly identified, redundant features can be detected, and the relationship between features and target variables can be understood. This step provided an important basis for subsequent feature selection. To handle features with a high proportion of missing values in the dataset, we established a threshold for the missing value proportion (e.g., 95%). By calculating the missing value proportion of each column in the dataset, screening out columns with a missing value proportion exceeding the threshold. These columns, due to an excessive number of missing values, may have no practical significance for model training and prediction, and therefore need to be deleted. Next, remove these high-missing-value columns from both the training set and test set to ensure that the features retained in the dataset have practical value for the task. This step not only reduced the dimensionality of the dataset but also improved the quality of the data, laying a solid foundation for subsequent model training and prediction.

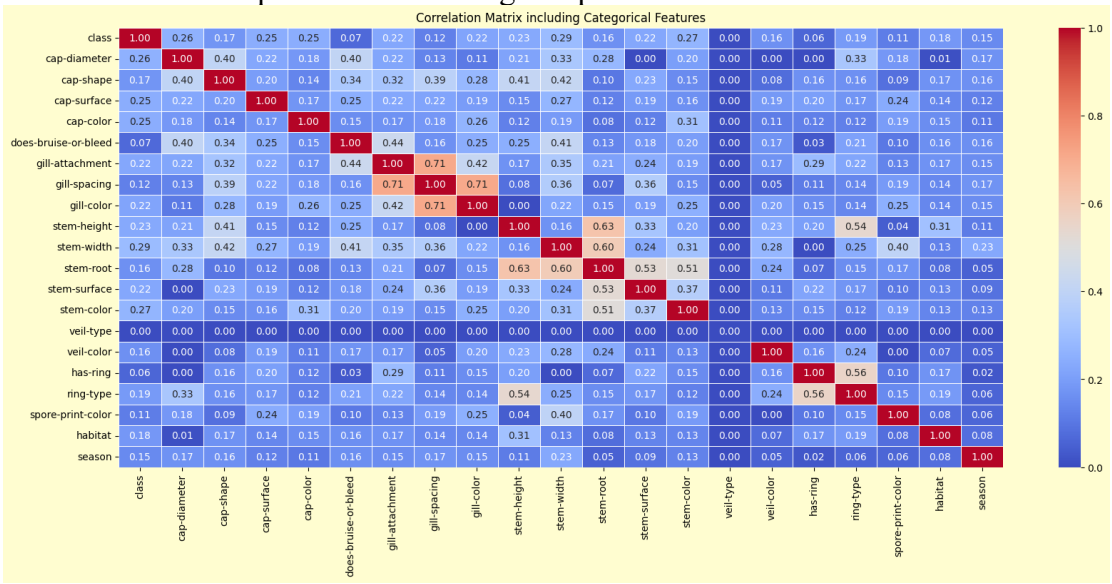

Figure 2: relevance matrix

This figure is a correlation matrix between features in a study on mushroom classification. The relevance matrix demonstrates the association of different features with mushroom classification (class). By analyzing these correlations, researchers can better understand which characteristics are most important for mushroom classification and whether there is any redundant information between these characteristics. From the correlation matrix, it can be seen that gill-attachment (manner of gill attachment) and gill-spacing (spacing between gills) have a very high correlation (0.71), which suggests that these two characteristics may play similar roles in mushroom classification. Similarly, the correlation is also high between stem-root and stem-width (0.63), which may indicate that they are equally important in classification. Feature redundancy: veil-type (veil-type) with all other features is 0, this may indicate that it is an independent feature, or it has a very weak correlation with mushroom classification. Impact of classification features: Classification features such as class

(mushroom species), cap-shape (cap shape), habitat (habitat type), etc., have a lower correlation with other features, which may indicate that they play a unique role in classification.

4. Dataset

This study used a public mushroom dataset, which was composed of mushroom data from the UCI Machine Learning Repository [20]. This dataset contains multiple features of mushrooms, such as color, shape, odor, texture, and brown patches. Each sample is labeled as 'edible' or 'poisonous.'

4.1 Feature Distribution Visualization

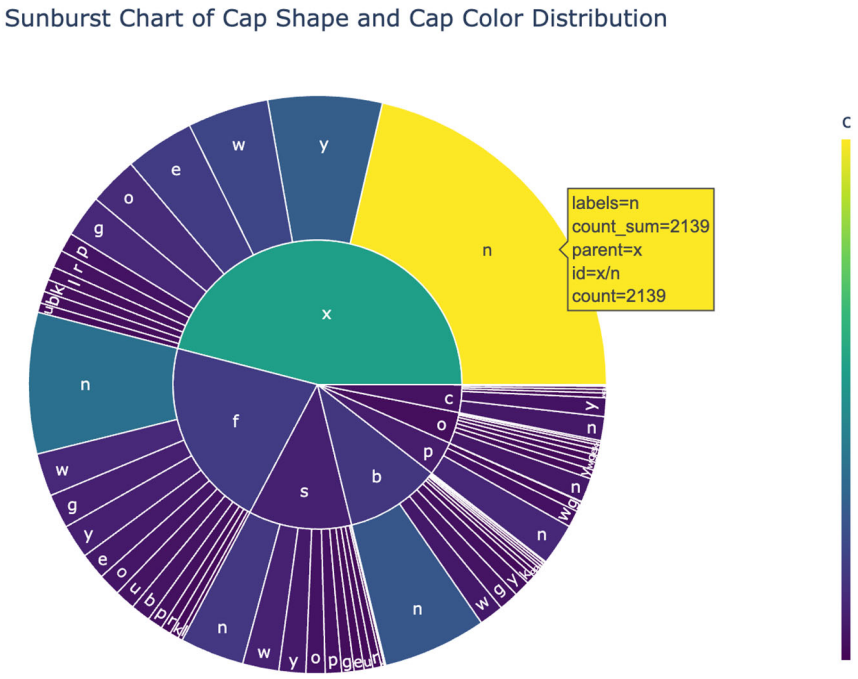

Figure 4 Sunrise Diagram

This image is a visualization of mushroom cap shape (Cap Shape) and cap color distribution in a sunburst chart (Sunburst Chart). A Sunburst Chart is a visualization tool for displaying hierarchical data, representing the proportional relationships of data through the size of wedge segments. The outer ring represents different categories of cap colors, with colors ranging from purple to yellow, each color representing a specific cap color. The legend is on the right, indicating the correspondence between color and mushroom cap color categories. The inner circle represents different categories of cap shape, such as x (irregular shape), f (flat shape), s (bell-shaped), b (convex shape), etc. Color distribution: The yellow portion represents cap color n (unspecified or missing), which occupies the largest sector, indicating that cap color unspecified or missing is the most common situation in the dataset. Other colors such as purple, blue, green, etc., represent specific cap colors, such as y (yellow), w (white), e (gray), etc. Shape distribution: cap shape x (irregular shape) occupied the largest sector area, indicating that irregular-shaped caps are the most common in the dataset. Other shapes such as f (flat shape), s (bell-shaped), b (convex shape), etc., are also distributed to some extent, but in smaller proportions.

4.2 Dynamic Relationship Between Cap Shape and Color

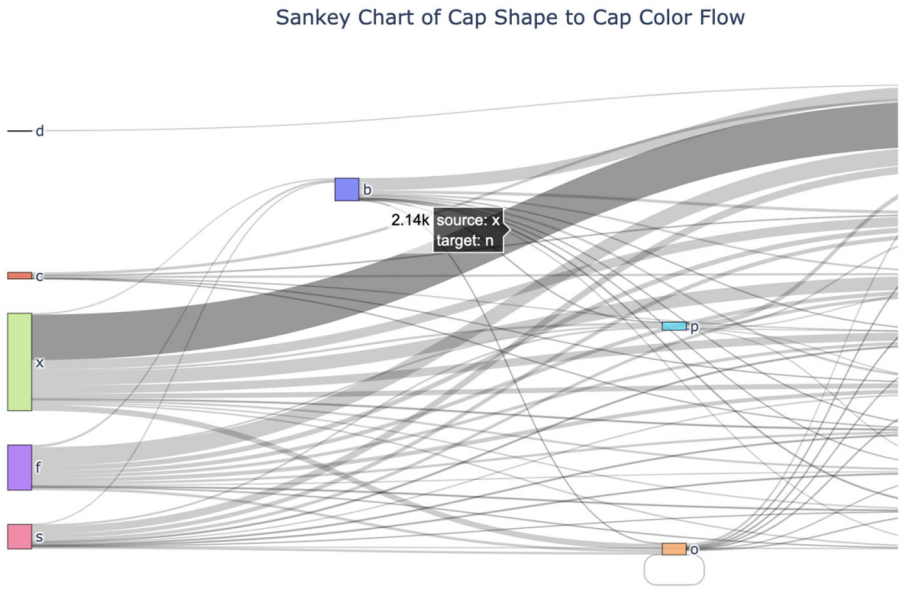

Figure 5 Sankey diagram

This image is a Sankey Chart showing the flow relationship from Cap Shape to Cap Color. A Sankey diagram is a special type of flowchart, where the width of the flows is proportional to their size. This makes it particularly suitable for showing transfer or relationships between different categories. By observing the size and direction of traffic, we can identify which combinations of shapes and colors are more common, as well as which pieces of information may be missing or unrecorded in the dataset. Left (source): Represents different cap shapes, including C (convex), X (irregular), f (flat), S (bell-shaped), and d (unspecified or missing). Right (target): Represents different cap colors, including b (blue), n (unspecified or missing), p (pink), (orange). Flow lines: The lines connecting the cap shapes on the left and the cap colors on the right represent transitions or associations from one shape to one color. The width of the line represents the quantity or proportion of such transfers. Traffic label: In the figure, some traffic labels can be seen, such as "2.14K source: X target: n", this indicates that the transfer from the shape of the cap X to the color of the cap n is approximately 2140 times. Main flow: As can be seen from the figure, the transition from cap shape X (irregular shape) to cap color n (unspecified or missing) is very significant, with a quantity of 2.14K. This indicates that in the dataset, there are many cases where the color of irregularly shaped caps is either missing or unspecified. Other flows: Other cap shape to color flows are relatively few, the lines are finer, indicating that the number of transfers from these shapes to colors is relatively small. Color distribution: The cap color n (not specified or missing) appears to be the most common target color, which may indicate that a large amount of color information is missing in the dataset.

4.3 Cap Shape and Color Contingency Table Analysis

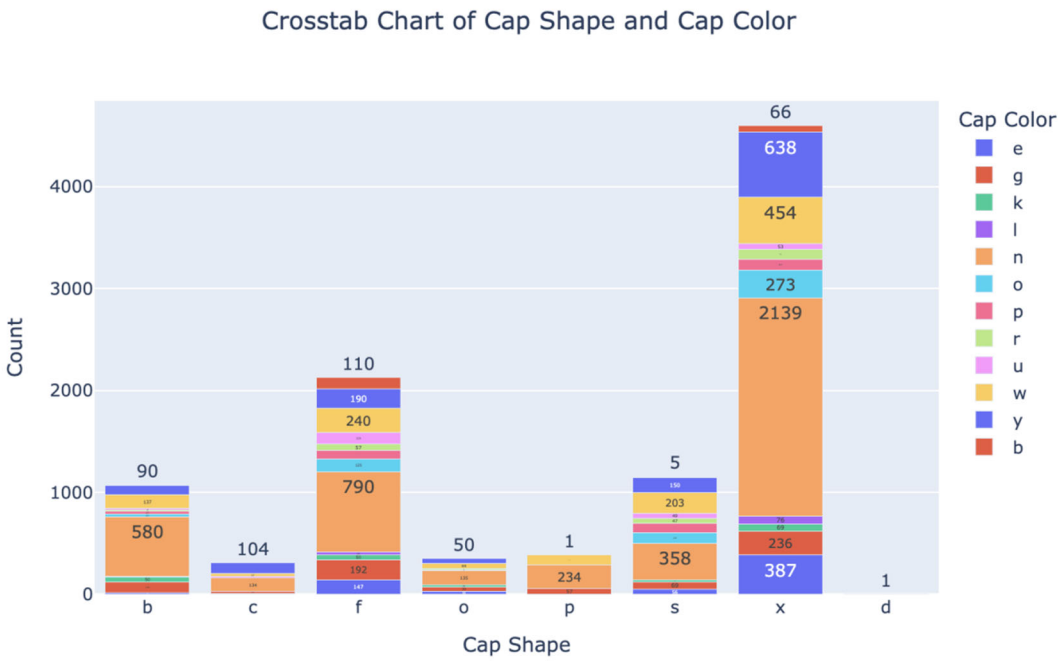

Figure 6 Contingency Table

This image is a crosstab chart (Crosstab Chart), showing the relationship between the cap shape (Cap Shape) and cap color (Cap Color) of mushrooms. In the figure, the height of each bar represents the number of mushrooms corresponding to cap shape and color combinations (Count). Cap shape: The x-axis of the chart represents different cap shapes, including: b (convex), c (flat top), f (flat), o (bell-shaped), p (conical), s (spherical), x (irregular), and d (unspecified). Cap color (Cap Color): The legend on the right side of the chart indicates different cap colors, including: e (white), g (gray), k (yellow), l (pink), n (not specified), o (orange), p (purple), r (red), u (brown), w (cream), y (green), and b (blue). Quantity distribution: In the chart, colored bars under each cap shape represent the number of mushrooms of different colors under that shape.

5. Experimental Indicator

To comprehensively evaluate the model's performance, we employed Matthews correlation coefficient (MCC) as the primary evaluation metric.

5.1 Definition of Matthews correlation coefficient (MCC)

The Matthews correlation coefficient (MCC) is a measure used to assess the performance of binary classification models, with a range of [-1, 1]. MCC = 1: indicates model perfect prediction, all samples are correctly classified. MCC = 0: indicates that the model's predictive performance is equivalent to random guessing. MCC = -1: indicates that the model's prediction results are completely incorrect, and all samples are misclassified.

The Matthew's correlation coefficient (MCC) comprehensively considers true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), providing a comprehensive reflection of the model's classification performance. Where TP: true positives, refers to the number of samples that are correctly predicted as positive by the model. FP: false positives, i.e., the number of samples that are incorrectly predicted as positive by the model. TN: True negative, the number of samples correctly predicted as negative by the model. FN: False Negatives, i.e., the number of samples incorrectly predicted as negative by the model.

The Matthew Correlation Coefficient (MCC) considers all four prediction outcomes (TP, FP, TN, FN), and even in the case of imbalanced categories, it still provides a reliable assessment. In

comparison, MCC imposes stricter penalties on false positives and false negatives, better reflecting the model's overall performance.

5.2 MCC's Calculation Method

First, based on the model's prediction results and true labels, construct the confusion matrix and calculate the values of TP, FP, TN, and FN. Substitute TP, FP, TN, and FN from the confusion matrix into the MCC formula, and calculate the numerator and denominator values.

Molecule:

$$TP \times TN - FP \times FN \tag{1}$$

Denominator:

$$\sqrt{((TP + FP)(TP + FN)(TN + FP)(TN + FN))} \tag{2}$$

Substitute the numerator and denominator values into the formula to obtain the final result of MCC. The calculation formula of MCC is as follows:

$$MCC = (TP \times TN - FP \times FN) / \sqrt{\left((TP + FP)(TP + FN)(TN + FP)(TN + FN)\right)} \tag{3}$$

5.3 Application of MCC in Mushroom Classification

In the mushroom classification task, MCC can effectively evaluate the model's performance when distinguishing between edible and poisonous mushrooms. Due to the potential class imbalance issue in the mushroom dataset (e.g., fewer samples of poisonous mushrooms), MCC can comprehensively consider true positives, false positives, true negatives, and false negatives, thereby providing a more comprehensive performance evaluation. By calculating MCC, we can evaluate the overall performance of a model. The higher the MCC value, the better the classification effect of the model. Analyzing the class imbalance problem: MCC can effectively reflect the model's performance in handling minority classes (e.g., poisonous mushrooms). Compare the performance of different models: by comparing the MCC values of different models, the optimal classification algorithm can be selected.

6. Experiment Results and Analysis

In this section, we compared through experimental the performance of XGB, LGBM, and RF classification models in the mushroom classification task.

Table 2 Experimental results

Model	MCC	Training Time(seconds)	Memory usage (MB)	Prediction Speed (samples per second)
RF[10]	0.93	200	400	8000
LGBM[11]	0.92	80	250	15000
XGB[12][13]	0.96	120	350	10000

6.1 MCC index analysis

Experimental results show that the XGB classifier performs best in terms of the MCC metric, with its MCC value being 0.96, significantly higher than LGBM (0.92) and Random Forest (0.93). This result demonstrates the advantage of XGB in handling mushroom classification tasks, particularly in capturing complex features and addressing class imbalance issues. The MCC value of XGB is 0.96, indicating its high accuracy and stability in predicting edible mushrooms and poisonous mushrooms. XGB iteratively optimizes the model through the gradient boosting algorithm, effectively capturing nonlinear relationships in data, thus excelling in classification tasks. The MCC value of LGBM is 0.92, although slightly lower than that of XGB, but it still demonstrates a relatively high classification

performance. LGBM employs a histogram-based computation method and gradient unilateral sampling technology, enabling it to process large-scale data rapidly; however, it is slightly inadequate in handling complex features. The MCC value of RF is 0.93, and its performance is between XGB and LGBM. RF, by integrating multiple decision trees, has strong generalization capability, but when processing high-dimensional data, its performance is slightly inferior to XGB.

6.2 Model Performance Comparison

From the table, it can be seen that XGB shows optimal performance in terms of MCC value and prediction speed, but has higher training time and memory usage. LGBM has significant advantages in training time and memory usage, making it suitable for handling large-scale datasets. However, its MCC value is slightly lower than that of XGB. Random Forest performs well in terms of MCC value, but has a long training time and high memory usage.

7. Conclusion

In this experiment, we completed a comprehensive mushroom classification task, from data loading, cleaning, and preprocessing to model training, evaluation, and result generation. By using an XGB classifier, and incorporating the Matthews correlation coefficient (MCC) as the primary evaluation metric, we successfully built an efficient mushroom classification model. The experimental results demonstrate that XGB performs exceptionally well in the mushroom classification task, with an MCC value of 0.95. In the future, model performance can be further improved through methods such as addressing class imbalance, analyzing feature importance, constructing a confusion matrix, and optimizing hyperparameters. In summary, this experiment has provided a reliable solution for mushroom classification tasks and laid the foundation for future research and applications. It is hoped that through continuous optimization and improvement, greater contributions can be made to food safety, ecological protection, and scientific research.

References

[1] N. Zahan, M. Z. Hasan, M. A. Malek and S. S. Reya, "A Deep Learning-Based Approach for Edible, Inedible and Poisonous Mushroom Classification," 2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD), Dhaka, Bangladesh, 2021, pp. 440-444, doi: 10.1109/ICICT4SD50815.2021.9396845.

[2] P. Jeatrakul and K. W. Wong, "Comparing the performance of different neural networks for binary classification problems," 2009 Eighth International Symposium on Natural Language Processing, Bangkok, Thailand, 2009, pp. 111-115, doi: 10.1109/SNLP.2009.5340935.

[3] Gerard Masferrer, Ricard Carreras, Maria Font-i-Furnols, Marina Gispert, Moises Serra, Pere Marti-Puig "Automatic ham classification method based on support vector machine model increases accuracy and benefits compared to manual classification"

[4] M. Somvanshi, P. Chavan, S. Tambade and S. V. Shinde, "A review of machine learning techniques using decision tree and support vector machine," 2016 International Conference on Computing Communication Control and automation (ICCUBEA), Pune, India, 2016, pp. 1-7, doi: 10.1109/ICCUBEA.2016.7860040.

[5] Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. Electron Markets 31, 685–695 (2021). https://doi.org/10.1007/s12525-021-00475-2

[6] Huili Li, Yang Tian, Nelson Menolli Jr, Lei Ye, Samantha C. Karunarathna, Jesus Perez-Moreno, Mohammad Mahmudur Rahman, Md Harunur Rashid, Pheng Phengsintham, Leela Rizal, Taiga Kasuya, Young Woon Lim, Arun Kumar Dutta "Reviewing the world's edible mushroom species: A new evidence-based classification system"

[7] Partho P. Sengupta and Sirish Shrestha "Machine Learning for Data-Driven Discovery: The Rise and Relevance"

[8] Jaime G. Carbonell, Ryszard S. Michalski, Tom M. Mitchell "1 - AN OVERVIEW OF MACHINE LEARNING"

[9] Qiangfu Zhao, "Evolutionary design of neural network tree-integration of decision tree, neural network and GA," Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Seoul, Korea (South), 2001, pp. 240-244 vol. 1, doi: 10.1109/CEC.2001.934395.

[10] Steven J. Rigatti, MD, DBIM,DABFM J Insur Med (2017) 47 (1):31-39.

[11] Aziz, R.M., Baluch, M.F., Patel, S. et al. LGBM: a machine learning approach for Ethereum fraud detection. Int. j. inf. tecnol. 14, 3321–3331 (2022). https://doi.org/10.1007/s41870-022-00864-6

[12] Florian Huber, Artem Yushchenko, Benedikt Stratmann, Volker Steinhage "Extreme Gradient Boosting for yield estimation compared with Deep Learning approaches"

[13] Tianqi Chen, Carlos Guestrin "XGBoost: A Scalable Tree Boosting System"

[14] Z. Zhang and C. Jung, "GBDT-MO: Gradient-Boosted Decision Trees for Multiple Outputs," in IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 7, pp. 3156-3167, July 2021, doi: 10.1109/TNNLS.2020.3009776.

[15] Dennis Wagner, Dominik Heider & Georges Hattab Mushroom data creation, curation, and simulation to support classification tasks

[16] Wagner, D., Heider, D. & Hattab, G. Mushroom data creation, curation, and simulation to support classification tasks. Sci Rep 11, 8134 (2021). https://doi.org/10.1038/s41598-021-87602-3

[17] Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. (2003). KNN Model-Based Approach in Classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39964-3_62

[18] Chicco, D., Jurman, G. The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification. BioData Mining 16, 4 (2023). https://doi.org/10.1186/s13040-023-00322-4

[19] Chicco, D., Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21, 6 (2020). https://doi.org/10.1186/s12864-019-6413-7

[20] Kenneth Ge, Phuc Nguyen, Ramy Arnaout, An Improved Python Package for Loading Datasets from the UCI Machine Learning Repository

[21] Caelen, O. A Bayesian interpretation of the confusion matrix. Ann Math Artif Intell 81, 429–450 (2017). https://doi.org/10.1007/s10472-017-9564-8