Mushroom Toxicity Classification

Jingwen Wu 1005284728
Ningxin (Cynthia) Li 1005934599
Hanwen Ling 1005809733
Xiang Dong 1004123334

***Abstract*—This project aims to develop a machine learning-based solution for distinguishing between edible and poisonous mushrooms. Leveraging a dataset from the UCI Machine Learning Repository, we explore various classification algorithms, including Logistic Regression, SVM, Decision Tree and Neural Networks, to build a reliable model that accurately categorizes mushrooms based on features such as shape, color, and habitat. Key components of the project workflow include data preprocessing, model training, hyperparameter tuning and performance comparison of the different models. After hyperparameter tuning, the logistic regression model shows high accuracy, while the rest models display perfect performance metrics on the test set. This project not only seeks to improve public safety by providing a practical tool for mushroom identification but also demonstrates the potential of machine learning in areas where expert knowledge is limited.**

***Index Terms*—Binary Classification, Decision Tree, Logistic Regression, Neural Networks, SVM**

I. Introduction

Mushrooms have always been considered one of our daily dietary supplements with nutritional value. However, the presence of poisonous varieties raises significant concerns regarding public health and safety. According to a newsletter from the Centers for Disease Control and Prevention (CDC), accidental mushroom poisoning causes nearly 1,400 emergency department visits annually [1]. Hence, the need for a reliable and accessible solution for mushroom toxicity is paramount. Utilizing the capability of machine learning to learn complex patterns from data, we intend to develop an innovative solution to classify mushroom toxicity, enhancing public safety and accessibility.

In the past, before the widespread use of machine learning, traditional methods of identifying mushrooms heavily relied on expert knowledge and personal experience, which were not always accessible or reliable for people in real life. Moreover, the variability in mushroom characteristics posed a significant challenge to human judgment.

Existing solutions to address the issue of mushroom toxicity primarily include cultural practices, scientific research, and clinical observation focusing on mushroom identification. One common approach to classifying mushroom toxicity is through conventional knowledge and cultural practices. In many indigenous communities around the world, villagers have a good command of knowledge regarding the identification of mushrooms. Through ethnobotanical research methods, experienced villagers can recognize poisonous mushrooms by specific characteristics like color, shape and smell [2]. Researchers and mycologists have conducted experiments and studies in the laboratory using collected mushroom specimens and proposed clinical classification systems based on observed patterns of symptoms to identify toxic syndromes associated with different mushroom species [3].

As a result, there remains a need for more comprehensive and accessible solutions by machine learning to reliably identify toxic mushrooms and prevent poisoning incidents.

In this project, we apply some traditional and effective machine learning algorithms and models to try to solve this problem. Machine learning can solve this problem due to its ability to learn complex patterns from data. We utilize a representative dataset, Mushroom dataset [4], which includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms, to train the ML models. Our goal is to develop a model capable of automatically classifying mushrooms with high accuracy.

We first preprocess the data to filter out the data with missing values and encode the categorical data into numerical representations. After that, we split the data into training dataset and test dataset for further model training process. For the models, we use the logistic regression as the primary solution and use three other models, which are SVM, decision tree and neural network to enhance the performance. We also apply hyperparameter tuning to try to improve the performance of each model and it is also proved to be effective.

II. System Design

A. Dataset

As we want to utilize the machine learning method to come up with a model that could correctly distinguish between the benign mushrooms and the poisonous mushrooms, first we need to find a suitable database to train the model. We choose the Mushroom dataset to train the models. This dataset includes the characteristics of 23 species of gilled mushrooms in the Agaricus and Lepiota Family. They are correctly classified to poisonous and benign, align with 22 important characteristics, such as the cap shape, odor, population and etc. They are important characteristics to identify the mushrooms and we hope the machine learning models can learn the patterns from the characteristics to justify whether an unknown mushroom is poisonous or not.

To have a better training performance, we also do a data cleaning and preprocessing phase before the training. We first check whether there exists any null value in the dataset. The result shows that all the entries have an value and we do not

need to filter any data. As shown in Figure 1, the values of the columns are representing in characters, which are not feasible in training. So we apply LabelEncoder to convert categorical text data into a numerical format where each unique label is assigned a unique integer as shown in Figure 2. After that, we extract the features as the training data and the labels representing whether the mushroom is benign as the target variable. We also split the data into 80% training set and 20% testing set.

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...
5	e	x	y	y	t	a	f	c	b	n	...

6 rows × 23 columns

Fig. 1. Raw dataset samples.

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	1	5	2	4	1	6	1	0	1	4	...
1	0	5	2	9	1	0	1	0	0	4	...
2	0	0	2	8	1	3	1	0	0	5	...
3	1	5	3	8	1	6	1	0	1	5	...
4	0	5	2	3	0	5	1	1	0	4	...

5 rows × 23 columns

Fig. 2. Dataset samples after preprocessing.

B. Logistic Regression

In our project, the primary goal is to distinguish between edible and poisonous mushrooms, which is a classic binary classification problem. To develop a model that is both accurate and interpretable, we first select logistic regression as our primary solution due to its high efficiency and widespread application in solving binary classification problems. The performance of logistic regression model significantly depends on the choice of the hyperparameters, including regularization strength (C), the type of penalty and the solver which is the algorithm we use in the optimization problem.

Through the application of grid search method, we explore a series of hyperparameter combinations. This exploration involves three different solvers (liblinear, lbfgs, saga), each compatible with specific penalties (l1 and/or l2), and 20 distinct values of C chosen from a geometric sequence between 10^{-4} and 10^{5}. This fine-tuning process not only prevents model overfitting, but also ensure the model achieves optimal performance and maintains its ability to generalize to unseen data. The results of hyperparameter tuning process demonstrates that the optimal combination is a regularization strengh of 3792.69, with a l2 penalty and the lbfgs solver. Notably, the l2 penalty and the lbfgs solver are widely employed on large datasets recognized for their stability and efficiency,

While logistic regression provides a solid foundation for our analysis, its limitations in handling nonlinear relationships in the dataset prompted us to further explore other alternative models to classify mushrooms.

C. SVM

To improve the performance, we use SVM to improve the performance of the primary solution. Compared to logistic regression, it can maximize the margin between the two classes. It can provide better generalization on unseen data, as it not only distinguishes the classes but also tries to do so with the widest margin. What is more, it can also handle non-linear boundaries and capture non-linear features. Kernels such as Radial Basis Function (RBF) enable SVM to find a complex non-linear boundary in the original feature space without transforming the data.

To find the best parameters for this problem, we use the grid search and cross validation methods to do a hyperparameter tuning. We tune SVM on three important hyperparameters, which are *C, kernel* and *gamma*. *C* is a regularization parameter. For larger value of *C*, it accepts a smaller margin if the classification result is correct. For smaller value of *C*, it encourages the margin to be larger. *kernel* decides which kernel function is used to transform the input data into a higher-dimensional space in the SVM model. It can perform linear and non-linear classifications. *gamma* is a kernel coefficient of non-linear kernels. It defines how far the influence of a single training example reaches, which controls the trade-off between bias and variance in the model.

To find the best hyperparmeters, we apply grid search and cross-validation (cv=5) to the model. We choose the set of hyperparameters which perform best on the validation data. Regarding to the values of the tuning parameters, we set four *C* ([1, 10, 100, 1000]), one linear kernel (linear) and one non-linear model (rbf) and three kernel coefficient of non-linear kernels (0.1, 0.2, 0.4). The best set of hyperparameters is: *C=1, kernel=rbf, gamma=0.1*. It is expected as the non-linear model can capture the non-linear relations while the linear model can't.

D. Decision Tree

To improve upon the primary solution of the logistic regression model, we also chose the decision tree model as an alternative, primarily due to its ability to capture complex non-linear relationships within the data. Decision trees can effectively model nonlinear relationships by recursively partitioning feature space into regions that separate binary classes. This flexibility allows decision trees to capture intricate patterns and interactions among mushroom characteristics, potentially leading to better classification performance. Additionally, decision trees offer clear and intuitive decision rules that can be easily understood and interpreted.

In the hyperparameter tuning process, the selection of hyperparameters such as *criterion, max_depth, min_samples_split, min_samples_leaf, and max_features* is crucial for optimizing the performance of the decision tree model. The cri-

terion parameter determines the function to measure the quality of a split which can take values of 'gini' or 'entropy'. *Max_depth* controls the maximum depth of the tree, while *min_samples_split* represents the minimum number of samples required to split an internal node. Additionally, *min_samples_leaf* is the minimum number of samples required to be at a leaf node, and *max_features* determines the number of features to consider when looking for the best split.

By systematically exploring different combinations of hyperparameters using the technique of grid search and cross-validation (cv=5), we could identify the configuration that yields the best performance on validation data. This fine-tuning process involves two different criteria (gini and entropy), four maximum depths (5, 10, 20, 30), four minimum sample splits (2, 3, 4, 5), three minimum sample leaves (1, 2, 4), and two options for max features (sqrt and log2). As a result, the best decision tree model has the following hyperparameters: *criterion='gini',max_depth=10, min_samples_split=2, min_samples_leaf=1*, and *max_features='sqrt'*. These hyperparameters were selected based on their ability to optimize the performance of the decision tree model and its generalization to unseen data.

As depicted in Figure 3, the decision tree model is very well visualized. The decision tree partitions the feature space based on the values of the features, with each node representing a decision based on a specific feature and threshold. Starting at the top, which represents the root node, "odor ≤ 2.5" stands for the initial split based on the characteristic of odor, using a threshold value of 2.5. It reveals that among 6499 samples, the majority (3388) are labelled as "Edible" based on this constraint. The Gini impurity at this node, measured at 0.499, reflects the degree of uncertainty in classification, suggesting a relatively mixed distribution of classes among the samples. Similarly, the internal nodes represent subsequent splits based on different features. Each node indicates the feature and threshold value used for splitting the data further. At the bottom, leaf nodes represent the final outcome of the decision process. They don't split further and correspond to a specific class label, either "Edible" or "Poisonous". Notably, there are two colors, blue and orange, in the plot. Nodes colored blue represent the "Poisonous" class, while orange-colored nodes denote the "Edible" class. The decision tree model's strength lies in its transparent nature, offering a clear visualization for us to interpret easily.

E. Neural Network

In addition to the SVM and decision tree models, we also selected a neural network as an alternative solution to logistic regression. The neural network offers significant advantages in detecting complex patterns within high-dimensional data, especially given that our dataset contains 23 features. Unlike logistic regression, neural networks excel in learning and distinguishing non-linear data, and are expected to achieve higher accuracy.

The architecture of our neural network consists of three linear layers, where the first two layers are combined with the

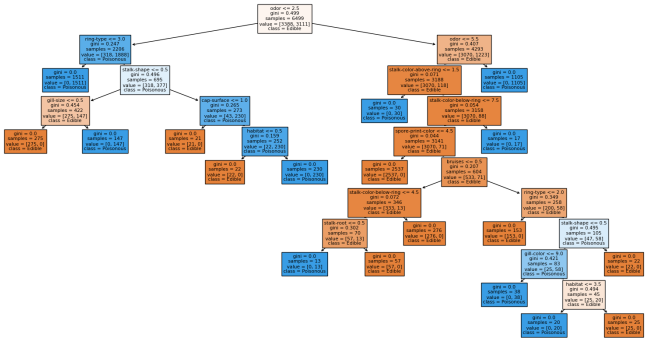

Fig. 3. Decision Tree Model Visualization.

ReLu activation function to add nonlinearity to our model. For the final layer, we use the sigmoid function as the activation, so the output will be the probability of being classified as 1, which represents poisonous. If the probability exceeds 0.5, we predict the mushroom to be poisonous. Our model calculates losses using binary cross-entropy and updates parameters utilizing the Adam optimization algorithm, which is an extension of stochastic gradient descent.

We split data into a 60% training set, a 20% validation set, and a 20% test set. Regarding hyperparameters, we focused on tuning the batch size and learning rate. We experimented with batch sizes of 128 and 256, and learning rates of 0.1 and 0.01, resulting in four different combinations. Generally, a smaller batch size leads to more weight updates per epoch, increasing training time and sensitivity to noise or outliers, but also allowing for more frequent updates and potentially a more refined learning process. Conversely, a larger batch size can make training each epoch computationally expensive and slow, while resulting in smoother optimization and robustness against noise. A large learning rate might cause the model's updates to oscillate and overshoot minima, leading to unstable training, while a smaller learning rate ensures cautious weight updates and a stable learning curve but may slow convergence. Therefore, we tuned the hyperparameters based on validation accuracy to keep a balanced batch size and learning rate.

Based on the results of hyperparameters tuning, the best test accuracy occurred at the learning rate = 0.01 regardless of the batch size. Both models converged at around epoch 10. All of the training, validation, and test accuracies could achieve 1 and no overfitting occurred. This indicates that our model are able to generalize and perform well on new, unseen data. The training and validation accuracy curve is shown in Figure 4. Therefore, we tend to use the larger batch size to enhance the stability of the model.

III. Results

In our project, we employ a series of performance metrics to comprehensively evaluate the performance of logistic regression, support vector machine (SVM), decision tree and neural network models in classifying mushrooms as either edible or poisonous. Firstly, we focus on the accuracy of these models to assess their overall performance on the test dataset, which

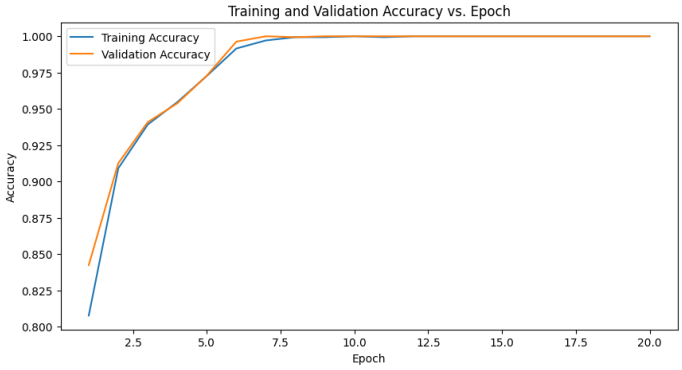

Fig. 4. Training and Validation Accuracy of Neural Network Model Over Epochs.

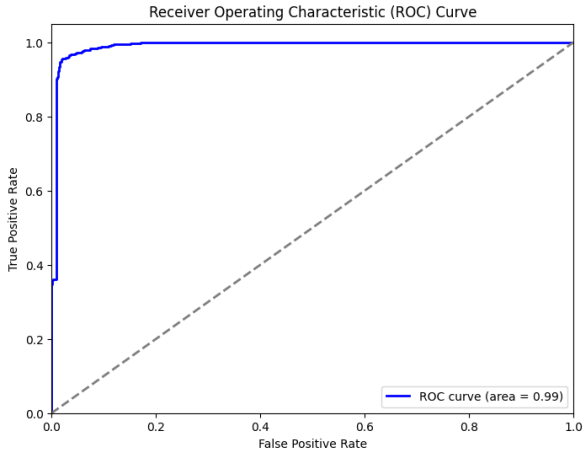

Fig. 5. AUC-ROC of Logistic Regression Model after hyperparameter tuning.

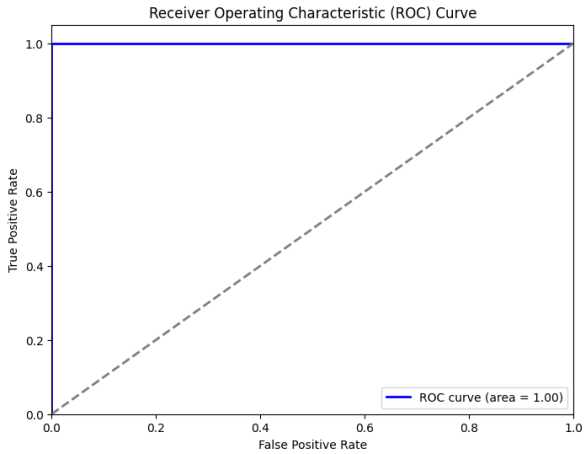

Fig. 6. AUC-ROC of SVM, Decision Tree and Neural Network Models after hyperparameter tuning.

indicates the proportion of total correct predictions made out of all predictions. In addition, given the potential health risk associated with misclassifying mushrooms, precision and recall are particularly emphasized. Precision represents the fraction of samples predicted as poisonous by the model that are actually poisonous, whereas recall indicates the fraction of actual poisonous samples that are correctly identified by the model. These two metrics are crucial for ensuring the reliability of the model in distinguishing poisonous mushrooms from the edible ones. Furthermore, since there might be a potential class imbalance between poisonous and edible mushrooms, we also include F1 score as another metric to evaluate the models. F1 score is a balanced measure of accuracy giving equal importance to precision and recall, therefore it provides a more comprehensive evaluation in the context of class imbalance. The final metric we use is the area under the receiver operating characteristic curve (AUC-ROC), which allows us to evaluate the performance of the models across different thresholds. This metric is vital for understanding how well the models can distinguish between the classes under different circumstances.

A. Comparison of Base Models

In the initial phase of our analysis, we evaluate the fundamental performance of our four machine learning models without any hyperparameter tuning. The base models and their performance metrics are summarized in Table I.

TABLE I
PERFORMANCE METRICS OF BASE MODELS

	Logistic Reg.	SVM	Decision Tree	Neural Net.
Accuracy	94.77%	98.52%	98.03%	96.43%
Precision	95.92%	99.87%	97.31%	95.25%
Recall	93.42%	97.14%	98.76%	97.44%
F1 Score	0.9465	0.9849	0.9803	0.9633
AUC-ROC	0.98	0.98	0.99	0.99

The above table presents a comparative view of accuracy, precision, recall and F1 score for logistic regression, support vector machine, decision tree, and neural network models. The logistic regression model shows a promising start with an accuracy of 94.77%, a precision of 95.92%, a recall of 93.42%, an F1 score of 0.9465 and an AUC-ROC value of 0.98. The SVM model performs the best among all four models with the highest accuracy of 98.52% and precision of 99.87%, the second highest recall of 97.14%, resulting the highest f1 score of 0.9849 and an AUC-ROC value of 0.98. The decision tree model follows closely and achieves an accuracy of 98.03%, a precision of 97.31%, a remarkable recall of 98.76%, an F1 score of 0.9803 and an AUC-ROC value of 0.99. The neural network model, while having a slightly lower accuracy of 96.43% and precision of 95.25%, with an acceptable recall of 97.44%, resulted in an F1 score of 0.9633.

These initial results provide a baseline for our subsequent hyperparameter tuning, and gives us great insights about the strength and weakness of our base models, guiding us to adjust our hyperparameter tuning strategy to further improve the performance of our models.

B. Comparison of Models After Hyperparameter Tuning

After the process of hyperparameter tuning, the performance of all of our four machine learning models significantly

improved. The models and their performance metrics after hyperparameter tuning are summarized in Table II.

TABLE II
PERFORMANCE METRICS OF BASE MODELS

	Logistic Reg.	SVM	Decision Tree	Neural Net.
Accuracy	96.49%	100%	100%	100%
Precision	97.22%	100%	100%	100%
Recall	95.65%	100%	100%	100%
F1 Score	0.9643	1.0000	1.0000	1.0000
AUC-ROC	0.99	1.00	1.00	1.00

The above table presents a comparative view of accuracy, precision, recall and F1 score for the four models after hyperparameter tuning. The performance of logistic regression has shown significant improvement, with its accuracy increasing to 96.49%, paired with an impressive precision of 97.22%, a recall of 95.65% and an F1 score of 0.9643. Notably, while not achieving perfect score, the AUC-ROC value of logistic regression also increases to 0.99, which underscores its outstanding classification ability.

In contrast, the SVM, decision tree, and neural network models have all achieved perfect performance after hyperparameter tuning, reflected by 100% across all metrics, including a flawless AUC-ROC value of 1.00. This level of performance indicates a perfect capability in classifying mushrooms.

IV. CONCLUDING REMARKS

After tuning the hyperparameters, all of the SVM, decision tree, and neural network models achieved an impressive 100% accuracy. Compared to our primary solution of logistic regression, these three alternative solutions perform better when identifying poisonous mushrooms, and each model demonstrates its unique advantages and disadvantages: the SVM model excels in handling higher dimensional features, decision tree has superior interpretability, and neural network can easily process large volumes of data. Based on the performance metrics and analysis of the data,it is challenging to declare which model is the best. For practical applications, a combined approach employing all three models in a majority voting system may be used to minimize the risk of misclassification caused by a single model and to enhance the reliability of mushroom classificaion, thus preventing serious food poisoning incidents.

In conclusion, our study demonstrates the effectiveness of machine learning algorithms in the toxic categorization of mushrooms. This project enables quicker and more practical mushroom classification, illustrating the potential of machine learning in the area of public health. We expect that our models to be integrated into food inspection processes or residential mobile applications, thus simplifying mushroom categorization and spreading safety information to the public, particularly benefiting for those without expert knowledge about mushrooms.

REFERENCES

[1] Centers for Disease Control and Prevention. (2021) Cdc and food safety newsletter: poisonous_mushrooms. [Online]. Available: https://www.cdc.gov/foodsafety/newsletter/poisonous_mushrooms.html

[2] A. Montoya, O. Hernández-Totomoch, A. Estrada-Torres, A. Kong, and J. Caballero, "Traditional knowledge about mushrooms in a nahua community in the state of tlaxcala, méxico," *Mycologia*, vol. 95, no. 5, pp. 793–806, 2003.

[3] J. White, S. A. Weinstein, L. De Haro, R. Bédry, A. Schaper, B. H. Rumack, and T. Zilker, "Mushroom poisoning: A proposed new clinical classification," *Toxicon*, vol. 157, pp. 53–65, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0041010118307281

[4] "Mushroom," UCI Machine Learning Repository, 1987, DOI: https://doi.org/10.24432/C5959T.