Mushroom Edibility Prediction

Zhikai Zhao

DSI

Github Repository

1 Introduction

Mushroom edibility prediction is to determine whether a mushroom is edible or poisonous based on various characteristics. This process is crucial for public health, ecological understanding, and the advancement of mycology and technology.

As for public health and safety, accurate identification of edible and poisonous mushrooms is crucial for foragers, chefs, and anyone consuming wild mushrooms. This research directly contributes to reducing incidents of mushroom-related toxicity, which can range from mild gastrointestinal discomfort to fatal outcomes. Regarding ecological and biological research, mushrooms play critical roles in ecosystems as decomposers, and research in this area contributes to our knowledge of biodiversity, ecological interactions, and environmental health. Also, this research helps deepen our understanding of mycology, the study of fungi.

I obtained the dataset from Kaggle [1] (which illustrates that the data is collected from Patrick Hardin, Mushrooms & Toadstools and Jeff Schlimmer, Mushroom Data Set [2]). This dataset includes 61069 hypothetical mushrooms with caps based on 173 species (353 mushrooms per species). Each mushroom is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended (the latter class was combined with the poisonous class). Of the 20 variables, 17 are nominal and 3 are metrical. The target variable is “class”, which indicates mushroom edibility with “p” (poisonous) and “e” (edible) (with the former one also containing mushrooms of unknown edibility).

In Kaggle, someone [3] has used Random Forest to predict the problem and achieved a high accuracy score of 100%.

2 Exploratory Data Analysis

Firstly, I figured out some basic information of the dataset. There are 3 continuous features “cap-diameter”, “stem-height”, “stem-width” and the other features are all categorical. And for the target variable, there are 33888 poisonous pieces and 27181 edible pieces, which means that the dataset is somewhat balanced.

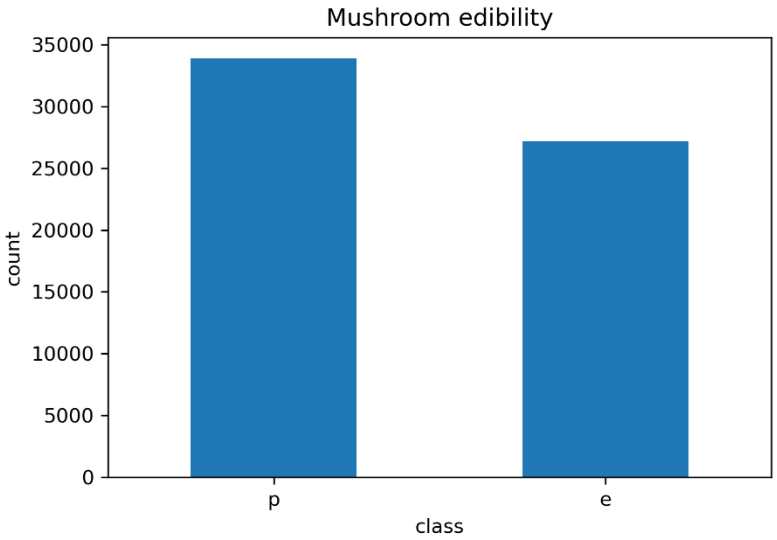

Figure 1 Target Variable Distribution

Then, I plot some figures about the continuous features. As for cap-diameter, I find that it is mainly distributed between 0cm and 20cm, and the mushroom tends to be poisonous when the cap diameter is less than 6cm.

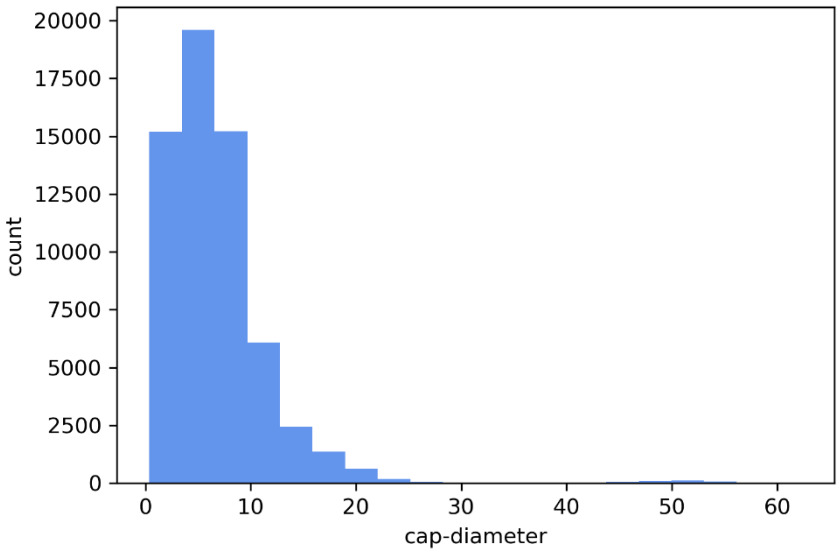

Figure 2 Distribution of cap-diameter

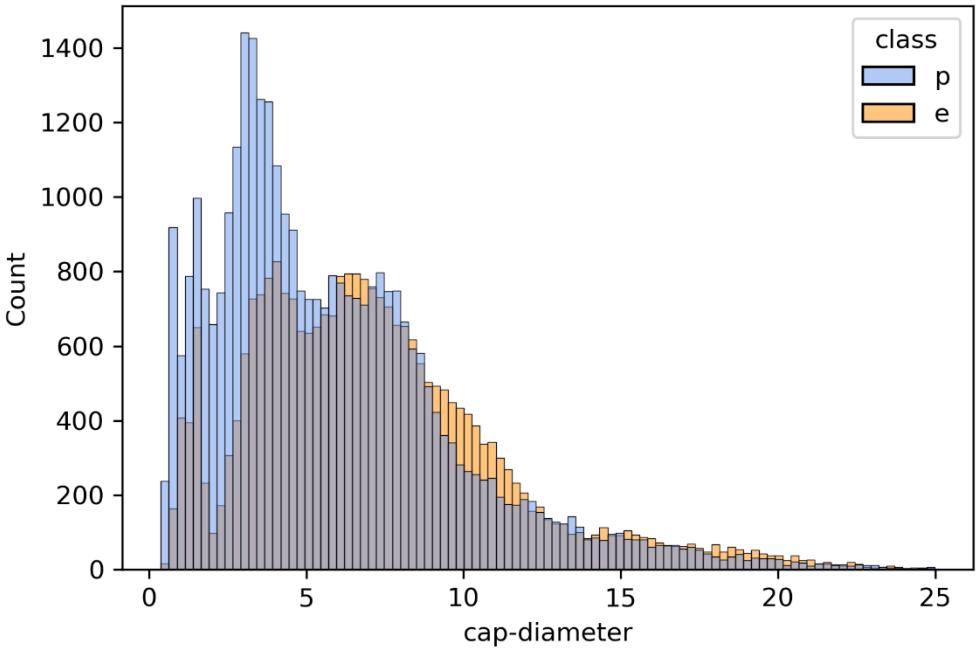

Figure 3 cap-diameter vs. class

As for stem-height and stem-width, the distribution is as follows. I can tell from the two figures that when stem-height is between 6cm and 8cm or stem-width is between 18mm and 26mm, the mushroom tends to be edible.

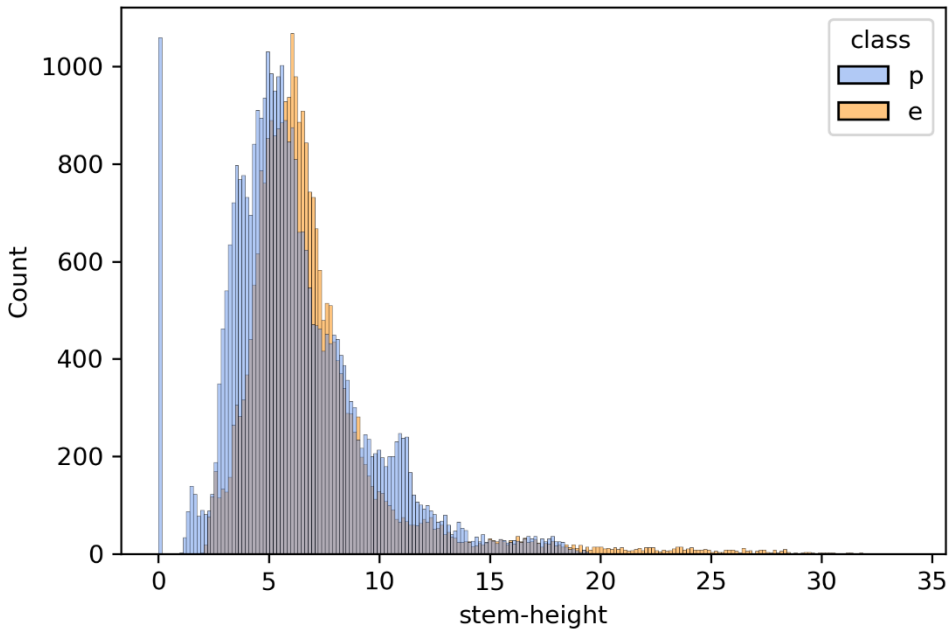

Figure 4 Distribution of stem-height

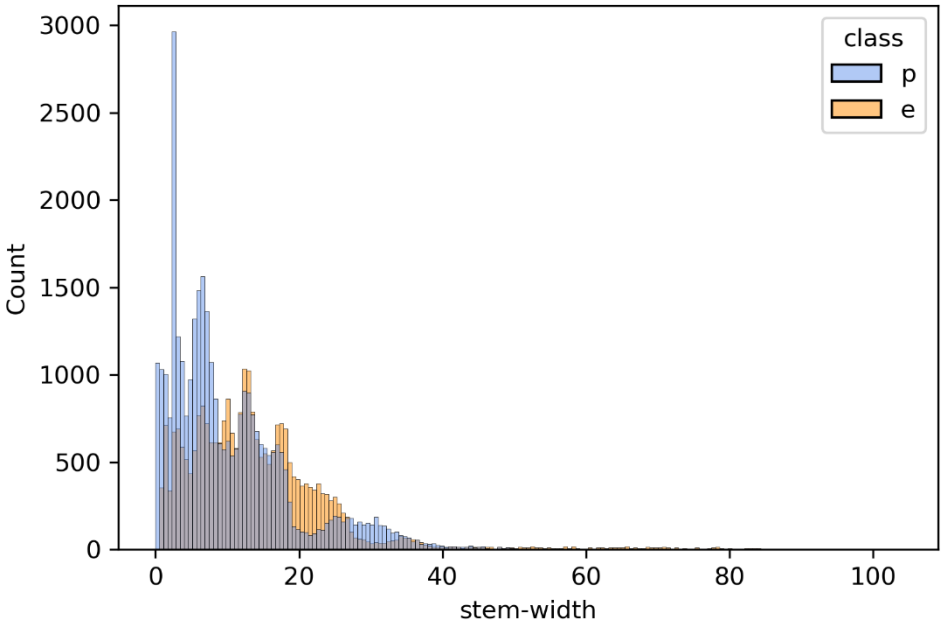

Figure 5 Distribution of stem-width

And I can see from the violin plots that the three features are somewhat similar, edible class has a wider range in these features compared to poisonous class.

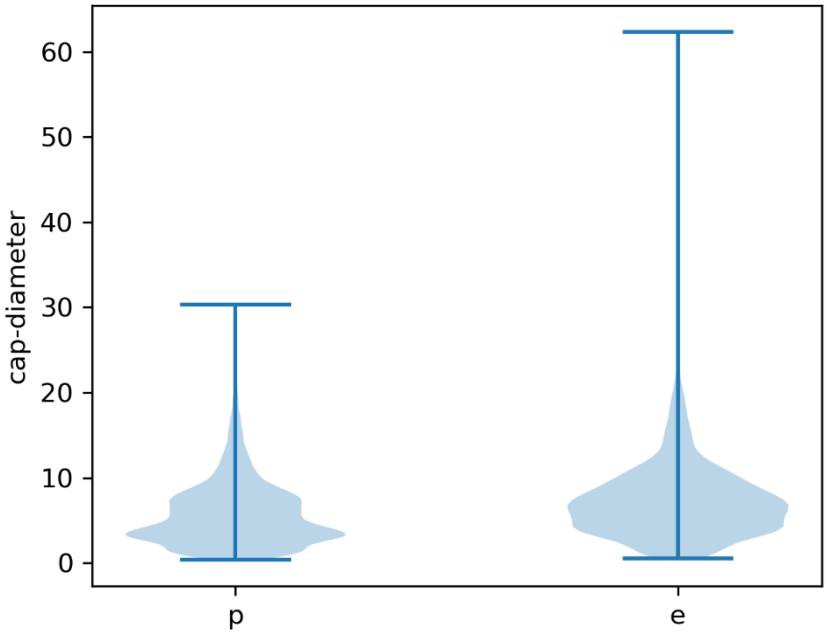

Figure 6 cap-diameter vs. class

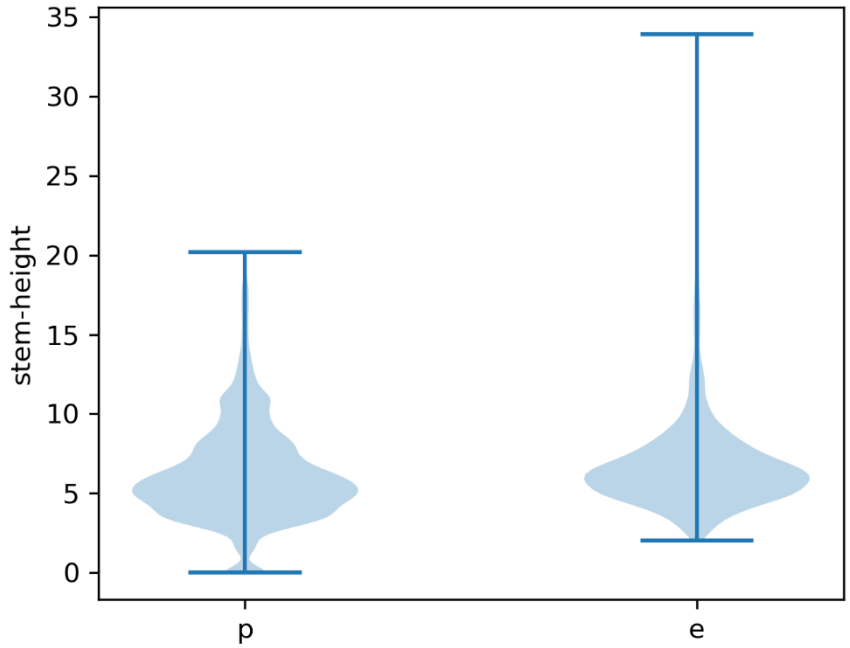

Figure 7 stem-height vs. class

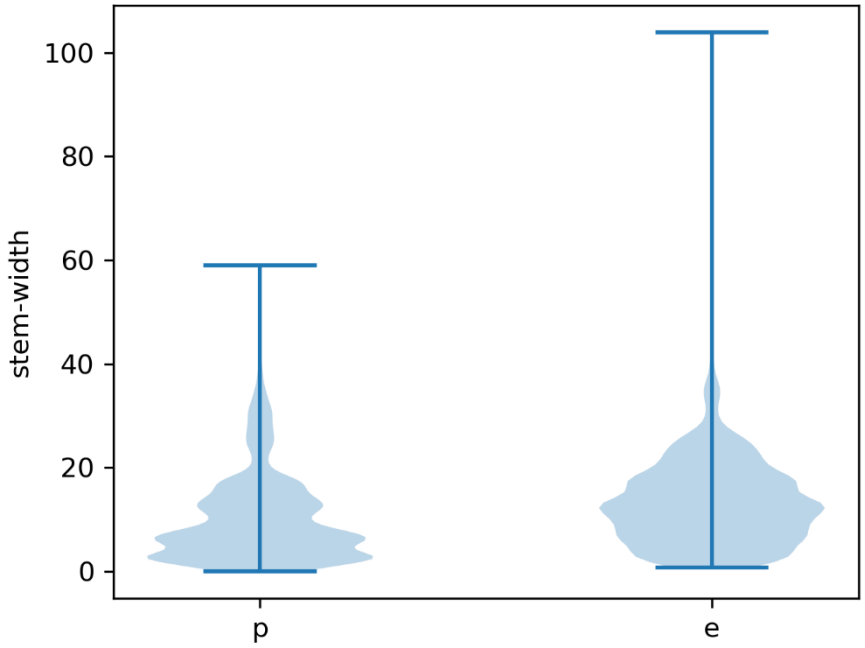

Figure 8 stem-width vs. class

Apart from numerical columns, I plot some figures about categorical features and it appears that the mushroom tends to be edible when it grows in spring and winter or the cap shape is spherical.

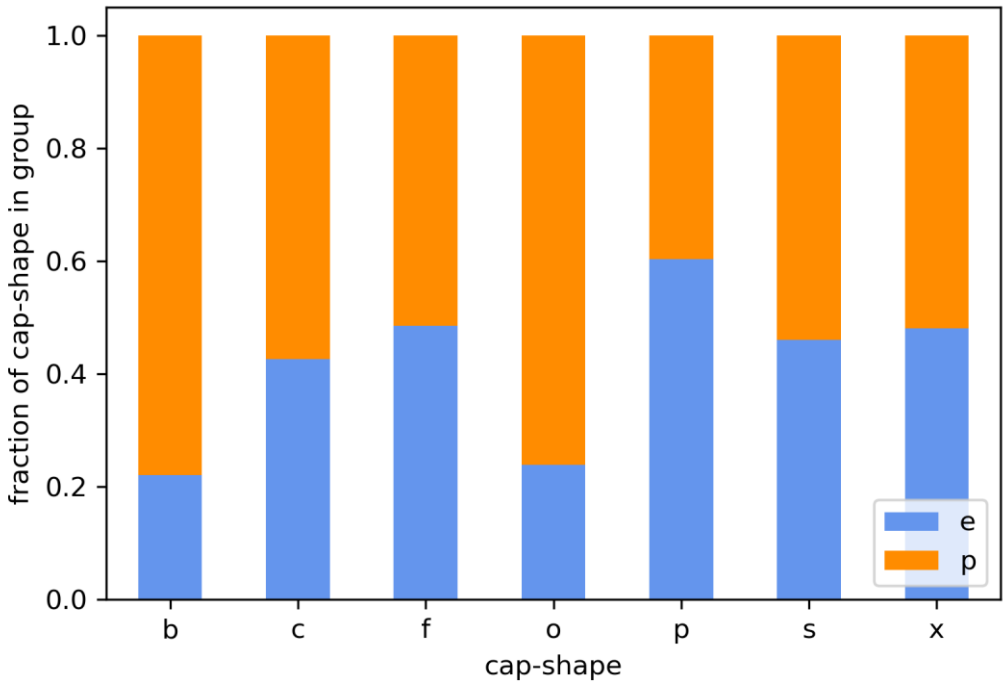

Figure 9 cap-shape vs. class

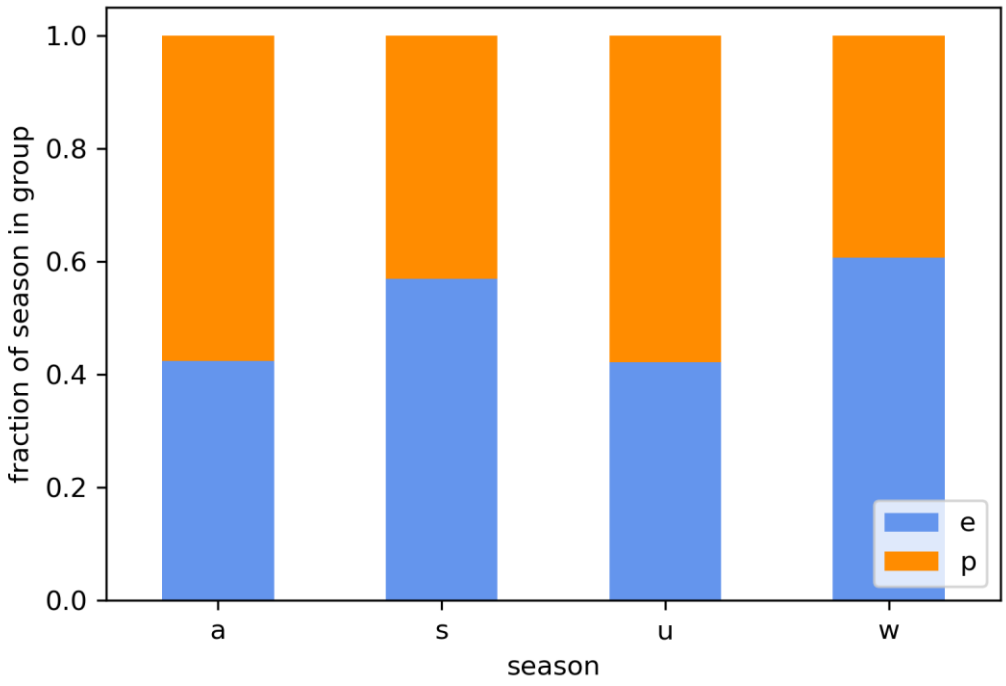

Figure 10 season vs. class

3 Methods

1) Data splitting

Firstly, I use basic train test split with the proportion of 80%/20% to create test set and other set because the dataset is somewhat balanced and there is no need to do stratified. Then I use stratified K-Fold with 4 folds to split train-validation set for other set to avoid imbalance and overfitting. Moreover, stratified K-Fold allows for efficient use of data. Each data point gets to be in a test set exactly once, and is used for training k-1 times. In the process of tuning hyperparameters, having a consistent distribution of classes across each fold ensures that the tuning process is not biased towards a specific subset of the data. In splitting, I set 10 different random states to calculate 10 different scores.

2) Data preprocessing

In the dataset, 100% of the data points have missing values, and 9 features have missing values and they are all categorical. So, it is impossible to just remove those data points or features. As for the categorical features, I use one-hot encoder because they are unordered and I treat these missing values as another category. As for the continuous features, I use standard scalar to ensure that mean is 0 and std is 1 for later model training.

3) Evaluation metric

I use accuracy as the metric because of this balanced dataset. It is an effective measure of a model's performance, which gives a quick assessment of how often the model is correct, regardless of the class. Also, accuracy is straightforward to understand and interpret. This simplicity makes it a good starting point for model evaluation, especially for those new to machine learning. Besides, the consequences of two types of errors will change under different circumstances, which means false positives (predict poisonous one as edible) are fatal to human health and false negatives (predict edible one as poisonous) are harmful to research.

4) ML pipeline

It contains data splitting and preprocessing, evaluation metric as above, and I use Grid-Search-CV to do parameter tuning for four different ML algorithms:

Logistic Regression (elastic net), Random Forest, KNeighbors, SVC. The parameters to tune are as follows.

Table 1 parameters values

Algorithm	Non-linear?	Parameters to tune	
Logistic Regression (elastic net)	no	C: [0.01, 0.1, 1, 10, 100]	l1_ratio: [0.1, 0.3, 0.5, 0.7, 0.9]
Random Forest	yes	max_depth: [1, 3, 10, 30, 100]	max_features: [0.25, 0.5, 0.75, 1.0]
KNeighbors	yes	n_neighbors: [3, 5, 7, 9, 11]	weights: ['uniform', 'distance']
SVC	yes	C: [0.01, 0.1, 1, 10, 100]	gamma: [0.01, 0.1, 1, 10, 100]

5) Uncertainties

For data splitting, it can introduce variability, especially if the splits are not representative of the overall dataset. Hence, I use stratified sampling to ensure representative splits and set 10 different random states to ensure that every time the four models apply to the same training set.

For the non-determinism Random Forest, as it performs well on every data set with the accuracy of 100%, I didn't take it into account.

For model selection and hyperparameter tuning, different models and hyperparameters can perform differently on the same dataset, so I use grid search for hyperparameter tuning and compare different models using the same evaluation metrics and splitting strategy.

For evaluation metrics, the choice can affect how the model's performance is perceived, especially in the context of specific business or application requirements. Thus, I just choose accuracy that align with the balanced dataset.

4 Results

The baseline accuracy score is 0.5659, and the scores of four models are all much higher than it, including standard deviations.

Table 2 Scores of 4 models

Algorithm	Accuracy Scores	Mean	Std
Logistic Regression (elastic net)	[0.8696584, 0.88250262, 0.87382333, 0.86935498, 0.85999615, 0.87715303, 0.86107173, 0.89613488, 0.90168477, 0.85689169]	0.87482715	0.01424146
Random Forest	[1,1,1,1,1,1,1,1,1,1]	1	0
KNeighbors	[1,1,1,1,1,1,1,1,1,1]	1	0
SVC	[0.98968189, 0.99157658, 0.99029627, 0.98963713, 0.98825658, 0.99078744, 0.98841524, 0.99358751, 0.99440619, 0.98779863]	0.990444346	0.0021008

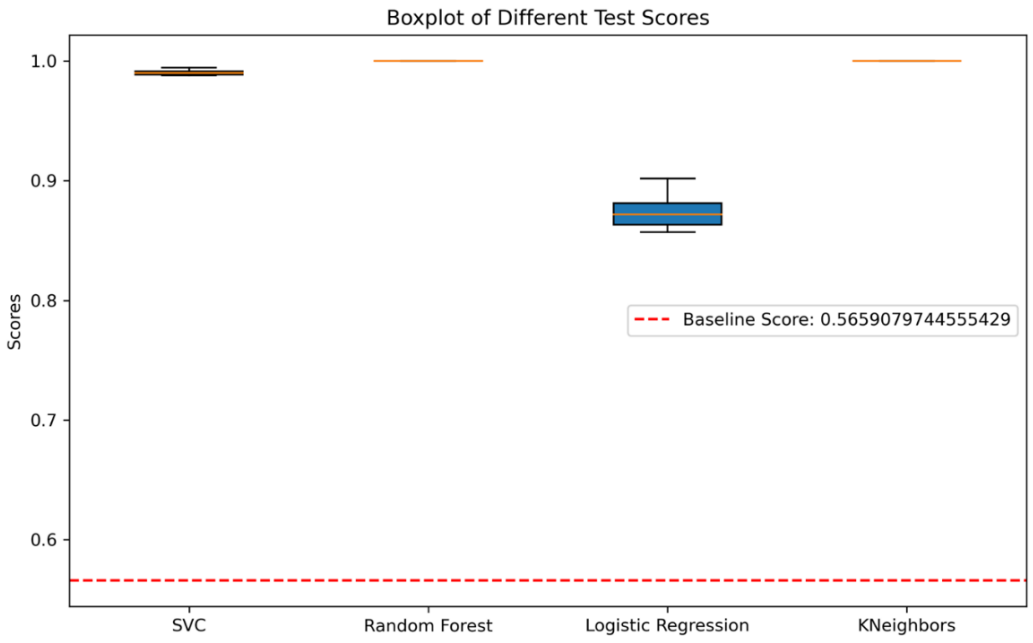

Figure 11 distribution of test scores

From the results, I can tell that both Random Forest and KNeighbors have the most predictive power, achieving a high accuracy of 100%. For interpretability, I choose KNeighbors as the model for later process because it is much simpler.

For the global feature importance, I utilize three methods: Perturbation, SHAP and XGBOOST (gain).

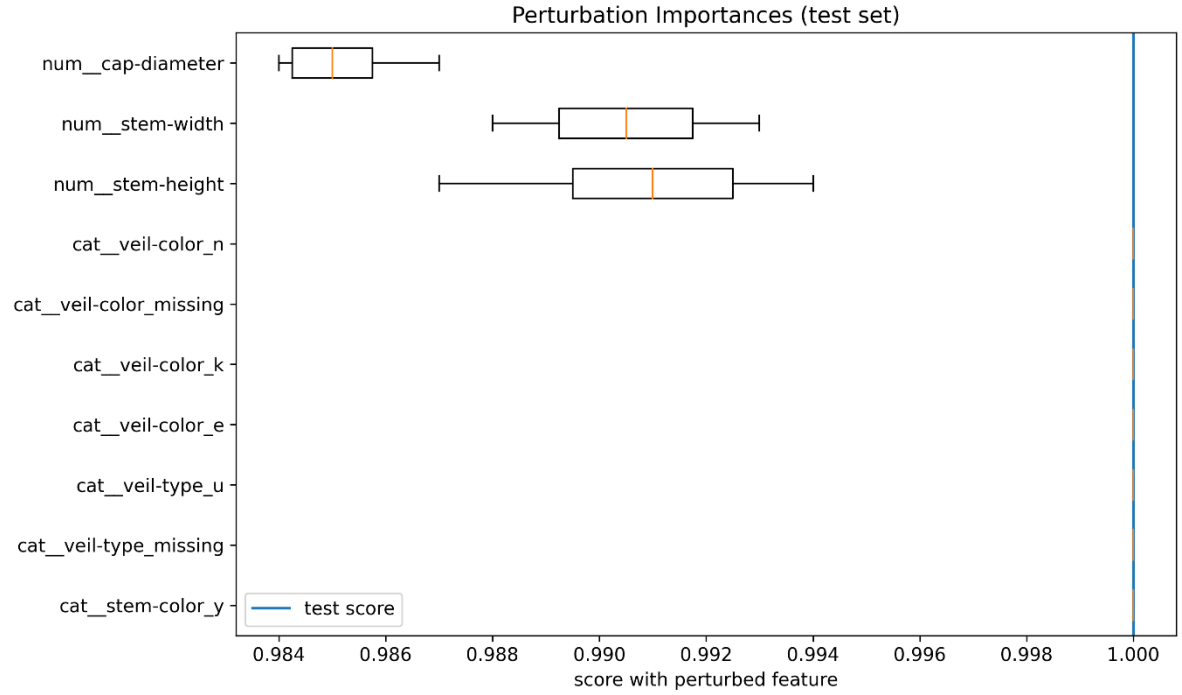

Figure 12 Top 10 features in perturbation

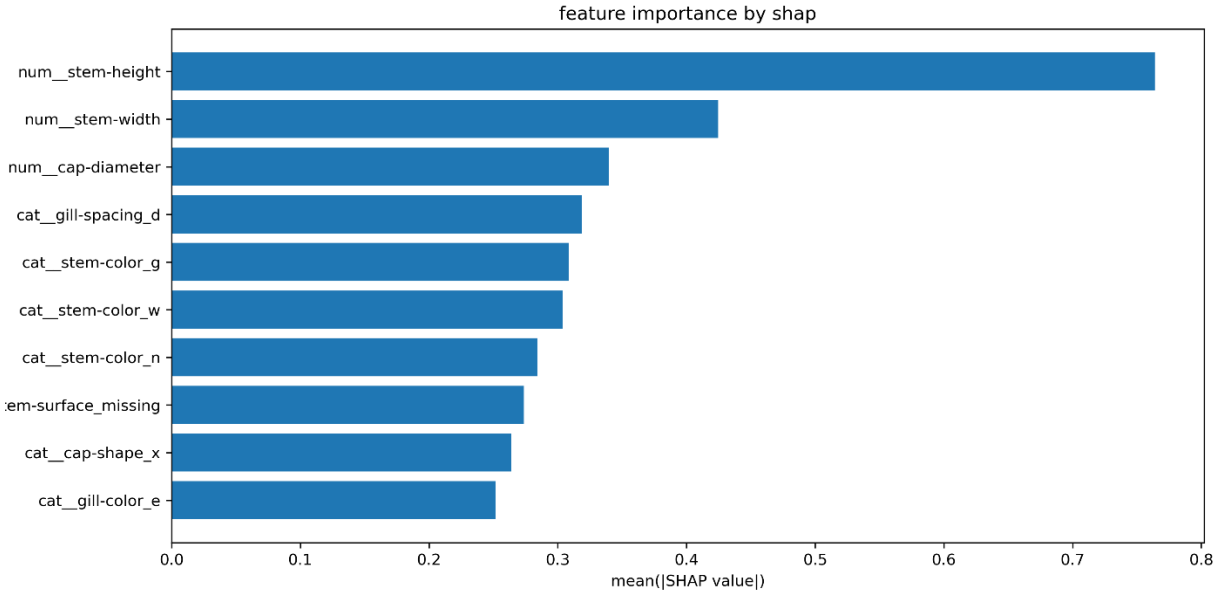

Figure 13 Top 10 features in shap

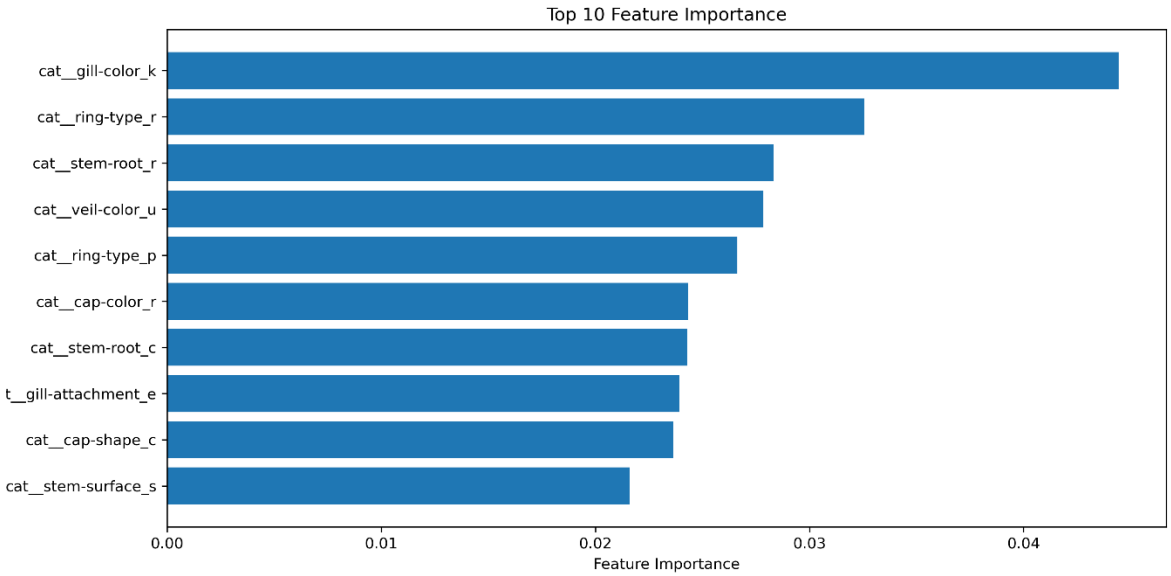

Figure 14 Top 10 features in xgboost

From these three figures, we can tell that all three numerical features rank top 3 in both perturbation and shap, but in xgboost some categorical features have higher importance than them.

Next, I choose two specific data points and plot their local feature importance by force plot.

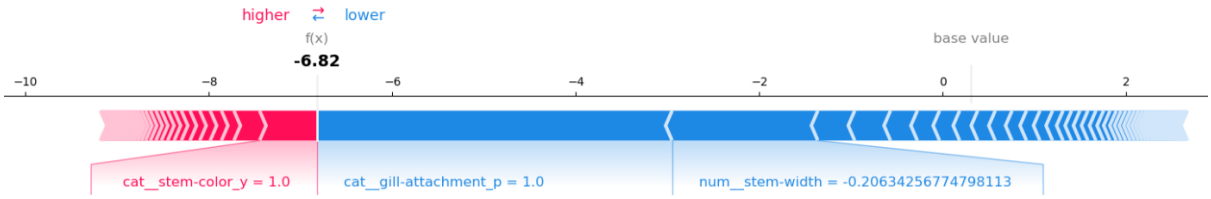

Figure 15 Local feature importance 1

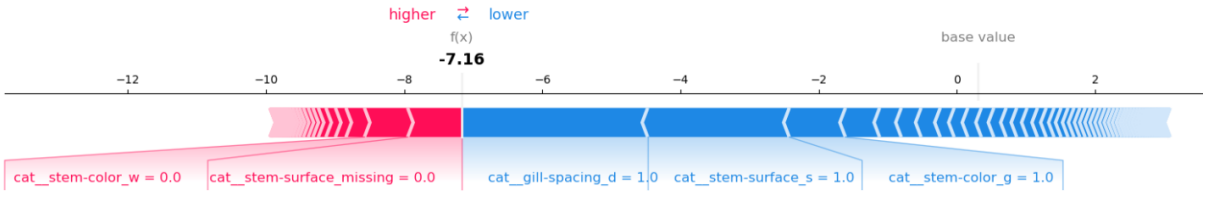

Figure 16 Local feature importance 2

From the local feature importance above, I can conclude that for data point 1, features like “cat_stem-color_y” contribute positively to the prediction while those like “cat_gill-attachment_p” and “num_stem-width” contribute negatively. For data point 2, features “cat_stem-color_w” and “cat_stem-surface_missing”

contribute positively to the prediction while those like “cat_gill-spacing_d” and “cat_stem-surface_s” contribute negatively.

As for the discrepancy in feature importance ranks, it may be because nominal features are ranked higher as they are used more frequently in splits, or they contribute more to the gain when they are used, even if they aren't the primary drivers of predictive accuracy as per perturbation or SHAP.

In the context of this practical problem, the implications of misclassifying an edible mushroom as poisonous are different from misclassifying a poisonous one as edible. Therefore, the importance of features should be interpreted not just by their mathematical contribution to model accuracy, but also by their impact on the model's reliability in differentiating between the two classes.

The differences in feature importance rankings highlight the need for diverse model interpretation methods to understand the different aspects of feature relevance and ensure robustness. In safety-critical applications like mushroom foraging, a conservative approach is warranted. Features that lead to classifying a mushroom as poisonous should be scrutinized to minimize the risk of false negatives (i.e., declaring a poisonous mushroom as edible).

Given that SHAP and Perturbation methods agree on the importance of numerical features, I would say that three continuous features “stem-height”, “stem-width”, “cap-diameter” are most important, and practitioners should carefully consider these features when making decisions based on the model's output. While those categorical features are less important and it is not that precise to determine these features in real life (i.e., decide whether it grows in spring or summer).

In summary, interpreting feature importance requires considering both the statistical significance and the real-world implications of the features. It is crucial to use multiple interpretation methods to get a comprehensive understanding of the features' roles in the model's predictions and to apply domain knowledge to mitigate risks associated with model-driven decisions.

5 Outlook

1) Weak points

Firstly, the way of preprocessing the categorical features generates too many features, totally 128 columns, which causes later model training to run so long. It occupies too much computational resource and is not that efficiency. Secondly, we may need to test more algorithms to find a best one, and it is not that reliable for only 4 testing algorithms. Thirdly, I haven't combined the real problem with this model and just choose a common way, which may cause that the evaluation metric is not consistent with the demand.

2) Improvement

As for feature engineering, I can use techniques like PCA (Principal Component Analysis) to reduce the number of features while retaining most of the variance in the data. This not only can improve performance but also helps in visualization and understanding. Also, it is more efficient to identify and retain the most significant features. Methods like mutual information, ANOVA can be used.

For simplifying the model, if interpretability is a priority and the scores are close, simpler models might offer more straightforward insights. Sometimes, using a simpler or more constrained model can increase interpretability and the efficiency even if it sacrifices some predictive power.

For more data to improve the model performance, I can collect genetic information. DNA sequencing or genetic markers could provide insights into the toxicity of mushrooms, as certain genetic sequences are often associated with the presence of toxins. Besides, interaction with other species is also useful. Data on surrounding flora and fauna, as some mushrooms may develop toxins or lose them based on interactions with other organisms.

6 Reference

[1] https://www.kaggle.com/datasets/devzohaib/mushroom-edibility-classification/data

[2] Wagner,Dennis, Heider,D., and Hattab,Georges. (2023). Secondary Mushroom Dataset. UCI Machine Learning Repository. https://doi.org/10.24432/C5FP5Q.

[3] https://www.kaggle.com/code/lucasfca/mushroom-edibility-classification

[4] https://christophm.github.io/interpretable-ml-book/

[5] https://xgboost.readthedocs.io/en/stable/