1 Introduction

Using data on mushrooms from our training set (*The National Archives on Mushrooms*), we would like to train machine learning models such that we can use the *Shroomster Pro Max™* to evaluate if new mushrooms we encounter are poisonous (=0) or edible (=1).

A breakdown of our dataset is as follows:

Labels: One binary class divided in poisonous=p (to be encoded as class 0) and edible=e (to be encoded as class 1)

Features: These are the features that the Shroomster Pro Max™ can determine for mushrooms in the wild

1. *cap-diameter:* float number in cm
2. *cap-shape:* bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o
3. *cap-surface:* fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e
4. *cap-color:* brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k
5. *does-bruise-bleed:* bruises-or-bleeding=t,no=f
6. *gill-attachment:* adnate=a, adnexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=?
7. *gill-spacing:* close=c, distant=d, none=f
8. *gill-color:* see cap-color + none=f
9. *stem-height:* float number in cm
10. *stem-width:* float number in mm
11. *stem-root:* bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r
12. *stem-surface:* see cap-surface + none=f
13. *stem-color:* see cap-color + none=f
14. *veil-type:* partial=p, universal=u
15. *veil-color:* see cap-color + none=f
16. *has-ring:* ring=t, none=f
17. *ring-type:* cobwebby=c, evanescent=e, flaring=r, grooved=g, large=l, pendant=p, sheathing=s, zone=z, scaly=y, movable=m, none=f, unknown=?

18. *spore-print-color:* see cap color
19. *habitat:* grasses=g, leaves=l, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d
20. *season:* spring=s, summer=u, autumn=a, winter=w

Data Files: Train Set (*mushroom_train.csv*) has 50213 rows, Test Set (*mushroom_test.csv*) has 10856 rows.

My general methodology involved data cleaning and preparing the data for machine learning by pipelining and augmenting, data visualization, modeling, and hyperparameter tuning and cross-validation. I had the best success with Decision Tree Classifier (74% accuracy) and AdaBoost Classifier (66% accuracy) after applying PCA by reducing to 3 dimensions and using the best performing hyperparameters. K-Nearest Neighbors algorithm performed somewhat poorly, achieving about 59% accuracy pre-hyperparameter tuning and about 56% accuracy post-hyperparameter tuning.

2 Methodology

2.1 Data Loading, Splitting, Exploration and Visualization

I visualized all of the numerical features using a histogram to see their distributions (if they were normally distributed or were skewed). Cap-diameter, stem-height, and stem-width were all skewed right, meaning there are many outliers with high values.

I visualized all of the categorical features, as well as the labels, using bar charts to see if some categories had higher frequencies than others. For class, there are many more poisonous mushrooms than edible mushrooms. This means the labels are imbalanced, which could affect the accuracy of our models. Thus, other metrics might be better to measure performance, such as recall, since some positives might end up classified as false negatives due to there being more negative labels in the dataset. For the other features, there are some categories that have many more observations than other categories. For example, does-bruise-or-bleed has a much higher proportion of category f than t. This might be a result of using the mode to impute nulls for categorical variables. While using the mode is typically not the best solution, in the case of our dataset it was better to use the mode rather than dropping rows, as the dataset is fairly small.

2.2 Data Pre-Processing

I first preprocessed my data by dropping the columns that had a lot of null values in the train set. Filling the null values might end up overly imbalancing the features, so I decided dropping those columns would be the best option. I ended up dropping stem-root, stem-surface, veil-type, veil-color, and spore-print-color from both the train and test sets.

Then, I imputed the remaining nulls in both the train set and test set. None of the numerical features had null values, but some of the categorical features had null values. Due to the small size of the dataset, I decided not to drop rows as that would give us very little data to train our models.

Instead, I used the mode of each feature with null values to fill the missing data in both the train set and test set.

The last step of preprocessing was to pipeline my data. I applied standard scaling and one-hot encoding to my data through the pipeline. Since none of the categorical variables were ordinal, it did not make sense to use label encoding, as that would assign some implicit ordering to the categories.

2.3 Data Augmentation

I engineered two more features for my data: stem-area (stem-height * stem-width) and cap-diameter-cat, which assigns a value based on 5 bins spanning the range of cap-diameter.

Stem-area might be useful as a feature because it incorporates both stem-height and stem-width. It's possible for a mushroom with low stem-height and high-stem-width to have a similar stem-area as a mushroom with high stem-height and low stem-width.

Cap-diameter-cat might be useful as a feature because binning helps mitigate the issue of the skewed distribution of cap-diameter by selecting bins such that each bin contains a similar number of observations.

2.4 Statistical Hypothesis Testing

I investigated the relationship between the numerical variables, i.e. cap-diameter, stem-height, stem-width, and stem-area (the feature I added to the data), and the mushroom class.

Cap-diameter is statistically significant because the 95% confidence interval is from 0.399 to 0.479 and the coefficient estimate is 0.4387, showing that 0 is not in the confidence interval.

Stem-height is barely statistically significant because the 95% confidence interval is from 0.004 to 0.102 and the coefficient estimate is 0.0531, showing that 0 is barely not in the confidence interval.

Stem-width is not statistically significant because the 95% confidence interval is from -0.031 to 0.077 and the coefficient estimate is 0.023, showing that 0 is in the confidence interval.

Stem-area is statistically significant because the 95% confidence interval is from -0.201 to -0.052 and the coefficient estimate is -0.1266, showing that 0 is not in the confidence interval.

Thus, stem-height and stem-width are perhaps not necessarily good predictors of mushroom class, and stem-area might be a better feature that incorporates stem-height and stem-width and is a better predictor of mushroom class.

2.5 Models of Your Choice (2 Distinct Models)

I chose to implement K-Nearest Neighbors Classifier and Decision Tree Classifier, as they are both models that perform well on data that is heavily categorical, and they are both highly interpretable. KNN typically achieves high accuracy, and it is also non-parametric, meaning it makes no assumptions about the underlying distribution; thus, it might fit the data better. Decision Tree is inexpensive in terms of making computations since it uses a greedy algorithm.

2.6 Ensemble Method (1 Ensemble Method)

I chose to implement AdaBoost Classifier as my ensemble method. Boosting is a great way to improve model performance by sequentially combining several weak decision trees into a single strong learner that can generalize well to new, unseen data. AdaBoost is the best out-of-box classifier that allows a classifier to learn from its shortcomings and improve, thus being a good option for this mushroom classification problem.

2.7 Hyper-Parameter Tuning

I tuned the hyperparameters for each model using GridSearchCV. Grid search tests all combination of the specified hyperparameters and selects the best values for the hyperparameters. Cross-validation is a technique where the training data is split into several subsets and one subset is reserved as test data. In the next iteration, a different subset becomes the test data. The best parameters I found include the following:

- **KNN:** algorithm: auto, n_neighbors: 2, weights: uniform
- **Decision Tree:** criterion: gini, max_depth: 5, splitter: random
- **AdaBoost:** n_estimators: 11

3 Results

The evaluation metrics for each of the models are as follows:

3.1 KNN

- **Accuracy:** 0.5584008843036109
- **Precision:** 0.6783375314861461
- **Recall:** 0.43365539452495977
- **F1 Score:** 0.5290766208251473

3.2 Decision Tree

- **Accuracy:** 0.7430913780397936
- **Precision:** 0.8003511852502195
- **Recall:** 0.7339774557165861
- **F1 Score:** 0.7657286854262915

3.3 AdaBoost

- **Accuracy:** 0.6625829034635224
- **Precision:** 0.7222901029848141
- **Recall:** 0.6663446054750403
- **F1 Score:** 0.6931903844543094

Decision Tree seems to perform the best based on the metrics; however, since we specified the "splitter" hyperparameter to be random, splitting at each node of the decision tree is not consistent. AdaBoost seems to have fairly good and consistent performance across all metrics, while KNN seems to have somewhat poor performance on the data.

4 Conclusion

I would use the AdaBoost Classifier for my adventure through *Mushroomia*. It has relatively good performance as evaluated by all four metrics, and is not dependent on any random factors. It has the best accuracy out of the non hyperparameter-tuned models.

One limitation of this project is the raw data. If there was more data, it might have been better to remove null values by dropping rows, since using the mode to impute nulls for categorical features ends up skewing the distributions. However, since the datasets would have been very small if rows were removed, I chose not to do that. In addition, the labels were imbalanced, causing the models to perform poorly. I would have oversampled the edible mushrooms or undersampled the poisonous mushrooms.

In terms of increasing model performance, GridSearchCV is very computationally expensive because it has to test all combinations of hyperparameters and apply cross-validation. If it had faster performance, I could have tested more hyperparameters.