CS M148 Project 3 Report

Matthew Craig
mcraig90505@g.ucla.edu

1 Introduction

The identification of edible mushroom species is no *truffle* matter. While some species may make for a delicious soup, others *spore*-t deadly levels of toxins. Without *mushroom* for error, an ill-informed *fun-guess* could have deadly consequences.

In this work, we investigate the use of machine learning models to correctly classify mushroom samples as edible or poisonous in order to aid in the study of mycology and to help identify potential food sources. We use as our dataset the Secondary Mushroom Dataset from the UCI Machine Learning Repository (Dua and Graff, 2017). For this project, we were given a pre-split train and test data set composed of 50,213 and 10,856 observations, respectively.

We first perform exploratory data analysis with scatterplots and bar charts to understand the distribution of the mushroom features and their usefulness in predicting the class label of edible or poisonous. We then engage in feature engineering to isolate further useful predictive features to apply machine learning to. Additionally, we perform statistical hypothesis tests on some features to confirm their predictive usefulness.

We investigate the use of several classes of supervised machine learning models for classification, including Support Vector Machines, Nearest Neighbor classification, Naive Bayes, Neural Networks, and Decision Tree ensembles. We select the best models based on initial results and engage in grid search hyperparameter tuning with cross validation to find the best performing models. Our hyperparameter tuning results in absolute improvements of single digit percentages on held-out training data. Finally, we apply our best models to the unseen test data and demonstrate that a Multi-Layer Perceptron neural network can achieve better than majority class guess performance.

2 Methodology

This section describes the methodology of our data exploration, machine learning pipeline, and model selection process.

2.1 Data Exploration

We chose to explore the categorical and numerical features of our data separately. Nine categorical features contained null values. We declined to impute null values before visualization, and instead represented null values as their own category within each feature.

For each of the 17 categorical features, we created paired bar charts showing the distribution of the categories within the feature and the percentage of mushrooms in each category with a label of edible. We also added a reference line with the percent edible for the entire dataset to each plot. The distribution bar plots allowed us to learn which features were rare, and to visualize the frequency of null or missing values among the categorical features. The paired plot of percent edible for each category gave us a first-order approximation of the importance of each category, and of the feature as a whole. By observing how much each bar deviated from the mean of the overall dataset (either higher or lower), we could gain a sense of which categories were good predictors of the class label and which added barely any information.

We noticed several categories across all the features with uniformly all edible or inedible labels, resulting in an edible percentage of 1.0 or 0.0, respectively. Unsurprisingly, these tended to be the smallest, both in relative terms compared to the other categories of their respective features, and in absolute number of examples in the training set. Most such ”perfect categories” were a few hundred samples, with the largest reaching only about 2,000, less than 5% of the dataset. Many categories were

within a few percentage points of the overall mean, with most deviating by less than 10%.

For the three numerical features, we created pairwise scatterplots with each point colored according to the class label. We also plotted two smoothed, fitted, overlapping distributions of each numerical feature, one for each class label. This approach allowed us to visually inspect for clusters of similar class labels. We identified two clusters of edible mushrooms, one with very high cap diameters and short stem heights and the other with small cap diameters and tall stem heights. The combined number of both clusters is only a few hundred samples, with most mushrooms of both classes falling somewhere in between. The plots of overlapping fitted distributions confirmed this finding, with the distributions of cap diameter and stem height showing longer, thicker right tails for edible mushrooms. The stem width feature did not show any particular clusters of interest.

We also plotted the correlation matrix of the three numerical features with the class labels. We found all three features to have a low (but positive) correlation with the class label edible. Stem width and cap diameter had a very high correlation at 0.78. Cap diameter and stem height had a correlation of 0.46, and stem width and stem height showing the lowest at 0.36. The high correlation of cap diameter and stem width was of potential concern for the presence of collinearity, which we addressed later.

2.2 Data Augmentation

In service of our feature engineering, we performed additional data visualization. We focused on interaction terms with one of the three binary features: has-ring, does-bruise-or-bleed, and veil-type (for which we treated null as its own category). For each feature, we split the dataset according to the binary feature and plotted the same bar charts of the percentage of mushrooms with a label of edible for each of the categories of the remaining features as before. This revealed a much larger number of "perfect categories", as well as significant differences in means for the same feature category between the two subsets. Most of these perfect categories were small, only a few hundred examples.

We focused our feature engineering on these perfect categories. We constructed eleven new features composed of an interaction between one of the binary features and another categorical feature. These were split roughly evenly between the three binary features. In several cases, we combined multiple categories of the same categorical feature together when they were all perfect categories (for example, when has-ring was true, cap shapes b, s, and o were all perfect categories with mean 0.0, so we combined them into a single interaction feature). We also created four interaction features for feature categories with significant differences in average percent edible when splitting across a binary feature. We focused on only the largest (30% or greater difference of means) and most impactful (more than 1400 training examples in the smaller category) differences, to ensure that the feature would be generally useful. Finally, we added four additional features to capture the perfect categories present in the features we later dropped in preprocessing.

We focused on this approach to feature engineering primarily because it created a "shortcut" to correct classification for some significant portion (single-digit percentage) of the training dataset. It would likely be more difficult for some of the models to identify these interactions across multiple categories since many of the perfect categories were very rare, so lumping them together helps by creating a more common feature. This is more useful for the individual classifiers like support vector machines and neural networks than for the ensemble decision trees, since the decision trees are already very capable of finding interactions between categories, but lumping similar pure categories together in the same feature still allows the decision tree to split several categories at once, instead of one at a time. This results in shallower decision trees and more accurate random forests.

2.3 Preprocessing

Our data pipeline is relatively straightforward. We dropped columns with more than 80% null values, which included stem root, stem surface, veil color, and spore print color. We elected to keep veil type despite the large number of null values because it was only a binary feature (null and u), and we felt that a null value was a reflection of the absence of the feature. For the other features with null values, we imputed the value corresponding to none (for gill attachment, gill spacing, and ring type) or in the absence of a none type, the most common category (for cap surface). Finally, we standard scaled our numerical features and one-hot encoded our categorical features. For one-hot encoding,

we elected to drop null values or the none type (when one existed), or the most common category otherwise. Ignoring unknown values in the test data, this resulted in a total number of 108 features.

2.3.1 Principal Component Analysis

To address the large dimensionality of our feature space, and to eliminate the potential collinearity between features that we remarked on earlier, we performed a Principal Component Analysis reduction on our data. Using the standard rule-of-thumb from the literature, we selected the first 26 components which captured 80% of the variance in the data.

We compared the performance of several classifiers on held-out data between the original and reduced data using cross validation. Two of the single models we tested (SVM and K-Nearest Neighbor) performed better on the reduced data, as measured by a several absolute percentage point increase in accuracy. Ensemble methods using decision trees performed approximately equally on the original and reduced data. Based on these findings, we decided to proceed using the PCA reduced data.

2.4 Statistical Hypothesis Testing

We selected a subset of the features to hypothesis test using Logistic Regression. We tested all three numerical features, as well as gill-attachment, gill-spacing, gill-color, veil-type, and season. The unreduced data was used. The raw performance of the Logistic Regression classifier on the test data is very poor at 45%, which is worse than a majority class guess.

We found all of our feature subset to be extremely statistically significant, with p-values approaching zero for nearly every feature. The first three coefficients reported correspond to the numerical features: cap diameter, stem-height, and stem-width. Cap diameter had a strong positive coefficient relative to the other numerical features, which accords with the thicker right tail we remarked on during our data exploration. The coefficient of 0.46 corresponds to an increase in odds ratio of about 59%. The coefficient of stem width was similarly positive, but much smaller magnitude. Somewhat surprisingly, the coefficient of stem height was negative. The fitted distribution may provide a clue as to why: outside of the long right tail of the edible class, the poisonous class has a greater probability mass distributed to the right side.

The next six coefficients (x4 through x9) represent the one-hot encoded values of gill-attachment. Most of the coefficients had a small magnitude, with one strongly positive (x6=1.29) and one strongly negative (x7=-2.96), corresponding to categories e and p, respectively. The coefficient for category p is the largest magnitude in the model, and results in reduction in the odds ratio of about 95%.

Coefficients x10 and x11 represent the values of gill-spacing. Both had positive coefficients, with the coefficient of category d more than twice the magnitude of category c. Both were much more positive on average than the combined null and f categories.

The values of gill-color are captured by coefficients x12 through x22. Almost all the coefficients are negative, since we observed in our data exploration that the most common category w (which was the category dropped in encoding) to have a higher percentage edible than every other category except b (x18=0.3183). Four categories had a coefficient less than -1, corresponding to a decrease in odds of more than 63%. We can identify these categories as n, y, f, and e.

Coefficient x23 corresponds to the binary feature veil-type (dropping the null category). It has a strong negative value at -1.8474, corresponding to a decrease in odds of 84%. We can see this relationship borne out in the bar chart in the exploratory data analysis.

The last three coefficients, x24 through x26, correspond to the encoded values of season after dropping the most common category a. We can see that all three have positive coefficients (since category a had the lowest mean percent edible), the strongest coefficient being x24=1.54 corresponding to category w. This is the largest positive coefficient in the model, and corresponds to an increase in odds ratio of 366%.

2.5 Individual Classifiers

We initially explored four individual classifier models: a Support Vector Machine, K-Nearest Neighbors, a Naive Bayesian Classifier, and a Multi-Layer Perceptron Neural Network. We tested all four models on held-out data from the training set using 5-fold cross-validation. The SVM classifier performed the best at 58.1% accuracy, and the MLP classifier performed slightly worse at 54.8% on the reduced data. Despite the relatively good

performance of kNN and Naive Bayes, we chose the SVM and MLP classifiers as our two models to proceed with hyperparameter tuning, because they have more expressive power than the other two models. We believed that the SVM and MLP, as non-linear classifiers, would be capable of discovering more complex interactions between terms than the simplistic kNN and Naive Bayes. A larger number of hyperparameters for SVMs and MLPs gives us a larger model space to explore in search of a better classifier, while kNN and Naive Bayes are already closer to their upper limit on performance.

The kNN and Naive Bayes models also make assumptions that, based on our exploratory data analysis, we believed were not reasonable. The k-Nearest Neighbor algorithm assumes that closeness on some distance metric equates to similarity of class label. However, the large feature space makes any distance metric problematic to interpret. Dimensionality reduction using PCA helps to mitigate this somewhat, but the issue is still potentially present. The Naive Bayes classifier assumes that features are independent of each other given the class label (the independence assumption), but as we already demonstrated during feature engineering, the appearance of perfect categories and large differences in mean value as a result of splitting across binary features reveals that the data does not bear out this assumption. Thus, we elect to choose the two classifiers which make less restrictive assumptions about the underlying data distribution.

2.6 Ensemble Classifiers

We initially explored two ensemble classifiers, both using Decision Trees: a Gradient Boosted classifier, and a Random Forest classifier. We chose to use Decision Trees in our ensemble method because, as discussed previously, we believed that decision trees would be capable of capturing complex interactions between categorical features very well. Ensemble methods using decision trees are also very robust to overfitting, which is a desirable property.

We tested both ensemble models on held-out data from the training set using 5-fold cross-validation. Both models performed very similarly. Ultimately, we chose the Gradient Boosted classifier because we expected a greater performance uplift on PCA reduced data after hyperparameter tuning compared to the Random Forest classifier.

2.7 Hyperparameter Tuning

We performed a grid search over hyperparameters using 10-fold cross validation to find the best model. We used a larger number of cross validation folds in order to improve generalization and hopefully reduce overfitting to the training set. All three of our models achieved ¿60% cross validation accuracy with the best parameters.

2.7.1 Support Vector Machine

We searched a range of regularization parameters C, from 0.01 to 1.0, and tested two kernels, RBF and sigmoid. The best parameters were C=0.5 with a linear kernel.

2.7.2 Multi-Layer Perceptron Neural Network

We tested three different network architectures: one shallow and narrow with two hidden layers of 50 and 30 units; one wider with 100 and 50 hidden units; and one deeper network with three hidden layers of 64, 64, and 32 hidden units. We tested two learning rates (1e-3 and 1e-4) and two activation functions, ReLu and hyperbolic tangent. The best parameters were the smallest network architecture (50, 30), smallest learning rate (1e-3), and hyperbolic tangent activation function.

2.7.3 Gradient-Boosted Decision Tree

Since decision tree ensemble methods are relatively robust to overfitting, we explored a wide range of estimators, from 100 to 300. We tested subsampling the data for each estimator at 1.0, 0.9, and 0.8, limiting the features to a random subset or all features at each split, and limiting the depth of each tree to 3, 5, or 7. Finally, we tested a range of minimum split sizes (2, 100, 200, 400). The best parameters were the largest number of estimators (300), with a depth of 5, a random subset of features, a subsample of 80% of data, and a minimum split size of 200.

3 Results

Our final results for all of our models achieve better than random guess on the test data. The final accuracy of each model trained with the best hyperparameters is summarized in Table 1.

Only the Multi-Layer Perceptron model achieved better than majority-class guess on the testing data. It is interesting to note the gap between results of 10-fold cross validation results on held-out training data and final results on the test data.

Model	Acc	F1
SVM	48.68%	30.04%
MLP	61.63%	61.42%
GBDT	48.21%	28.46%

Table 1: Final accuracy and F-1 score.

The precision and recall scores of each classifier are summarized in Table 2.

Model	Precision	Recall
SVM	68.23%	19.26%
MLP	72.30%	53.38%
GBDT	67.84%	18.00%

Table 2: Final recall and precision.

Examining the precision and recall scores, as well as the confusion matrices of each classifier, gives us some insight into the poor performance of the models. The SVM and GBDT models both demonstrate a heavy bias in favor of predicting the negative label (inedible). Almost all of the true negative examples are predicted as negative, which leads to relatively high precision scores.

This observation makes sense when we consider that the training data is moderately class imbalanced, with about 58% of the examples in the training data being true negatives. By contrast, the testing data is imbalanced almost perfectly opposite, with 57% of the testing examples being true positives. This results in the SVM and GBDT models learning to predict negative for almost all the examples, except for some select positive examples. While this achieves ¿60% accuracy on the training data, the 15% absolute shift in class balance for the test data results in poorer performance. It is clear that the test data is not identically distributed to the training data.

The MLP classifier performs better than the other two models because it more properly categorizes most of the true positives as positive labels. This results in a better overall accuracy and a much better recall score, but consequently a much higher (absolute) number of Type I (false positive) errors. It is possible that the neural architecture is better able to learn its own feature representations, so the improvement is attributable to extracted features which perform better than our hand-designed features.

Our cross-validation strategy of using a grid search to tune the hyperparameters achieved single-digit absolute percentage increases in accuracy. Due to computational limitations, we were only able to explore a limited range of parameters for each model class. It is likely that a more fine-grained approach using a method other than grid search (e.g. binary search, random search, or coordinate descent over a larger range) could yield additional performance improvements from searching a wider range of hyperparameters and saving time testing suboptimal combinations.

4 Conclusion

Our final results show that the Multi-Layer Perceptron neural network architecture is the best performing classifier. However, the large absolute number of Type I errors is concerning for real-world applications of identifying poisonous mushrooms. In actual practice, the risk of a false positive (predicting a poisonous mushroom as edible) is much more severe than a false negative. We may prefer a model that minimizes Type I errors even at the cost of lower overall accuracy.

A better metric to optimize models for would be precision, since we want to reduce the likelihood of a Type I error. Based on precision, the MLP classifier still outperforms the other two models. However, a precision of 74% is not exactly reassuring when it comes to poisonous mushrooms. It is likely that by optimizing for precision (or more usefully, an F-score with $\beta < 1$ to prevent predicting all negative labels), we could achieve a much safer classifier. This approach could be best suited to a decision tree ensemble method which is well adapted to finding interactions of features that result in pure leaf nodes.

As previously discussed, this problem is difficult because the testing data is not identically distributed to the training data. It is likely that shuffling and resplitting the data so the overall class distribution is identical between the training and testing datasets would result in much better performance.

Beyond concerns about the class balance, we also noted a large number of null values present in several categorical features in the dataset. Nine features were missing more than 20% of their values, five were missing more than half, and four of those were missing more than 80%. We dropped the columns with the most missing values, but the large number of null values makes reliable estimation and especially interpretation difficult. The

large number of categories within several features (mainly the features corresponding to color) also makes it difficult to visualize interactions between features and complicates visualization and feature engineering.

In this work, we investigated the use of machine learning models to correctly classify mushroom samples as edible or poisonous. We are excited about future applications of machine learning models to other tasks. Despite the lackluster performance of our models on this dataset, we still feel like *champignons* for demonstrating our data science abilities.

Acknowledgments

We would like to acknowledge the efforts of the CS M148 Introduction to Data Science course teaching staff, especially TA Lionel Levine.

References

Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.

A Code

Our entire code is attached as Appendix A.