Ethan Wong
CS M148
305319001

CS M148 Project 3 Report

I. Introduction

Mushrooms are a delicious but potentially dangerous food that people from all over the world enjoy. In *Mushroomania*, there is an even wider variety of mushrooms to choose from. However, just like in the real world, not all of the mushrooms are safe to eat. The goal of this project is to analyze different characteristics of mushrooms to observe which ones are poisonous and which are to safe to eat. Information from encountered mushrooms are collected using the *Shroomster Pro Max*. There is also a large dataset known as *The National Archives on Mushrooms* that we can use to analyze trends and train data science models on.

In order to learn which mushrooms are poisonous and which ones we can safely consume, we will have to combine a variety of data science techniques to analyze the data at our disposal. The created models will have different levels of success in categorizing the mushrooms. Being able to classify the mushrooms will be extremely important as it can mean the difference between life and death.

II. Methodology

A. Data Loading, Splitting, Exploration, and Visualization

After loading the training and testing data, I spent some time examining the contents to get a better understanding of what I was working with. I first read the csv files into their own variables and looked at the first few rows of each dataset using the head() function. I also used the info() function to examine the data types and the amount of entries that were in each set. These are simple, clear methods for seeing examples of the values that can be in each column. Doing this also allowed me to see that there were at least a few columns that would have null values.

To visualize the data, I created histograms with matplotlib. Histograms are the charts that provide the most visual clarity for me, so I created a histogram for each of the categories. These were especially useful because they gave me an idea of how the data was distributed within each feature. I observed that most of the features were not very balanced among the separate categories. I also noticed that the ‘veil-color’ feature could be dropped since all of the mushrooms had the same color. My investigation into the data helped me better understand the data so I could approach it with the correct strategies.

B. Data Pre-Processing

Before getting into data models and classification, I processed the data so it would be easier to work with. The first step I took was to remove the ‘class’ column from the data and store it in a separate variable. This column tells whether a given mushroom is poisonous or not. It is important that this information is withheld when training models as the models will just observe this column and make classifications based on this single feature instead of truly understanding the rest of the columns. This is bad because it means the model will not generalize well, and perform poorly on test data. I also dropped the ‘veil-color’ column because all the mushrooms fell into the same category, so it would not provide much insight into whether a mushroom is edible or poisonous. Next I had to impute the data. There were many columns that had null values, so I replaced the NaN values with the most common value for the given column. I decided on this simple imputation strategy because I noticed in the histograms that many of the features had a single category that dominated - so it made sense to group null values in with the largest category. Finally, I created a pipeline for the data. I made sure to one-hot encode all the categorical features so that they were represented in numerical values. This step is crucial because it would not be possible to create models with non-numeric data.

C. Data Augmentation

I augmented the given data by adding two new features. First, I created a column named ‘stem-size’. This value is the product of the ‘stem-height’ and the ‘stem-width’. I also created a column called ‘mushroom-size’. This value is the sum of ‘stem-size’ and ‘cap-size’. I thought that these new features would be useful because they would give an overall sense of the size of the mushroom instead of its individual parts. I hypothesized that there may be a relationship between size and whether a mushroom is poisonous or not, so I created these features to investigate.

D. Statistical Hypothesis Testing

I used the statsmodel.api with the OLS() function to perform my statistical hypothesis testing. I performed the testing on five features: cap-diameter, stem-height, stem-width, (these three were provided in the dataset) stem-size, and mushroom-size (these were two new features that I created). I analyzed the numerical features to determine how significant the relationship between the mushroom size attributes and edibleness was. Running the OLS on these features yielded P-values of 0.000, 0.001, 0.004, 0.000, and 0.000. These were all less than the standard p-value threshold of 0.05, which means that there is a statistically significant relationship between these features and whether a mushroom is poisonous or edible. Performing these tests showed that it was indeed important to consider the size of a mushroom when classifying it, so my hypothesis from the data augmentation proved to be true.

E. Models of Your Choice

The first model that I chose to work with was K-Nearest-Neighbors (KNN). I picked this model because I had some experience with it from previous projects, and I was curious to see how it worked with this dataset. KNN typically does not work well when the dataset has many dimensions because it becomes difficult to determine which data points are actually the closest neighbors. The mushroom dataset has a sufficiently large amount of features, so normally using a KNN classifier would yield a subpar model. However, I reduced the dimensionality of the dataset using principal component analysis (PCA). I hypothesized that the PCA processing would make the data easier for KNN to work with and I was correct. The KNN model I created had an above-average accuracy score of 0.794 on the test data, so the dimensionality reduction definitely helped to produce a working model.

The second model I chose to work with was Support Vector Machine (SVM). Again, I selected this model because I had some experience with it from previous projects. I theorized that SVM might work well because it usually works well in high-dimensional datasets. I also knew that SVM would be slower as it would have to deal with the large mushroom dataset, and I wanted to see how much slower it would be than other models. Running the code for SVM was indeed slow (taking a couple minutes to finish executing). The results were slightly better than the KNN model, with accuracy, precision, recall, and F1 score on the test data being 0.819, 0.811, 0.783, 0.796 respectively. Both k-nearest-neighbors and the support vector machine performed relatively well.

F. Ensemble Method

The ensemble method I decided to use was random forest. I wanted to try using it after learning about it in lecture. I thought that random forest would be a strong model choice for this dataset since the dataset was quite large, and a decrease in decision tree interpretability would not cause any issues because I was mainly concerned with analyzing the model's performance. The model ended up with a r^2 score of 0.7376. While r^2 score does not tell the entire story of how well a model performs, it is usually a solid indicator. This r^2 score was satisfactory in telling me that Random Forest created a decent model. I found that generating a model with random forest was considerably slower than non-ensemble models such as KNN, SVM, and linear regression due to the amount of processing that needed to be done.

G. Hyperparameter Tuning

I performed hyperparameter tuning on the four models: logistic regression, KNN, SVM, and random forest. I used a similar approach for all four models. I used GridSearchCV with a variety of parameters and let it find the best score and the associated parameters. The models and their associated parameters can be found in the table below. Performing the grid search on these models was the most computationally expensive operation in this project by far as execution time spanned from minutes to hours.

Logistic Regression	{'max_iter': 100, 'solver': 'newton-cg'}
K-Nearest-Neighbors	{'n_neighbors': 10}
Support Vector Machine	{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Random Forest	{'max_depth': 20, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}

III. Results

Through my experimentation, I found that logistic regression actually performed the best in terms of accuracy. It had an accuracy score of 0.84, precision of 0.87, recall of 0.84, and F1 score of 0.85 on the test data. Logistic regression performed marginally better than KNN (accuracy=0.794), SVM (accuracy=0.819), and random forest (accuracy=0.743). In terms of the models that I chose, I would elect to use SVM as it performed better than KNN, but it did take a long time to execute. In a time crunch I believe that using KNN would be satisfactory because its performance was not that different from SVM and executing it would save some time. Of course, model selection can vary based on which statistics you want to be maximized. In my analysis of the data, accuracy is the most straightforward which is why I would select the model with the highest accuracy.

I chose to cross-validate the data using k-fold cross-validation. This was a simple but effective method that was discussed in class. It is easy to implement and not too computationally expensive. AS explained above, I also used GridSearchCV for the hyperparameter tuning. This is another form of cross-validation that I used in order to determine the best performing parameters for each model.

IV. Conclusion

For my journey through Mushroomania, I would choose to use the logistic regression model. I think this model provides a good balance of speed and performance as it had an accuracy score of 0.84, precision of 0.87, recall of 0.84, and F1 score of 0.85 on the test data. Logistic regression is also computationally quick in terms of creating the model and optimizing it, which would be preferable if I was out in the wild foraging for mushrooms to eat — I would not want to wait around waiting for code to execute if I was starving.

There were some limitations with the project as I analyzed the mushroom data. For one, the data is imbalanced with a 57% to 43% split of the test data. This may have affected the results of my models as I ran them on the test data since it is always best to have data that is as balanced as possible. There were also quite a few null fields in the data that I had to impute. Ideally there should be no missing data in *The National Archives on Mushrooms*, but I had to work with the information I had available. The test data was also missing some values of categorical features. This is clearly not optimal, but some clever manipulation of the data allowed for a sufficient workaround.

Overall, this project was quite challenging but interesting at the same time. The lack of explicit instructions made it a bit confusing to tell what we were expected to do, but that added to the challenge of completing it. The Piazza and discussions also proved helpful with important clarifications. In the future I think it would be easier for the instructors and for students if there was a clearer rubric or a guideline of expectations regarding each section of the project. Thank you for designing a fun project!