CSM148 Project 3 Report

1. This was a data science project intended to give us the experience of exploring and analyzing a dataset from start to finish in a real-world setting. Our objective was to analyze various data points about mushrooms to find which ones were edible and which ones were poisonous. I first loaded, split, explored, and visualized the data to prepare for my analysis. Then, I pre-processed and augmented the data to make it ready to be fit by various models. Then, I conducted a statistical hypothesis test to investigate the relationships between variables. I used dimensionality reduction to make the data simpler. I then tested the data using 2 non-ensemble models and 1 ensemble model. Finally, I tuned the hyperparameters for the 3 models until I found the model that performed the best on the test data. My best model came from optimizing a KNearestNeighbor model as I will explain later.
2. Problem 2
    a. I originally called describe() and info() on the train data to explore the training data. I found that there were 50213 total data points, however, there were a lot of missing values. Specifically, the stem-root, veil-type, veil-color, and spore-sprint-color columns did not even have 10000 non-null values. I later dropped these columns. I repeated the same process for the test data as well. I noticed that there were a similar percentage of non-null values for these columns in the test data too. I also visualized the numerical data with a histogram. Based on the shape of the graphs, most of the numerical data points were close to 0, with relatively few outliers having very large values. I explored the data in this way to see the data both visually, and also to see the proportion of missing values. This helped me better understand the data and also plan how I was going to pre-process the data.
    b. I preprocessed the data by first changing all the labels to 0 and 1 instead of ‘p’ and ‘e’. For the gill-attachment and ring-type columns, I replaced any null values with ? as defined in the spec. Finally, I dropped the stem-root, veil-type, veil-color, spore-sprint-color, cap-surface, gill-spacing, and stem-surface columns because they were the remaining columns with missing values. I then pipelined the data to make it easier to process the data later. I used the ColumnTransformer and the OneHotEncoder to make the categorical data easier to process. All the column changes I made to the training data, I also did to the test data for consistency.
    c. I created the stem-width-height and the cap-stem additional features for both my training and test data. The stem-width-height is the product of the stem-width and stem-height. The cap-stem is the product of the cap-diameter and stem-height. These features both combine the physical dimensions of the mushroom in unique ways, which could be used to identify certain mushrooms. For example, an indicator of an edible mushroom is that the area of the stem is very large, even if the individual width or height is relatively smaller. These relationships can be explored by adding these new features.
    d. I conducted statistical hypothesis testing on the numerical features of the training data with the training labels. Specifically, I analyzed the cap-diameter,

stem-height, stem-width, stem-width-height, and cap-stem features to see if there was any meaningful correlation between those individual features and the training labels. The null hypothesis was that there was no meaningful correlation. However, the p-values for all of the features was below the alpha threshold of 0.05, which means we reject the null hypothesis. This means that all of the aforementioned features do have some correlation with the training labels. This means that the results were statistically significant and we can likely use this data to train a model to predict if mushrooms in the test set are edible or poisonous.

e. I implemented a DecisionTreeClassifier and a KNeighborsClassifier. They are both applicable to this problem because this is a classification problem, not a regression problem. The DecisionTreeClassifier can make decisions on thresholds of certain attributes to see if a good prediction can be made the KNeighborsClassifier will output a model that will predict the edibility of a mushroom based on similarly featured mushrooms, which makes sense in the real world too. Both models had a reasonable accuracy rate, but could definitely be optimized further.
f. I implemented a BaggingClassifier for my ensemble method. It uses a DecisionTreeClassifier as the default type of estimator. As explained earlier, a DecisionTreeClassifier can be a good fit for the model because it is a classification problem that could be explained by decisions based on certain features.
g. For the first DecisionTreeClassifier, I tuned the criterion based on gini, entropy, or log-loss, the max-depth from a range between 1 and 30, and the random state was set to 42 for repeatability. The best hyperparameters were entropy for criterion and a max-depth of 4 with a score of 0.543. For the KNeighborsClassifier, I tuned the number of neighbors from a range between 1 and 15, and the weight calculated uniformly or based on distance. The best hyperparameters were 4 neighbors calculating weights uniformly with a score of 0.497. Finally, for the BaggingClassifier, I tuned the number of estimators from a range between 1 and 10, each estimator having between 20% and 80% of the samples, and between 20% and 80% of the features. The best hyperparameters were 2 estimators with max samples and features of 20% with a score of 0.556.

3. The accuracy, precision, recall, and F1 score for each model used on the test data are listed in the PDF. The optimized KNeighborsClassifier had the best accuracy on the test data with 0.599, which is better than the 57% threshold given to us on Piazza. The other models did not even pass the 57% accuracy threshold. Additionally, since eating a poisonous mushroom could be fatal, we want to analyze the models with a metric that punishes false positives. This metric is precision. The optimized KneighborsClassifier also had the highest precision with 0.777. While the accuracy and precision values are the highest calculated, neither are high enough to actually be used in practice unfortunately. I used the KFold cross-validation strategy, which is automatically implemented under the hood when using GridSearchCV for hyperparameter tuning. I set the cv parameter to 5 for each GridSearchCV, so I did a 5-fold cross validation strategy. I

chose this strategy because it was built into the grid search and it allows the model to be validated against other models to ensure that the model is reflective of the dataset.

4. As mentioned earlier, I would use the KNeighborsClassifier with the best test accuracy of 0.599 and the best precision value of 0.777. While these are not great numbers, they are better than all of the other models I trained and tested. What this tells me is that I likely need a more complex model or need to widely increase the range of the hyperparameters that I tune to find the optimal model. Some limitations with the project are that I could only visualize a subsection of the data easily because there was not much numerical data. Additionally, I did not have as much time necessary to optimally train my model using various methods and even more hyperparameters. If this was a real-world project, I would likely train my model for weeks and months to identify real-life poisonous mushrooms. Also, a lot of the raw data was missing, so my models were not even able to account for some of the features of mushrooms that could actually have been vital in identifying a mushroom's edibility. Overall, I had a great experience with this project and learned a lot!