CIS 520 Fall 2020, Project Report

Determining Toxicity of Mushrooms from Physical Characteristics

Team Members:

- Jacob Glenn; Email: `jacobrad@seas.upenn.edu`
- Megan Harley; Email: `mlharley@sas.upenn.edu`
- Max Roling; Email: `mroling@seas.upenn.edu`

Abstract

Mushroom foraging has become an increasingly popular pastime over the past several decades, both in the United States and around the world. However, such a hobby can have insidious dangers, as there are some mushrooms, even in the United States, that are highly poisonous to consume. For example, from 1999 to 2016, there were 133,700 cases of exposure to hazardous mushrooms, with the majority of these exposures occurring through ingestion [1]. Distinguishing a poisonous mushroom from an edible mushroom can be extremely difficult, as there is no straightforward criterion for edibility in mushrooms. Instead, often chemical testing or experimentation is required to determine the toxicity of a mushroom, which is often expensive. The existence of a learned algorithm for determining mushroom toxicity is thus a useful and desirable tool. Therefore, in this project, we will examine a data set of mushrooms, which provides a collection of physical attributes for each mushroom, as well as its binary classification as either poisonous or edible. Our goal will be to apply various machine learning methods to this data to classify mushrooms based upon their observable attributes, utilizing feature selection and various types of regularization. In so doing, we will use methods including Logistic Regression, K Nearest Neighbors, Kernel regression, Random Forests, SVM, Gradient Boosting, and an ensemble method for binary classification, adjusting each method to our use case and occasionally testing out several variants therein. After our analysis, we find that a random forest approach with a threshold probability set to emphasize the importance of avoiding false negatives yields the best performance, with an accuracy of 99.237%. This analysis also allows us to understand which attributes of mushrooms are most associated with edibility. Here, we find that the odor, ring type, gill color, and spore print color are most associated with edibility classification.

1 Motivation

While mushrooms often appear in cuisine around the world, a sizeable minority of the species of mushrooms are highly toxic to consume. Due to the vast quantity of species and appearances of mushrooms, classifying them as either poisonous or edible can pose a significant challenge. Current widely used methods for the classification of mushrooms are chemical determination, animal experimentation, fungal classification, and individual experiences [2]. However, these methods are highly costly and in some cases have questionable morality. Therefore, in this paper, we investigate the use of a machine learning algorithm to classify mushrooms by toxicity. Using a data set of mushrooms compiled from The Audubon Society Field Guide to North American Mushrooms [6], we will train a variety of algorithms on a training data set, where we impose the true ratio of poisonous to edible mushrooms observed in the real world. We will then assess the performance of these methods using both overall accuracy on a testing set and the sensitivity of the method. The goal of our analysis is to create an algorithm that classifies mushrooms by toxicity with a high level of accuracy, where, because this algorithm in use would be used to determine if a given mushroom were safe to consume, we place a priority on the correct classification of poisonous mushrooms in particular.

2 Related Work

The classification of mushrooms is a problem that has been studied widely, both with our particular data set and others, and many models have been used to this end. Much work has been done informally, on sites like Kaggle, and these notebooks describe various modeling approaches to this particular data set. However, there have also been academic papers published regarding the use of machine learning methods to address the problem of binary classification of mushrooms. We will outline several of these approaches in this section.

First, in the paper *Mushroom Classification Using Feature-Based Machine Learning Approach [4]* , the researchers use images of various mushrooms for classification. In their paper, they describe a simple (Pre-CNN era) method to successfully classify these images into poisonous and non-poisonous categories. Specifically, they found that an SVM classifier had the best performance in their analysis, as compared to KNN, Logistic Regression, Decision trees, and ensemble classifiers. This suggests that, in our analysis, a support vector machine (SVM) classifier may prove powerful, though, as the data set is quite different, we will want to test other approaches as well. Now, we note here that, while this data set is fundamentally different from the data set we use in this report, as we are working with categorical variables instead of images, the goal of this study is the same as the goal here. The results of this particular study suggest that the prediction of mushroom edibility based on visible features of a mushroom (which here are coded as categorical variables instead of features in an image) is a tractable problem, and therefore one that will allow for a successful analysis and classification.

Next, in *Mushroom Toxicity Recognition Based on Multigrained Cascade Forest[2]*, the researchers employ our particular data set to predict toxicity using a multi-grained cascade forest approach to achieve a $> 98\%$ accuracy and a very high AUC. They also test out logistic regression and SVM, though these methods achieve lower accuracy. These results suggest that a forest or ensemble approach could be powerful for our classification task. This paper, as well as others, show that this data set is highly useful for bench-marking various ML algorithms and comparing them, as well as testing new adaptations to existing algorithms. They also achieved good results with an SVM and logistic regression.

With this past work in mind, we now proceed first with an exploratory analysis of our data set and then turn to the implementation of a variety of models for supervised binary classification.

3 Data Set

The data set we are using was contributed to the UCI Machine Learning repository on April 27, 1987. This data set is located at: https://www.kaggle.com/uciml/mushroom-classification. The data set is comprised of descriptions of hypothetical samples of mushrooms which correspond to 23 different species of gilled mushrooms which are in the Agaricus and Lepiota families. The descriptions were compiled using The Audubon Society Field Guide to North American Mushrooms [6]. Each mushroom species is classified as either certainly edible, certainly poisonous, or unknown (and therefore not recommended for consumption). In our data set, these mushrooms of unknown toxicity are also recorded as being poisonous for this reason. Note that, according to the field guide, there is no simple, easy to follow rule for determining the edibility of a mushroom, meaning that the solution to classification will not be a simple check of some specific characteristic or combination of characteristics.

The goal of this analysis is, using the characteristics of mushroom samples recorded in the data set (a list is provided in the appendix to this paper), to categorize the mushrooms as either edible or poisonous. Such an algorithm could then be employed by mushroom foragers to determine whether mushrooms found were likely either to be safe to eat or poisonous. We would also like to understand which features are most indicative of a poisonous mushroom.

The data set has 8,124 observations, with 22 categorical attributes recorded for each observation. Of these observations, 52% were edible and 48% poisonous. The attributes that are provided give information about the appearance and scent of the mushroom itself, as well as the environment in which it was found.

Some key attributes that will be of note in our analysis are odor, gill color, and ring type. After one-hot encoding each of these attributes, there will be 117 binary features created implicitly. For the reference of the reader, we also include a diagram showing what some of the parts of the mushroom described by the characteristics refer to below.

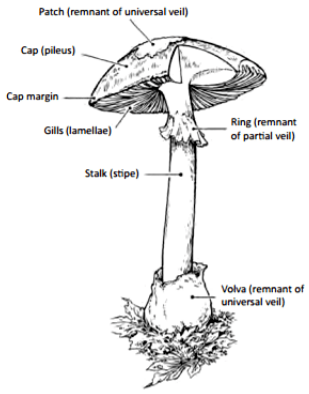


Figure 1: General morphology of Agaricus mushroom [3]

Before we proceed with the binary classification task of this project, we would like to get a better understanding of the data set we have through some simple exploratory analysis. A main goal of this project is to get a better understanding of which features of mushrooms are most useful in determining the edibility of that sample. Therefore, we would like to get some idea of how each of the features is correlated with the classification of the mushroom (as poisonous or edible) as well as how each is correlated with the other features. However, note that we have all categorical data. If we were to compute the correlation between one-hot vector encodings of each feature, we would get a unwieldy and difficult to interpret matrix of correlation values. Therefore, we instead turn to Cramér's V measure of association between categorical variables, which, like correlation, is symmetric and assigns a value between 0 and 1 to a pair of two variables. This statistic is computed for categorical variables as follows:

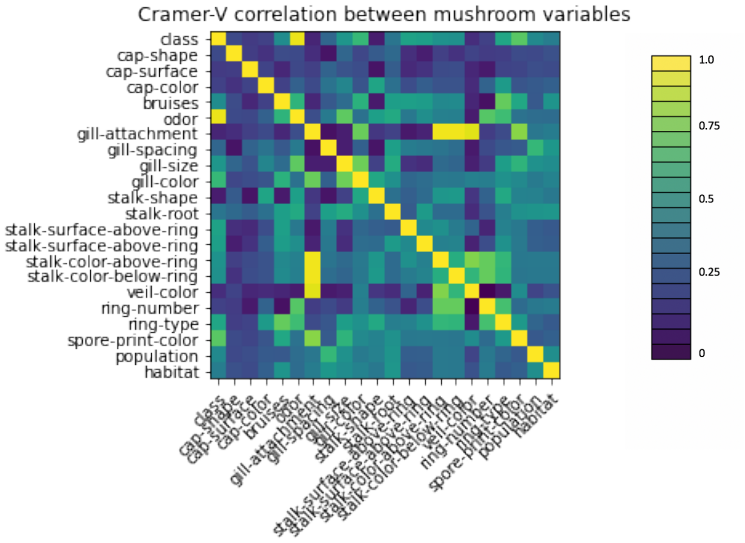
Let X and Y be categorical variables where X can take r values, which we will denote X_i, and Y can take k values, which we will denote Y_j. Then, for $i \in \{1, \ldots, r\}, j \in \{1, \ldots, k\}$, we define $n_{i,j}$ to be the number of times (X_i, Y_j) is observed in our data set. We denote n_i to be the frequency of X_i, n_j to be the frequency of Y_j, and n to be the total number of observations. Then,

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{k} \frac{\left(n_{i,j} - \frac{n_i n_j}{n}\right)^2}{\frac{n_i n_j}{n}}.$$

Then, Cramér's V measure is given by:

$$V = \sqrt{\frac{\chi^2/n}{\min(k-1, r-1)}}.$$

For each pair of variables in our data set (excluding veil-type, as there is only one category that appears in our data for this variable), we compute this measure to get an understanding of the association between our variables. The following figure shows these associations.



The above plot shows that odor is highly associated with class, suggesting that this variable will be of high importance in our classification methods. Other variables with high levels of association with edibility class are ring type, gill color, and spore print color. Before we proceed with our machine learning approaches to this classification problem, we will have a closer look at each of these variables to see if we can understand, at least simply, how they relate to edibility classification. Note also that we observe a high level of association between gill attachment type and the variables describing the stalk color above the ring, the stalk color below the ring, and veil color. This clear association could aid us in feature selection and provides some information of interest in how characteristics of mushrooms relate. Now, we proceed to examining those variables highly associated with edibility class in our data set. We begin with the odor of the mushroom.

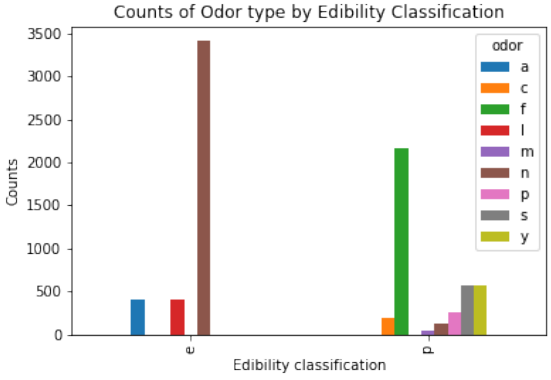


Figure 2: Odor type by edibility classification. On the x-axis, “e” signifies edible, “p” signifies poisonous. The odors listed in the legend are identified by letters, which stand for: “a” - almond, “c” - creosote,“f” - foul, “l” - anise, “m” - musty, “n” - none, “p” - pungent, “s” - spicy, “y” - fishy.

From the above plot, we can see that the odors almond, anise, and none are strongly associated with edibility, while the odors creosote, foul, musty, pungent, spicy, and fishy are more associated with being poisonous. There is a very clear distinct between the type of odors observed with poisonous mushrooms and those observed with edible mushrooms, which corroborates the high level of association we computed. Next, we consider the ring type of the mushroom.

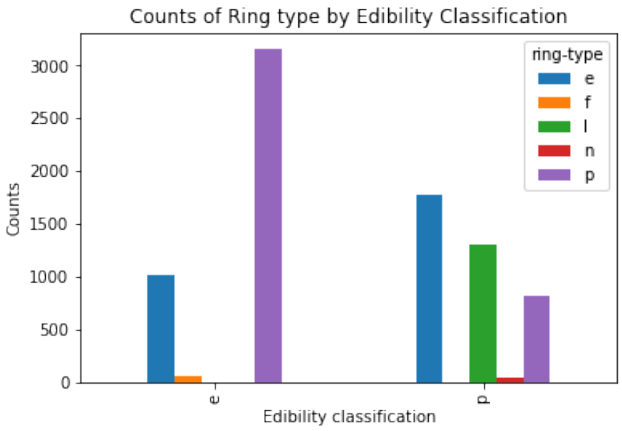


Figure 3: Ring type by edibility classification. On the x-axis, "e" signifies edible, "p" signifies poisonous. The ring types listed in the legend are identified by letters, which stand for: "e" - evanescent, "f" - flaring, "l" - large, "n" - none, "p" - pendant

While the frequencies for both toxicity classes of evanescent and pendant ring types suggest that these types do not provide much information about the edibility of a mushroom, we can see that a large ring is strong evidence for a mushroom being poisonous within our data set. Furthermore, a flaring ring type is evidence of an edible mushroom and no ring is evidence of a poisonous mushroom, though the counts within the data set of these kinds of mushroom are too low for this evidence alone to be a strong case for edibility.

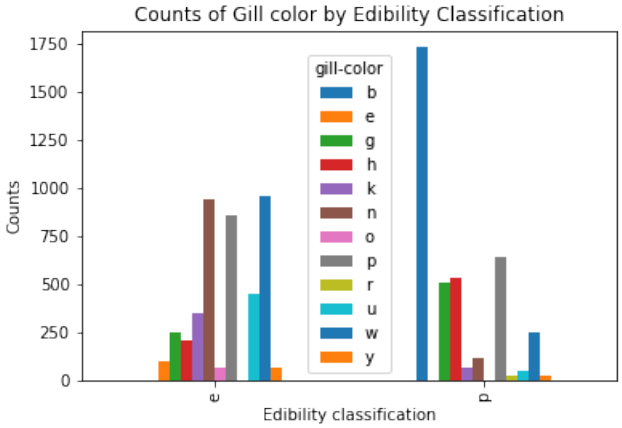


Figure 4: Gill color by edibility classification. On the x-axis, "e" signifies edible, "p" signifies poisonous. The gill colors listed in the legend are identified by letters, which stand for: "b" - buff, "e" - red, "g" - gray, "h" - chocolate, "k" - black, "n" - brown, "o" - orange, "p" - pink, "r" - green, "u" - purple, "w" - white, "y" - yellow

Next, we look at the frequencies of gill colors of the mushrooms in our data set by toxicity class. From the above plot, we can see that, while most gill colors appear in both the edible and poisonous classes, a gill color of buff is strong evidence that the sample is poisonous, while brown and white gills are more likely to signify edibility. Finally, we will look at the difference between our two toxicity classes by the spore print color of the mushrooms.

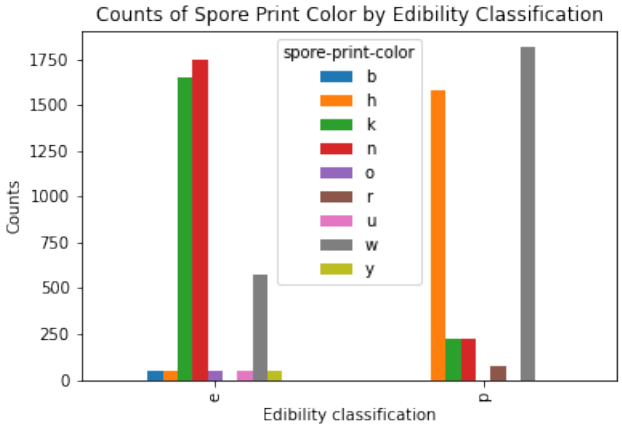


Figure 5: Spore print color by edibility classification. On the x-axis, “e” signifies edible, “p” signifies poisonous. The spore print colors listed in the legend are identified by letters, which stand for: “b” - buff, “h” - chocolate, “k” - black, “n” - brown, “o” - orange, “p” - pink, “r” - green, “u” - purple, “w” - white, “y” - yellow

By the above histogram, split out by edibility classification, we can see that a spore print color of “chocolate” is strongly associated with being poisonous. On the other hand, spore print colors of black and brown are more highly associated with a mushroom being edible. Therefore, these particular spore print colors may aid in the classification of mushrooms by edibility. From the above basic analysis of our data set, we have gained some understanding of how various traits of a mushroom are associated with its edibility. Now, we proceed to our main problem of predicting the classification of a mushroom sample as either poisonous or edible based on the traits included in our data set.

Finally, to more easily visualize this data, we did a principal component analysis on the testing dataset and produced 2D and 3D scatter plots. We chose to model the testing dataset as it contains equal parts poisonous to benign mushrooms, and makes it easier to visualize the distinctions between both classes. In both of the below plots, the yellow points indicate poisonous mushrooms and the purple points represent benign mushrooms. While there are certain clusters of each mushroom type visible in the plots, the problem is obviously not trivial, as there is a fair amount of overlap between the two classes. While this is exaggerated by reducing to a much smaller dimensional space, we can be reasonably confident that in a higher dimensional space, this overlap still exists to some extent, and can make the prediction problem more challenging. These plots foreshadow that we probably won’t be able to attain a 100% accuracy with any of our models, and furthermore the efficacy of a model will be gauged by how well they classify the ”harder” mushrooms, i.e the points that are very close on these plots but have opposite classes. However, more capable models will most likely be good at this and attain a high degree of accuracy.

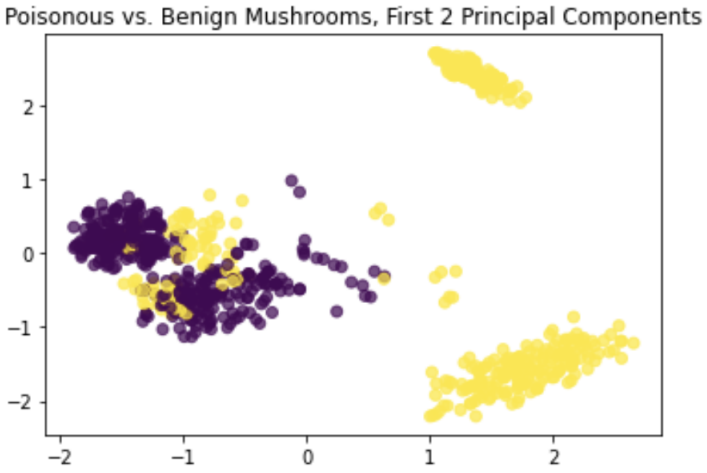


Figure 6: Training data projected into two dimensions. Yellow datapoints indicate poisonous mushrooms, purple datapoints represent benign mushrooms.

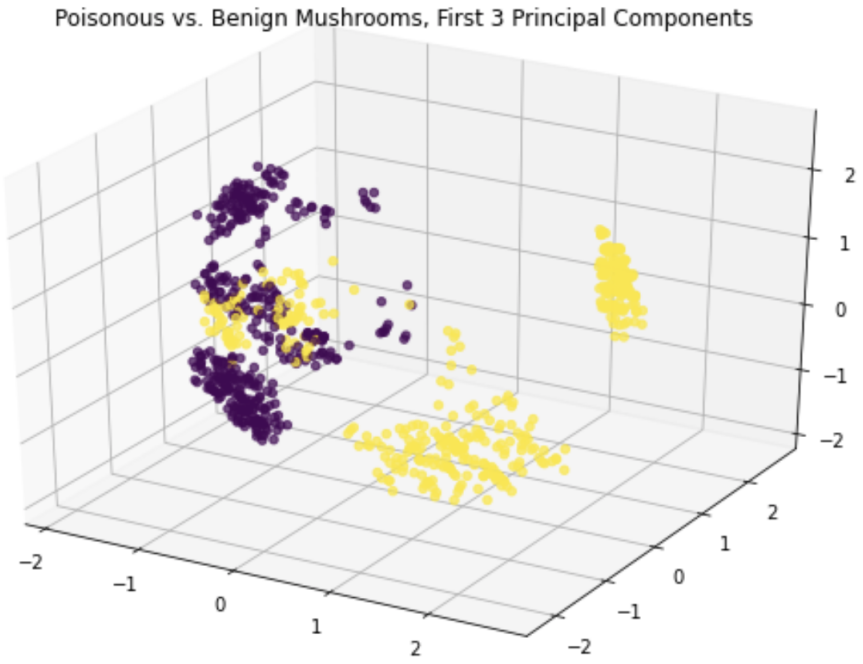


Figure 7: Training data projected into three dimensions. Yellow datapoints indicate poisonous mushrooms, purple datapoints represent benign mushrooms.

4 Problem Formulation

The goal of our analysis is to use the physical characteristics of mushrooms to determine edibility. Therefore our formulation is a supervised binary classification, where the label we wish to predict is whether a given mushroom is edible or poisonous. We will pre-process the data by transforming the categorical descriptions of the input variables to one-hot encoding vectors.

Now, we would like to formulate our analysis in a manner that reflects the true distribution of mushrooms in the region of interest (here, the United States). The data set, as contributed, has roughly equal proportions of poisonous and edible mushrooms. However, only between 1% and 2% of mushroom species are actually poisonous [6], so our data set reflects a distribution very different from reality. Furthermore, even a simple logistic regression trained on a subset of the data as-is yields almost perfect accuracy, because of the large number of poisonous mushrooms included. Therefore, in order to frame our analysis in a manner more akin to use-cases, we will adjust the distribution of edible and poisonous mushrooms in the following way:

Given the fact that the true percentage of poisonous mushrooms is only 1 to 2 percent, we will impose this distribution on our training data set. To do this, we will first split our data set into 80% for training, 10% for validation, and 10% for testing. Then, for the training data set, we will randomly remove 98.25% of the poisonous mushrooms, leaving a edible-poisonous mushroom split of approximately 1.5%. We do not adjust the ratio of edible to poisonous mushrooms in the validation or testing data sets. Our justification for this decision is that, to make the problem more realistic, we want to train our algorithms on a distribution similar to that in the real world, instead of using the highly skewed distribution given in the data set. However, we do not wish to discount the importance of accurately classifying poisonous mushrooms as compared to classifying edible mushrooms, particularly if this algorithm is used to determine whether or not a mushroom should be consumed, as falsely classifying a poisonous mushroom as edible could have disastrous consequences. Therefore, we maintain an approximately 50-50 split in the validation and testing sets to reflect this. In order to assess the performance of our various implementations of binary classification, we will consider both the accuracy and sensitivity of each. Note that we have encoded the "poisonous" class as the positive class, so our use case suggests a focus on the proportion of true positives that we get correct.

Because we have removed a sizeable portion of the training data set, we now have a training - validation - testing split of approximately 70% training, 15% validation, and 15% testing. We have chosen to leave out a validation set again to reflect the importance of classifying the poisonous class of mushrooms correctly. Therefore, when we tune our models using the validation set, we can more completely understand how our model performs on poisonous mushrooms and select our model hyperparameters accordingly.

5 Methods

Because the mushroom data set is manageable in size, computation speed in training and testing, as well as memory, are not primary concerns for our analysis. Rather, we will be focused on testing accuracy as well as sensitivity. Our baseline methods will be majority vote and Logistic Regression. We then evaluate and compare the accuracy of a tuned Logistic Regression with elastic net regularization, KNN, weighted KNN, Kernel Regression, Random Forests, SVM, and Gradient Boosting. Next, we will evaluate an ensemble method. In our analysis, we will introduce and evaluate a more tailored distance metric to our scenario and test out different regularization terms. Specifically, we will see if a KNN model using a weighted distance metric can achieve the same accuracy as a KNN model using a simple Euclidean metric. We will also see if using a different regularization term while training a Logistic Regression model can achieve the same accuracy as a Logistic Regression model with a ridge penalty. Because the loss function varies across our methods, we will specify the loss function separately for each where applicable.

6 Experiments and Results

In this section, we provide descriptions of the experimentation and results we achieved with the various methods mentioned in the previous section. This experimentation includes regularization, hyperparameter tuning, and feature selection.

6.1 Baseline

In order to assess more complex methods later in this section, we first test out two baseline methods of supervised binary classification to see how well these simple approaches fare.

6.1.1 Majority vote

First, suppose we simply elect to classify every mushroom in our testing set by the majority class observed in the training data set. For instance, using the training set described previously, we would clearly label every mushroom as edible. Doing this, we achieve an accuracy on the test data set of 51.106%. However, we misclassify every poisonous mushroom in our testing data set. For any realistic use-case for our analysis, this would make this method absolutely unusable.

6.1.2 Logistic Regression

Our other baseline method will be logistic regression with a ridge penalty, using the *sklearn* implementation LogisticRegression. The loss function here is cross-entropy. As our basic logistic regression does not have any critical hyperparameters to tune, we again do not need to use our validation set here. Running logistic regression, we achieve 93.612% testing accuracy, with the following confusion matrix, where we have 1 denoting the poisonous class and 0 denoting the edible class of mushrooms.

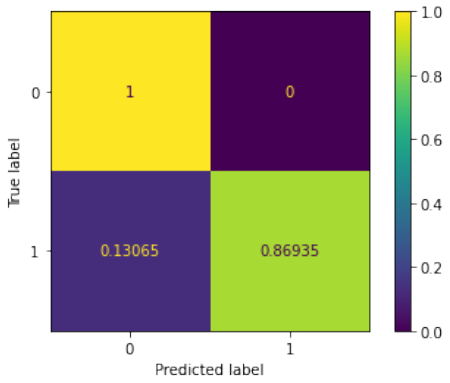


Figure 8: Confusion matrix for baseline logistic regression method. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

From the above figure, we see that we achieve 100% accuracy on the edible class of mushrooms and 86.935% accuracy on the poisonous mushrooms (sensitivity).

6.2 K Nearest Neighbors

6.2.1 K Nearest Neighbors using the Minkowski Metric

Note: The Minkowski distance of order p between two $1 \times n$ vectors x and y is defined as $d_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$.

Now, we will train a basic K Nearest Neighbors classifier on our data to attempt to classify elements as either poisonous or edible. We use the *sklearn* implementation KNeighborsClassifier for both implementations of KNN. Note that KNN has no loss function. To do this, we need to select several hyperparameters. In this case, we are restricting ourselves to the Minkowski distance metric. Therefore, we need to select the order of this distance, p, whether to assign all neighbors equal weights or to weight them by distance, and how many neighbors to use for each point. To determine the best values for these hyperparameters, we perform a grid search, training our model on the training data set for each combination of potential hyperparameter values

and testing the performance of the model on our held-out validation set. We then select the hyperparameter combination that yields the best accuracy on the validation set. In this case, our grid search yielded the following hyperparameters: number of neighbors: 3, weighting: by distance from point (as opposed to equal weights for all neighbors), and p = 1 (Manhattan distance).

When we run our model with these optimal hyperparameters, we get an accuracy on the entire test set of 97.420%. We achieve 100% accuracy on the edible mushroom class in our testing data set, and 94.724% accuracy on the poisonous class of mushrooms (which amounts to 21 poisonous mushrooms misclassified). We provide the normalized confusion matrix below for our results to show accuracy percentages for each class.

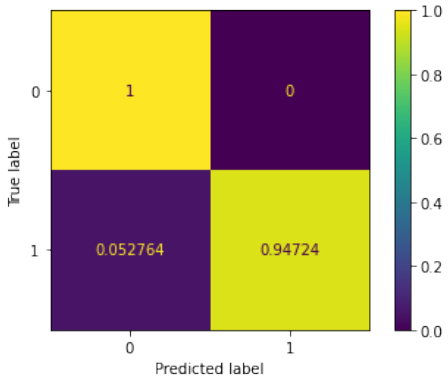


Figure 9: Confusion matrix for KNN classification method using the Minkowski metric. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

As we can see from the above results, the KNN approach clearly improves upon our baseline methods, even when using a fairly straightforward metric. However, the remaining misclassified poisonous mushrooms still represent a challenge to our analysis, as use-cases for our algorithm may require high levels of accuracy on the poisonous mushrooms in particular (for example, if it were to be used by foragers to determine the edibility of mushrooms they had collected). Therefore we proceed to examine other methods of binary classification.

We note here also that the use of the L1 metric on binary vectors (as we have one-hot encoded our data) is equivalent to the Hamming distance, which counts the number of positions in which two vectors differ. In our case, the Hamming distance between two mushroom observations will be equal to the number of characteristics for which the two mushroom samples differ. However, as we saw in our exploratory data analysis, not all characteristics provided in our data set are particularly associated with edibility, while some are highly associated. Therefore, we may wish to perform feature selection prior to implementing a classification algorithm.

6.2.2 K Nearest Neighbors using Weighted Minkowski Metric

Now, the above KNN classifier performed fairly well on our testing data set. However, we note that our choice of using a simple weighting scheme that is simply built in could be improved upon. We saw in our exploratory analysis of the data set, that some variables were more highly correlated with mushroom edibility class than others. Therefore, we propose the use of a metric weighted "by hand," using these correlations.

Specifically, we will proceed as follows: Within our training set, we will compute the correlation coefficient (Pearson's r) between each of our variables and the label (poisonous or edible). We will then use the absolute value of this correlation coefficient as the weight of each of our variables in a weighted Minkowski metric. This will effectively drop out variables with a correlation of 0 with the edibility class and diminish the influence

of those variables with lower correlations with our output class. Note that we are only using the correlations we can observe within the training data set, not the whole data set, as we want to avoid any kind of data snooping. Again, we will tune our hyperparameters using a grid search, with performance assessment for each combination of hyperparameters done using the accuracy on the validation set. Here, we again tune the following hyperparameters: number of neighbors and p value for the Minkowski metric. Unsurprisingly, we again arrive at optimal hyperparameters of 3 neighbors and $p = 1$, as in the previous KNN case. Then, when we test on our testing data set, we achieve 97.789% accuracy and a confusion matrix as follows:

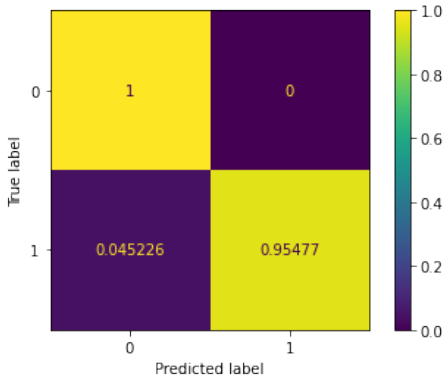


Figure 10: Confusion matrix for KNN classification method using the weighted Minkowski metric, where we weight by correlation coefficient. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

We can see that using this weighting by correlation does improve the performance of the KNN classifier, as here we only misclassified 4.523% (or 18 out of 398) of the poisonous mushrooms in the test set, as opposed to 5.276% (or 21 out of 398).

6.3 Logistic Regression

6.3.1 Logistic Regression with Elastic Net Regularization

Now, we consider implementing elastic net regularization on a logistic regression, as opposed to only L2 regularization, as we did for our baseline. Again, we use the *sklearn* implementation of logistic regression, and the loss function is cross-entropy. Recall that elastic net regularization combines L1 and L2 penalties. The benefit of doing this is to impose sparsity on our fitted coefficients and, in so doing, perform, in a sense, feature selection. Because we have 117 features in our data set, with relatively few observations, this may be beneficial.

For this method, we need to perform some tuning on our hyperparameters, specifically the hyperparameters for the coefficient on the regularization parameter, C, and the L1 ratio. These will describe how we impose our elastic net penalty. Now, to tune these hyperparameters, we perform a grid search over combinations of the two, training the model on our training data set and assessing performance through accuracy on our held-out validation set. Doing this, we arrive at the optimal choices $C = 10$ and an L1 ratio of 0.6. Training this chose model and evaluating it on our test set yields an overall accuracy of 98.894%. The below confusion matrix shows the performance by class of our model.

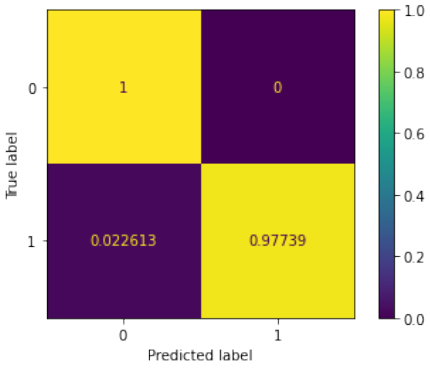


Figure 11: Confusion matrix for logistic regression with elastic net regularization. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

We can see that we achieve 97.739% accuracy on the poisonous class of mushrooms, which means that we misclassified 9 out of 398 poisonous mushrooms in our test data set. Thus, we can see that the elastic net regularization greatly improves the performance of the logistic regression model for our data set. We posit that the reason for such a marked improvement between the basic logistic regression implementation with L2 regularization and a tuned elastic net regularized logistic regression is two-fold. First, tuning our hyperparameters allows our model to fit the data better, resulting in improved performance. Secondly, elastic net regularization combines an L1 and L2 penalty, meaning it imposes scarcity on our weights and sends some to zero. As we saw in our exploratory analysis, some variables in our data set were clearly important to classification, while others were less so. Imposing and tuning the degree of an elastic net regularization means that we only effectively kept variables in our model that would aid in classification. Therefore, the improvement in performance of this iteration of logistic regression as opposed to the previous one could be due to the fact that we have dropped unimportant or unhelpful variables out of our model, in some cases completely (via elastic net regularization).

6.3.2 Building off of Elastic Net Logistic Regression by Changing Threshold Probabilities

After hyperparameter tuning, the above Logistic Regression with Elastic Net Regularization performs admirably. However, we can take this one step further by tuning another lesser-known hyperparameter of logistic regression, the threshold at which a decision is made. Logistic regression models predict the probabilities of the data point belonging to each class. Typically, the class with the highest predicted probability is selected – in this binary case, the class with a probability of greater than 0.5 will be selected. But altering this threshold value can have positive implications for the model and can make sense for certain purposes. In this instance, false negatives are much more dangerous than false positives. The event that a poisonous mushroom is labeled as benign is much worse than a benign mushroom being labeled as poisonous. To that end, it will be advantageous to decrease the threshold from 0.5 to something smaller, so as to decrease the false negative rate and (potentially) increase the false positive rate. More mushrooms will be categorized as poisonous, especially those for which the model is undecided (probabilities hovering around 0.5).

After a second round of hyperparameter tuning with the model on the validation set to find an appropriate threshold value, we arrive at a value of 0.01 as a solid threshold value, where mushrooms with probabilities higher than this will be classified as poisonous. This tuning was slightly subjective – the threshold chosen maximizes the overall accuracy, but is not the best threshold value for minimizing the false negative rate. Intuitively, the threshold value to minimize the false negative rate is the smallest possible threshold, but this also had a strong adverse effect on the overall model accuracy. Luckily, the threshold value of 0.01 that maximized the model accuracy also had very reasonable 1.8% false positive rate on the validation set, which was less than half the default threshold false positive rate, and also had a higher accuracy. The higher accuracy is attributable to the fact that the model tends to be much more confident in its predictions of benign

mushrooms than poisonous ones, meaning that when presented with a benign mushroom, the probability of it not being poisonous was exceedingly high, so high in fact that there were no false positives even with this threshold. This indicates that theoretically we could have chosen a smaller threshold, as 0.005 also yielded the same confusion matrix, and only a threshold value of 0.001 triggered 10 false positives. However, the value of 0.01 maximizes the accuracy while also being large enough to hopefully not trigger any false positives in the test set.

After examining this final model on the test set, we observe an overall accuracy of 99.03%, with a false positive rate of 1.8% and a sensitivity of 98.2%. This model yields some of the best results we have seen yet on this data set, and has the added benefit that it will purposefully reduce the number of false negative results at the expense of more tolerable false positive results.

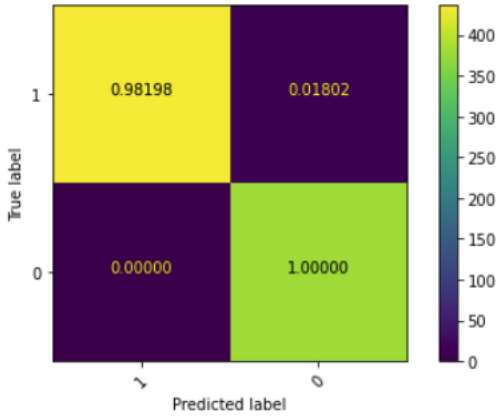


Figure 12: Confusion matrix for logistic regression with elastic net regularization and threshold of 0.01.

6.4 Kernel Regression

Next, we consider using Kernel Regression with a Gaussian kernel, where the loss function is cross-entropy. We need to tune the width parameter σ for our model. To do this, we test on our validation set 10 choices for σ from 0.1 through 5. When we test each of these choices, by training on our training set and then checking performance on the validation set, we achieve the best accuracy with $\sigma = 0.644$. We note that under this hyperparameter tuning, we have selected a kernel width that is comparatively small, suggesting we only want to take into account those observations quite similar in most features to the instance we wish to classify.

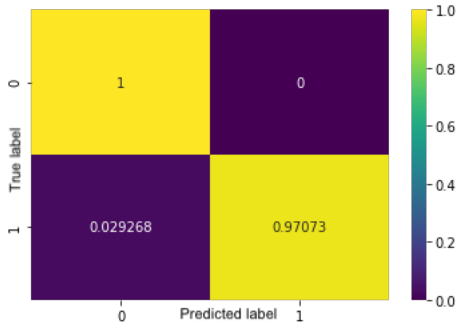


Figure 13: Confusion matrix for the fitted Kernel Regression algorithm results. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

The below figure shows the confusion matrix for the results on our testing data of this Kernel Regression model. With this choice of σ, we achieve 98.565% accuracy on the testing data set. We can see that this method improves upon the KNN implementations we tested, while our final Logistic regression with elastic net regularization performance attains higher accuracy and sensitivity. Here, we achieve 97.073% accuracy on the poisonous mushroom class.

6.5 Random Forests

Next, we implement a random forest classifier on our mushroom data set. Here, the loss function is Gini impurity. To do this, we use the textitsklearn method RandomForestClassifier. In order to select our hyperparameters, we perform a grid-search for the hyperparameters number of estimators, maximum tree depth, min_samples_split, and min_samples_leaf. We will also search over several different decision threshold probabilities, as in our logistic regression implementation, as this method also outputs probabilities. This resulted in the following hyperparameter selections: number of estimators: 300 , maximum tree depth: 8, min_samples_split: 5, and min_samples_leaf: 1, threshold probability: 0.01. When we implement the model with these choices of hyperparameters, we get a testing accuracy of 99.237%, which just beats the performance observed on our final logistic regression with elastic net implementation. The confusion matrix for the testing results is below.

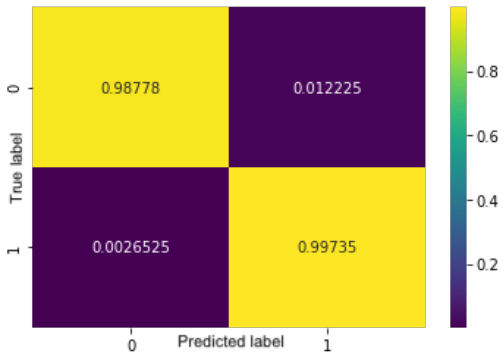


Figure 14: Confusion matrix for the random forest algorithm results. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

We can see that here, we have achieved our best-yet sensitivity, with 99.735% accuracy on poisonous mushrooms, an improvement from all previous methods. This translates to misclassifying only 1 poisonous

mushroom in the testing data set. However, because of our tuned threshold value of 0.01, we do misclassify 1.223% of the edible mushrooms in the testing data set as poisonous. This reflects the fact that, with such a low threshold, any mushroom the model is even slightly unsure about is classified as poisonous. This reflects the true costs of our use-case, as misclassification of a poisonous mushroom is much more expensive than misclassification of an edible mushroom.

We would also like to understand which features were most important in our classification. When we output the feature importance from our model, we get that the 5 most important features were foul odor, buff colored gills, a large ring type, no odor, and a spore print color of "chocolate". Therefore, we can see that, as anticipated, those features that we found to be most highly associated with the edibility classification of the mushrooms are in general those of highest importance in our model.

6.6 SVM

6.6.1 Hyperparameter-tuned SVM

Next, we consider a penalized Support Vector Machine approach to our problem, using the SVC method in *sklearn*, with hinge loss as our loss function. First, we will use an L2 penalty on a linear SVM. Doing this, we simply need to tune the regularization parameter C for our model. To do this, we test on our validation set various choices for C from 10^{-6} through 10 by factors of 10. If we test each of these choices, by training on our training set and then checking performance on the validation set, we achieve the best accuracy with $C = 10$. The below figure shows the confusion matrix for the results on our testing data of this penalized SVM model. With this choice of C, we achieve 98.894% accuracy on the testing data set.

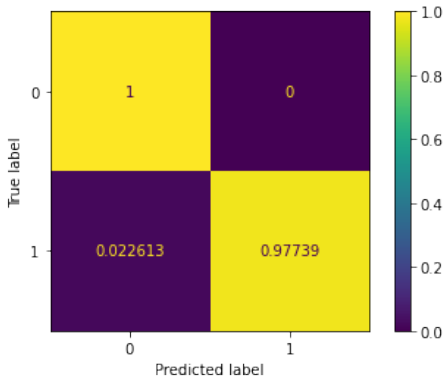


Figure 15: Confusion matrix for the penalized SVM algorithm results. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

From these results, we can see that we once again achieve 100% accuracy on the edible mushroom class in our testing data set. In addition, we achieve 97.739% accuracy on the poisonous mushroom class (which gives us 9 out of 398 poisonous mushrooms misclassified), an improvement from the random forest implementation, and identical to the performance of our logistic regression with elastic net penalty.

If instead we used an L1 penalty to impose sparseness on our coefficients, which in effect does feature selection, we might expect that we could improve the performance of the SVM somewhat, as we saw previously that in this data set, not all features are equally important. However, using an L1 penalty results in nearly identical performance, as both misclassify 6 poisonous mushrooms out of 370.

6.6.2 SVM with Resampled Data: SMOTE

Here, we try a common oversampling technique called the Symmetric Minority Oversampling TEchnique (SMOTE) [5], which aims to rebalance the data in the training set by "inventing" new, similar datapoints in the minority class to rebalance the dataset. The idea is to create new data, rather than merely oversampling the existing data, as that could lead to significant overfitting. When creating new datapoints, a point from the minority class is first selected, then a small number of its neighbors are computed using K-Nearest Neighbors. The paper doesn't exactly specify the K value to use, but common consensus is that 5 neighbors are selected. From these neighbors, one of them is selected at random. Next, the difference in values for each feature of the two datapoints (original datapoint and randomly selected neighbor) is computed. This difference is multiplied by a random number between 0 and 1, then added to the feature values of the original datapoint, to determine a new, random, but similar datapoint, which is labeled as part of the minority class. After re-selecting a new datapoint from the minority class and repeating this process many times, enough new datapoints are computed that the dataset is re-balanced. While other varieties of SMOTE algorithms exist such as borderline SMOTE, we present the original, baseline algorithm as it is the most widely available and well known variant.

This re-sampling methodology has the potential to fix the asymmetry problem of this data, but not without introducing new issues. Most notably, the synthetic datapoints generated could be significantly different enough from the original datapoints that exist – there is the potential for a new datapoint representing a benign mushroom to be added which is mislabled as poisonous. And even if this is not true, the noise introduced could lead to synthetic datapoints representing mushrooms that no not exist in the wild, as perhaps there is some combination of variables that do not describe any wild mushroom at all, but still appear in the dataset.

Despite using SMOTE to over-sample the training set, the model results are essentially equivalent as before. The hyperparameters of the SVM trained on this model remain largely unchanged from the previous model, with a C value of 10 still prevailing. The accuracy of the model on the test set is 98.83% with a sensitivity of 97.5%. It appears that SMOTE has had negligible impact, if any, on model performance. In the interest of saving space, we have chosen to omit the confusion matrix for this model.

6.7 Gradient Boosting

Now, we will implement gradient boosting for our binary classification utilizing the open source gradient boosting implementation XGBoost. The loss function is again cross-entropy. We will include both L1 and L2 regularization terms in training our model. In order to tune hyperparameters, we perform a grid search for a selection of hyperparameters, training the model on our training data set and determining the accuracy of the resulting model on the validation set. We tune the following hyperparameters: number of estimators, maximum tree depth, learning rate, and L1 and L2 regularization term coefficients. The tuned hyperparameters are: 100 estimators, a maximum tree depth of 5, a learning rate of 0.00001, L1 regularization coefficient 0.001, and L2 regularization coefficient 100. Because we have unbalanced data in our training set, we also set the parameter scale_pos_weight to the ratio of the number of negative cases (number of edible mushrooms) to the number of positive cases (number of poisonous mushrooms). Because we have adjusted our training data set to have roughly 98.5% edible mushrooms to 1.5% poisonous mushrooms, we set scale_pos_weight to 65.

Once we have tuned our hyperparameters, we then train our optimized model and compute the accuracy on the testing data set. This yields an overall accuracy of 98.526% and the following confusion matrix:

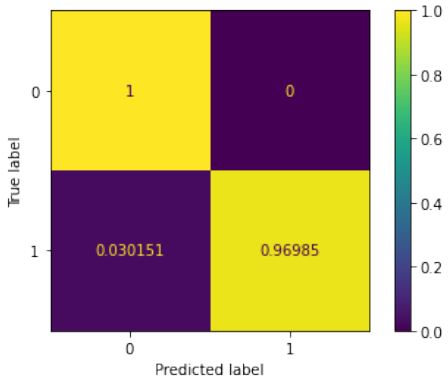


Figure 16: Confusion matrix for the gradient boosting algorithm results. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. The numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

We can see that while XGBoost improves upon the accuracy of our baseline majority vote and logistic regression methods, as well as random forests, it does not perform as well as the SVM approach. This is because we misclassify 3.015% (or 12) of the poisonous mushrooms. This method also allows us to look at feature importances. When we return these values, we get that the top 5 most important features in our model were having no odor, having a club stalk root, having a rooted stalk root, having a yellow cap color, and having an almond odor.

6.8 Ensemble

Next, we tried an ensemble method using Auto-SKLearn. This generated an ensemble model that performed very well on the training set, but only average on the test set. The Auto-SKLearn library employed does automatic hyperparameter optimization "automagically" (well, actually it uses a Bayesian hyperparameter tuning scheme, but we like that word), and constructs an ensemble model consisting of various other models, with an "importance" weight for each model. We ran the AutoML library for 10 minutes, and gave it a maximum of one minute per sub-model to train and optimize. The library requires only a testing and training set, so we provided a 70 - 30 train test split, with the hopes that under the hood it would create a similar validation set as to the previous models. The training accuracy was 99.97%, while the testing accuracy was only 89.68%. This indicates that the ensemble model is overfitting the training dataset. To be consistent with the other methods, the ensemble method has an accuracy of 97.26 % on the entire dataset. Below is a confusion matrix describing the model results on the test set.

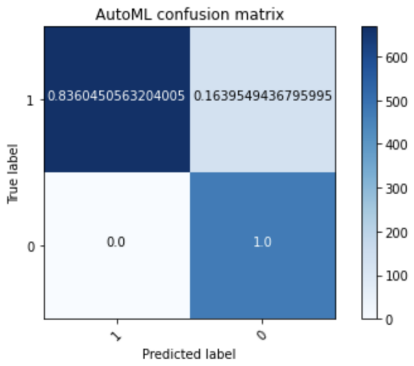


Figure 17: Confusion matrix for the ensemble boosting algorithm results. A label of 1 denotes the poisonous class, and a label of 0 denotes the edible class. As was true before, the numbers within the cells represent the proportion of samples with the given true label that were assigned the predicted label.

Below is an abbreviated table summarizing the ensemble model created, including the top 15 models from the ensemble. The rest have insignificant weights and do not contribute much to the prediction made.

Table 1: Summary table for ensemble model

Classifier Type	Weight	Hyperparameters
Passive Aggressive	0.12	C=0.005, loss=squared hinge, tolerance= 0.0002
Random Forest	0.12	min samples split=9, max features = 0.9
Decision Tree	0.08	min samples split=14, max depth factor = 1.6
Gradient Boosting	0.08	max bins=225, max leaf nodes = 8
Gradient Boosting	0.06	max bins=225, max leaf nodes = 18
Gradient Boosting	0.06	max bins=225, max leaf nodes = 7
Random Forest	0.06	min samples split= 7, max features = 0.395
Adaboost	0.06	max depth=8, num estimators =489
KNN	0.06	num neighbors = 3
Extra Trees	0.06	max features = 0.358, min sample split = 7
Random Forest	0.04	max features = 0.332, min sample split = 2
Gradient Boosting	0.04	max leaf nodes = 5, max bins = 255
Liblinear SVC	0.04	loss = squared hinge, C=797.6
Adaboost	0.02	max depth = 9, num estimators = 477
Gradient Boosting	0.02	max leaf nodes = 3, max bins = 255

7 Conclusion and Discussion

Provided below is a table summarizing the results of the various methods above, listing both the overall accuracy on the testing data as well as sensitivity. From this table, we can see that our two best performing classifiers, both in terms of overall accuracy and sensitivity, were logistic regression with elastic net regularization and thresholding, and random forests, again with thresholding. We do note however that, with the exception of our baseline methods of majority vote and logistic regression with L2 penalty, the performance of all of our methods is quite close.

Our inability to surpass 99.237% accuracy could, in part, be due to the fact that there are certain types of poisonous mushrooms that are in our data set that were not randomly sorted into our training set, meaning that our models were not trained to recognize them, leading to an incorrect classification. For instance, even when we changed the thresholding probability in the random forest model to 0.01, we still misclassified one

Table 2: Summary table of results

Method	Overall accuracy	Sensitivity
Majority vote	51.106%	0%
Logistic regression (L2 penalty)	93.612%	86.935%
Logistic regression (elastic net)	98.894%	97.739%
Logistic regression (elastic net with threshold)	99.028%	98.198%
KNN (Minkowski)	97.7420%	94.724%
KNN (weighted Minkowski)	97.789%	95.477%
Kernel Regression	98.565%	97.073%
Random Forests	**99.237%**	**99.735%**
SVM	98.894%	97.739%
SVM with SMOTE sampling	98.834%	97.50%
Gradient Boosting	98.526%	96.985%
Ensemble	97.26%	83.6%

poisonous mushroom, meaning our model was very sure (>99% probability) that this mushroom was edible. The improvement in performance we observed when we adjusted our probability threshold does however show us that, in general, our model was much more sure of its classification of edible mushrooms that it was sure of its classifications of poisonous mushrooms. This makes sense, as we changed the proportions of our data in the training data set to be imbalanced, with most observations coming from the edible class. Another related concern could be the size of the data set itself, as we only have a total of approximately 8,000 observations. While this is a large data set to compile by hand, this could mean that a training subset may not be entirely representative of the relevant population of mushrooms.

Overall, the success of the random forests and logistic regression models make sense based upon our data set to number of features ratio as well as our imposed class imbalance on training data. We saw here the success of changing the threshold probability for models which output probabilities. For an extension to this method, the use of regression to obtain values that "act" like probabilities for those methods that output only a classification instead of a probability could be implemented in future analyses. In addition, more exploration into asymmetric loss functions, different metrics, and regularization techniques could also yield even better results on binary classification for this data set.

8 Appendix

Table 3: Variables included in dataset

Variable name	Possible values and percentages in dataset
class	edible (51.8%), poisonous (48.2%)
cap-shape	convex (45%) , flat (38.8%), knobbed (10.2%), bell (5.6%), sunken (0.4%), conical (0.05%)
cap-surface	scaly (39.9%), smooth (31.5%), fibrous (28.6%), grooves (0.05%)
cap-color	brown (28/1%), gray (22.6%), red (18.5%), yellow (13.2%), white (12.8%), buff (2.1%), pink (1.8%), cinnamon (0.5%), green (0.2%), purple (0.2%)
bruises	no (58.4%), yes (41.6)
odor	none (43.4%), foul (26.6%), fishy (7.1%), spicy (7.1%), anise (4.9%), almond (4.9%), pungent (3.2%), creosote (2.4%), musty (0.4%)
gill-attachment	free (97.4%), attached (2.6%)
gill-spacing	close (83.8%), crowded (16.2%)
gill-size	broad (69.1%), narrow (30.9%)
gill-color	buff (21.3%), pink (18.4%), white (14.8%), brown (12.9%), gray (9.2%), chocolate (9.0%), purple (6.0%), black (5.0%), red (1.2%), yellow (1.1%), orange (0.8%), green (0.3%)
stalk-shape	tapering (56.7%), enlarging (43.3%)
stalk-root	bulbous (46.5%), missing (30.5%), equal (13.8%), club (6.8%), rooted (2.4%)
stalk-surface-above-ring	smooth (63.7%), silky (29.2%), fibrous (6.8%), scaly (0.3%)
stalk-surface-below-ring	smooth (60.8%), silky (28.3%), fibrous (7.4%), scaly (3.5%)
stalk-color-above-ring	white (54.9%), pink (23.0%), gray (7.1%), brown (5.5%), buff (5.3%), orange (2.4%), red (1.2%), cinnamon (0.4%), yellow (0.1%)
stalk-color-below-ring	white (54.0%), pink (23.0%), gray (7.1%), brown (6.3%), buff (5.3%), orange (2.4%), red (1.2%), cinnamon (0.4%), yellow (0.3%)
veil-type	partial (100%)
veil-color	white (97.5%), orange (1.2%), brown (1.2%), yellow (0.1%)
ring-number	one (92.2%), two (7.4%), none (0.4%)
ring-type	pendant (48.8%), evanescent (34.2%), large (16.0%), flaring (0.6%), none (0.4%)
spore-print-color	white (29.4%), brown (24.2%), black (23.0%), chocolate (20.1%), green (0.9%), yellow (0.6%), buff (0.6%), orange (0.6%), purple (0.6%)
population	several (49.7%), solitary (21.1%), scattered (15.4%), numerous (4.9%), abundant (4.7%), clustered (4.2%)
habitat	woods (38.7%), grasses (26.4%), paths (14.1%), leaves (10.2%), urban (4.5%), meadows (3.6%), waste (2.4%)

References

[1] William E Brandenburg and Karlee J Ward. Mushroom poisoning epidemiology in the united states. *Mycologia*, 110(4):637–641, 2018.

[2] Chenxi Huang, Yingying Wang, Jixiang Du, Hongbo Zhang, and Xiuhong Yang. Mushroom toxicity recognition based on multigrained cascade forest. *Scientific Programming*, 2020:8849011, 2020.

[3] MD SAIFUL ISLAM. Cultivation techniques of edible mushrooms.

[4] Pranjal Maurya, Nagendra Pratap Singh, Bidyut B Chaudhuri, Masaki Nakagawa, Pritee Khanna, and Sanjeev Kumar. *Mushroom Classification Using Feature-Based Machine Learning Approach*, volume 1022 of *Proceedings of 3rd International Conference on Computer Vision and Image Processing*. Springer Singapore :, Singapore :, 1st ed. 2020. edition.

[5] L. O. Hall W. P. Kegelmeye N. V. Chawla, K. W. Bowyer. Smote: Synthetic minority over-sampling technique. *Journal of Artifical Intelligence Research*, 16:p. 321 – 357, 2002.

[6] Barry L Wulff. The audubon society field guide to north american mushrooms, 1982.