Mushrooms: Edible, Poisonous, or Unknown?

Matthew Borja

School of Computer Science and Engineering
California State University, San Bernardino

Abstract

There exists a mantra in the world of mycologists: "Never eat a mushroom unless you can positively identify it." With approximately 14,000 species of mushrooms that we know of, the process of classifying them seems so daunting that someone's better off sticking to store bought mushrooms for their culinary needs. Unless, a supervised machine learning algorithm could be utilized to assist in classification.

The purpose of this project is to experiment with two different supervised machine learning algorithms, namely Decision Tree Classification and Naïve Bayes Classification, to determine which algorithm will excel in accurately classifying mushrooms as poisonous or edible.

1. Introduction

With thousands of different species of mushrooms, all with a plethora of phenotypes, the question I hope to answer is if it is possible for a supervised machine learning algorithm[1] to accurately predict poisonousness? Furthermore, how do the two algorithms, Decision Tree Classification and Naïve Bayes Classification, measure up?

The Mushroom Data Set is complementary of the University of California, Irvine Machine Learning Repository. The data set contains 8,124 data points. Of these, 2,480 data points are missing an attribute and will be removed from the data set, leaving 5,644 data points under research. There are twenty two attributes under consideration: cap shape, cap surface, cap color, bruises, odor, gill attachment, gill spacing, gill size, gill color, stalk shape, stalk root, stalk surface above ring, stalk surface below ring, veil type, veil color, ring number, spore print color, population, and habitat. The classification value is a binary value of edible or poisonous.

2. Data Set

The Mushroom Data Set contains the following attributes:

- Cap shape = bell, conical, convex, flat, knobbed, sunken
- Cap surface = fibrous, grooves, scaly, smooth
- Cap color = brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
- Bruises = T, F
- Odor = almond, anise, creosote, fishy, foul, musty, none, pungent, spicy
- Gill attachment = attached, descending, free, notched
- Gill spacing = close, crowded, distant
- Gill size = broad, narrow
- Gill color = black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow
- Stalk shape = enlarging, tapering
- Stalk root = bulbous, club, cup, equal, rhizomorphs, rooted
- Stalk surface above ring = fibrous, scaly, silky, smooth
- Stalk surface below ring = fibrous, scaly, silky, smooth
- Stalk color above ring = brown, buff, cinnamon, gray, orange, pink, red, white, yellow
- Stalk color below ring = brown, buff, cinnamon, gray, orange, pink, red, white, yellow
- Veil type = partial, universal
- Veil color = brown, orange, white, yellow
- Ring number = none, one, two
- Ring type = cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone
- Spore print color = black, brown, buff, chocolate, green, orange, purple, white, yellow
- Population = abundant, clustered, numerous, scattered, several, solitary
- Habitat = grasses, leaves, meadows, paths, urban, waste, woods
- Classification = edible, poisonous

3. Decision Tree Classification

"Decision Trees are a non-parametric supervised learning method used for classification and regression." [2] Given an input or instance $\vec{x}$ a decision tree will predict the output label y by beginning at a root node and traveling down to a leaf node. The result is a generally pyramidal shaped tree of leaf nodes at the ends and internal nodes used for splitting attributes among the upper levels.

Building a Decision Tree involves utilizing a top-down recursive divide-and-conquer approach in which an attribute is chosen for the root, instances are split into subsets and repeated until all instances have been labeled with a class. Throughout this process, the most important characteristics we want to maintain are size and the purity. A decision tree that is too large can often result in overfitting, which suggests the model is too complex and can lead to a large test error. Therefore an ideal Decision Tree is one that is as minimal as possible while accurately predicting the class of a given instance. Inherently, the purity of internal nodes is fundamental to both maintaining a small tree and accurately predicting an instance's class. At every branch, we want to choose an attribute that will result in the purest subnodes. The impurity of a subnode can be determined with a variety of measures, such as Entropy, Classification Error, and Gini Index. For this experiment, the Gini Index was utilized.

$Gini(N) = \sum_i^k \frac{|N_i|}{|N|} \times Gini(N_i)$

The Gini Index is calculated on every subnode of a given node. The maximum impurity a subnode can possess in the Gini Index is 0.5 which means it is comprised of half of one class and half of another class. We want a subnode with a Gini Index of zero for an optimal Decision Tree. A final Decision Tree is obtained when the data cannot be split any further and sometimes there will exist a number of impure leaf nodes, especially if identical instances have different classes.

The main advantages of utilizing Decision Tree Classification are that they are inexpensive to construct since they are O(n log n) to train and extremely fast at classifying unknown records since they are O(n) to traverse. [3] They are easy to understand visually since the output is a tree with straight-forward steps that anyone could follow. Finally they are resilient: they can both handle noise and redundant or irrelevant attributes. On the other hand, they are prone to overfitting, they can potentially become exponentially large if an optimal tree is not found, they can neglect interactions between attributes, and unbalanced data sets can generate biased trees.

4. Naïve Bayes Classification

"Naïve Bayes Classifier is a supervised machine learning algorithm, which is used for classification tasks. It is also known as a probabilistic classifier since it is based on Bayes' Theorem. [4] Furthermore, the application of Bayes Theorem is done with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. [5]

$P(A|B) = \frac{P(B|A)P(A)}{P(B))}$

Bayes Theorem is composed of conditional probabilities of an event A occurring given B is true multiplied by the conditional probability of an event A occurring divided by the probability of B occurring.

Given an input vector $\vec{x}$ the goal of Naïve Bayes is to predict an output label y by finding a value of y that maximizes the probability of y occurring given $\vec{x}$ is true. Thus, the Naïve Bayes Classifier is:

$P(x|y) = \prod_{i=1}^{n} P(x_i|y)$

This formula allows us to estimate the conditional probability of x_i occurring given y_j is true. The maximal probability will ultimately be chosen and will classify the new instance to y_j.

One major flaw of Naïve Bayes is in the case that one of the conditional probabilities is zero, then this dominates the whole expression turning it to zero as well. In situations like this it is possible to work around this issue by using other estimations of conditional probabilities, such as Laplace probability estimation.

5. Experiment

Google's Colaboratory will be used to conduct the experiment. Colaboratory was selected because its hassle-free deployment and the universal nature of the software simplifies submitting deliverables. Furthermore, the Scikit-Learn library will be utilized for its easy-to-use and well-documented code base.

In its original state, the attributes of this data set were denoted by characters approximately representative of the word they described. This was problematic for feeding the data into Scikit-Learn's algorithms as Decision Tree and Naïve Bayes implementation required the data to be numerical. Pre-processing of this data consisted of replacing all 5,644 data point's twenty two attributes with a numerical categorical representation starting at zero and discretely incremented to encompass all of the characteristics of a given attribute. As aforementioned, 2,480 data points were missing an attribute and needed to be removed from the data set. I wrote a program in C++ to accomplish both tasks; to remove these data points and to process a character string into a string of numbers.

This newly processed data was all ready for experimentation. Beginning with Decision Trees, a script was set up in Google Colab. The data was uploaded and checked by internal functions to ensure consistency and proper dimensionality. The class portion of the data was separated into its own variable. After numerous iterations, a test size of 40% was determined to be the ideal percentage. Next experimentation with the maximum amount of leaf nodes found ten to be an ideal amount. The cross validation score was called upon the decision tree with the training subset of data as parameters along with specifying a cross validation of ten for optimal results. Using scikit-learn's internal function "DecisionTreeClassifier", a decision tree object could be generated with a maximum of 10 leaf nodes, the

training subset of data could be fitted into a decision tree and now prediction data could be extracted.

Utilizing scikit-learn's internal "CategoricalNB" function, an object could be instantiated to begin processing the data according to Naïve Bayes methods. In this case, a test size of 25% was experimentally determined to be most ideal. A mean cross validation score was calculated from the CategoricalNB object along with the training subset of data with a cross validation of ten selected. The CategoricalNB object was fitted and prediction of the test data could commence.

In both experiments the following scores were calculated to measure the performance of these two algorithms. The test error was calculated with the mean squared error function with y test data and y prediction data as parameters. The test accuracy was calculated with the accuracy score function with the same parameters. For Naïve Bayes function, there exists a score function which is equivalent to the test accuracy, and is used instead for that experiment. The training error was calculated by predicting a subset of data from the training instance vector and promptly was fed into the mean squared error function along with the y training data subset. The training accuracy was calculated with the above accuracy score. Finally, a confusion matrix was generated from the test classes and predicted classes.

6. Evaluation

The Decision Tree machine learning algorithm was the first experiment. To better understand the flow of the data in this algorithm, many parameters had to tested. The first parameter to be experimented with was the test size. Intuitively, this played a role in the fine balancing act of underfitting and overfitting as this parameter would determine how much of the data would act as training data and how much would act as test data. I ran the algorithm over the range of 35% to 85%, recorded the results in a spreadsheet and plotted the test accuracy against the training accuracy. These two data points had unexpected behavior in the middle of the range with one being greater than the other then inverting, however the highest possible score of both as well as the smallest difference between these two scores was ultimately selected to avoid overfitting or underfitting patterns. Once the test size was determined to be 40%, experimentation continued with maximum leaf node size. As anticipated, the accuracies drastically grew proportionally as the maximum leaf nodes increased from 2 to 10. I did not expect to calculate a 100% test accuracy and 100% training accuracy with 10 maximum nodes. At 8 maximum leaf nodes, the decision tree had a test accuracy and training accuracy of 99.5% and 99.6% respectively.

Implementing Naïve Bayes Classifier began with understanding the trends of the data set. Although the CategoricalNB is much simpler than DecisionTreeClassifier in the possible parameters as well as what its return value is, the test size was the main focus of this experiment. Various trials were conducted on the range of test sizes from 10% to 75%. Any test size above 75% resulted in errors from the script. Surprisingly, the accuracies topped out at around 20% before falling significantly as the test size grew. At 20% test size, both the score and training accuracy were the highest at 98.1% and 98.1% respectively, with a difference of only 0.874%. These were the best values of the set. It should be noted that even at the ranges with poorer values, such as at 75% test size, the score and accuracy were still above 94.5%.

Comparatively, these two machine learning algorithms both achieved high scores. Decision Tree Classification accomplishing a perfect score seems a little too perfect. I could not determine if there was an error in the experiment, or if this is something that is legitimately achievable. Although, I want to intuitively say that the data did not have conflicting classifications. Therefore, the decision tree could excel. CategoricalNB performed well with a test error of 1.86% and a training error of 2.08%. Achieving a test accuracy of 98.1% and a training accuracy of 97.9% is no small feat.

7. Conclusion

In conclusion, both Decision Tree Classification and Naïve Bayes Classifier performed better than I expected. I am inclined to believe the experiment was performed well, although the missing data that had to be removed did alter the balance of the classification percentage. I believe discrepancies will likely be derived from this aspect. However, I believe both supervised machine learning algorithms demonstrated it was possible to determine poisonousness based off of phenotypic traits in mushrooms. Possible applications of this data set could be integrating cameras to filter through mushrooms in an industrial setting where various unknown mushrooms might be processed. Another possible application could be to assist mycologists in the discovery of new mushrooms poisonous characteristics based off our current knowledge and the discrepancies between actual poisonousness and machine learning would reveal the ways we could improve the field.

Acknowledgments

A special thanks to the University of California, Irvine's Machine Learning Repository for the data set. I would like to thank Dr. Zhang for a deeply informative and rich learning experience in the field of Machine Learning.

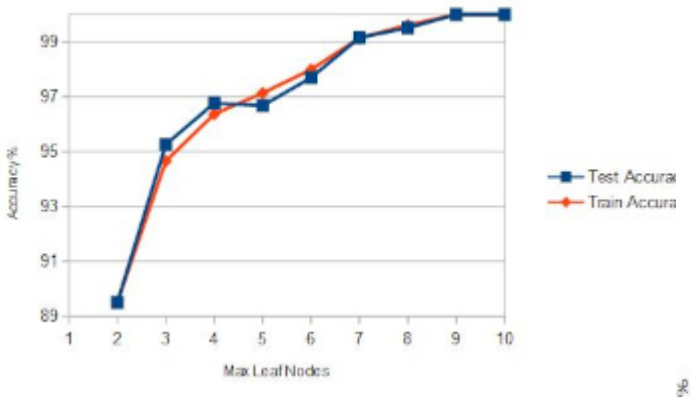

Figure 1. DT: Accuracy vs Max Leaf Nodes

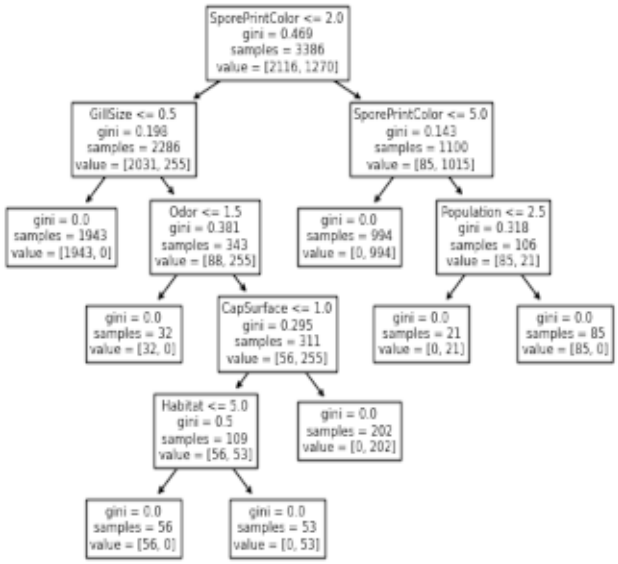

Figure 2. Decision Tree

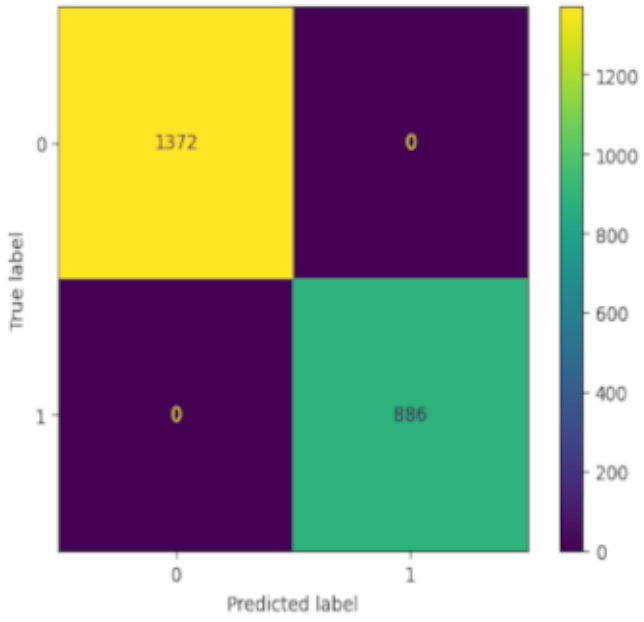

Figure 3. DT: Confusion Matrix

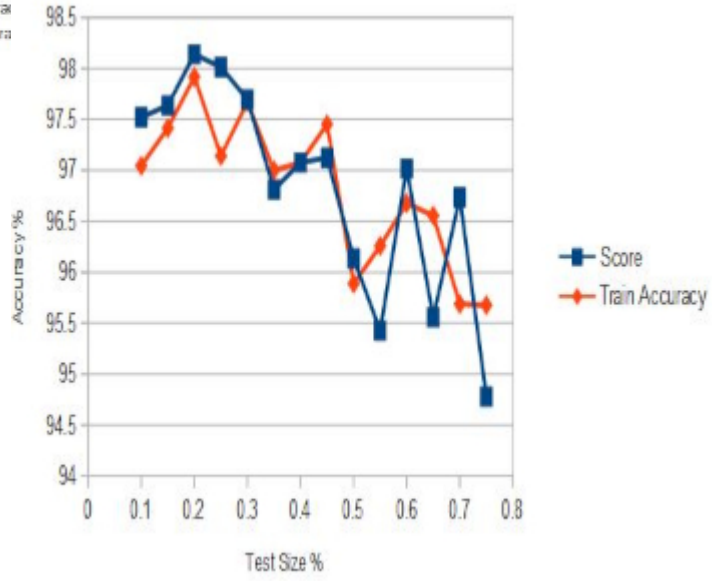

Figure 4. NB: Accuracy vs Test Size

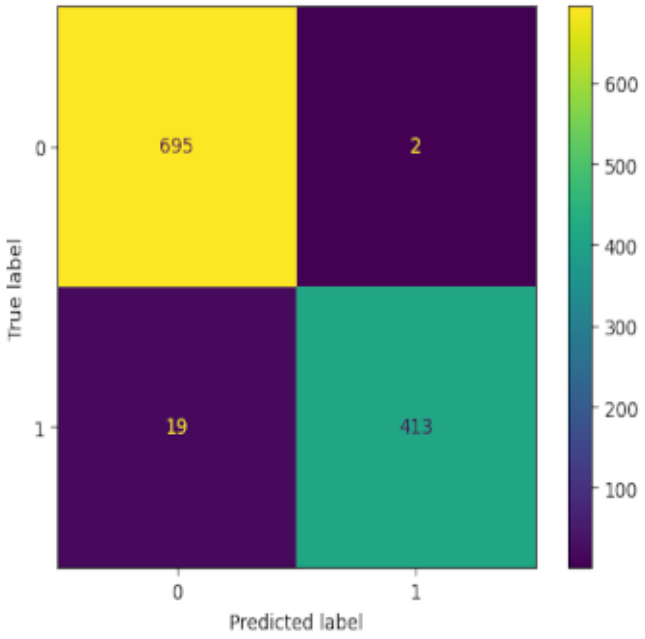

Figure 5. NB: Confusion Matrix

References

[1] Mitchell TM, et al. Machine learning, volume 1. McGraw-hill New York, 2007.

[2] URL scikit-learn.org/stable/modules/tree.html.

[3] Shrivastav A. Almost everything you need to know about decision trees (with code), Apr 2020. URL https://towardsdatascience.com/almost-everything-you

[4] URL https://www.ibm.com/topics/naive-bayes.

[5] URL scikit-learn.org/stable/modules/naive_bayes.html.