Statistical Learning Project II:

Classification of Mushrooms as Edible or Poisonous by Odor

Rachael Hawthorne

8/3/2020

Introduction

Dataset

The dataset I am using for this project was obtained from the UIC Machine Learning Repository and contains 8124 hypothetical mushroom samples from 23 species of mushrooms in the *Agaricus* and *Lepiota* Family. Each mushroom sample is described with 22 physical characteristics and belongs to either class poisonous or edible. The features used to describe the mushroom samples are: cap shape, cap surface, cap color, presence of bruises, odor, gill attachment, gill spacing, gill size, gill color, stalk shape, stalk root, stalk surface above the ring, stalk surface below the ring, stalk color above the ring, stalk color below the ring, veil type, veil color, ring number, ring type, spore print color, population, and habitat. Each of these features also include 2 or more levels as described at https://archive.ics.uci.edu/ml/datasets/Mushroom.

Research Question

Using this dataset along with random forest, PCA and k-means clustering, and SVM models, I hope to show that odor can be accurately used to classify a mushroom as poisonous or edible on its own. I also hope to find any other easily recognizable attributes that can be used to further improve the accuracy of classifying mushrooms in the field. The importance of this question lies in there not being a commonly known feature that can be used to identify mushrooms that are safe to eat. Finally, I am curious to know which method will perform the best on an entirely categorical data set. The reason for this being that I have found that many of the methods I have tried to use in the past are intolerant of categorical data.

Methods

Preprocessing

Preprocessing was performed in the same manner as my previous project and is restated below:

"*The dataset indicates that there are missing values in the stalk-root feature due to some mushrooms not having a stalk attached. To deal with this, I first searched the dataset for the missing values to see how many of them there are. Upon searching, I found that there were 2,480 samples where the stalk root feature is listed as missing. Because there were so many rows with missing values in that feature, I decided to remove the entire feature instead of removing the rows, as removing the rows would shrink the dataset too much. I do not believe that removing this feature will affect the outcome of any of the classification techniques simply because it is apparent that there is a good probability that when finding a mushroom in the field, it will have a missing stalk root.*

When looking at the data, I also found a problem with another feature: veil type. Unlike all of the other features every mushroom sample a partial veil type, making this feature completely redundant for classification. To deal with this issue, I removed the entire veil type feature.

Following the removal of the previously stated features. I converted each feature to a factor for the classification techniques to know that they are to be treated as categorical variables. Any other data processing done is specific to the classification technique and will be discussed under the appropriate subheading. After the preprocessing was completed, the variables and their levels were as follows:"

```
'data.frame':   8124 obs. of  21 variables:
 $ class                    : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
 $ cap.shape                : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
 $ cap.surface              : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
 $ cap.color                : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
 $ bruises                  : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
 $ odor                     : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
 $ gill.attachment          : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
 $ gill.spacing             : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
 $ gill.size                : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
 $ gill.color               : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
 $ stalk.shape              : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
 $ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ stalk.color.above.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ stalk.color.below.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ veil.color               : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
 $ ring.number              : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
 $ ring.type                : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
 $ spore.print.color        : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
 $ population               : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
 $ habitat                  : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

Random Forest Decision Trees

Algorithm. Random forests are a type of decision tree ensemble method used for classification and regression. For this algorithm, a specified number of decision trees are created using bagging, a technique that creates multiple training sets with resampling in order to reduce variance and overfitting. For each tree created, the algorithm uses a formula (Gini index, entropy, etc.) to determine the best attribute to split the data and the best split point within that attribute. This continues for each node created until a stopping condition is met (usually either determined by node purity or size). Once all trees are created, each sample in the test set is run through all trees and the results are averaged to obtain a final classification.

Implementation. In my implementation of a random forest for my data set, the data set was first split into training and testing sets with the training set containing 75% of the samples and the testing set containing 25% of the samples. I then created multiple random forests with differing numbers of trees to determine how many trees would be optimal to include. Afterwards I decided to use n = 500 for my number of trees and included each of my predictors in the tree construction in order to help determine the most important variable(s) for determining edibility of a mushroom. The random forest was then cross validated with the test set and the obtained results were put into a confusion matrix. I also created a variable importance plot to obtain information on which variables were more most important for classification.

PCA & Clustering

Algorithm. Firstly, PCA is a dimensionality reduction technique intended for use in highly dimensional data sets in order to reduce the dimensionality without losing important information from the data set. Second, the clustering algorithm I used was k-means clustering. K-means is a

clustering technique that plots all points in a data set and then assigns all points to a number of randomly selected mean points according to their distance from the mean points. After points are assigned, means are then recalculated, and points are reassigned to the appropriate cluster. This continues until a stopping condition is met (usually when there is no change in the mean.)

Implementation. For my implementation I decided to perform PCA on my data set and then use the first two principal components to run k-means clustering. From reading online, I found mixed opinions on whether PCA can be meaningfully used for categorical variables, with the consensus seeming to lean towards no, but I decided to try it anyway. To perform PCA on my data set, I first had to dummy code all of my variables before running the algorithm. I used a scree plot to determine which principal components I would keep in my analysis and separated PC1 and PC2 into their own data frame to use for the k-means algorithm. I chose the number of desired clusters to be 2 because the data set has 2 classes, edible and poisonous. I then plotted the k-means results as well as a plot showing the actual class separation. Additionally, I created another plot colored to show the odor of each point to see the separation of those as well.

Support Vector Machine

Algorithm. A support vector machine is a machine learning algorithm that uses a kernel system to transform data and find an optimal boundary to separate your data. SVMs can be used for classification and regression and can either be linear or non-linear. Linear SVMs, as the name implies, attempts to find the best linear boundary between your data while non-linear SVMs do not have that restriction and can form any shape of boundary.

Implementation. To start, I first separated the features I wanted to include in the analysis (odor and spore print color) into a separate data frame along with the class column and proceeded to dummy code the features. Then I split the data into training and testing sets as I did for my decision tree. I opted for using a radial SVM because it seemed to perform the best when tested against the other options. For my model, I also wanted to test different levels of cost and gamma, so I ran multiple models with cost as 1, 5, 10, and 100, and gamma as 1, 2, 3, and 4. Then I used the best model to proceed with prediction and gathered the results into a confusion matrix.

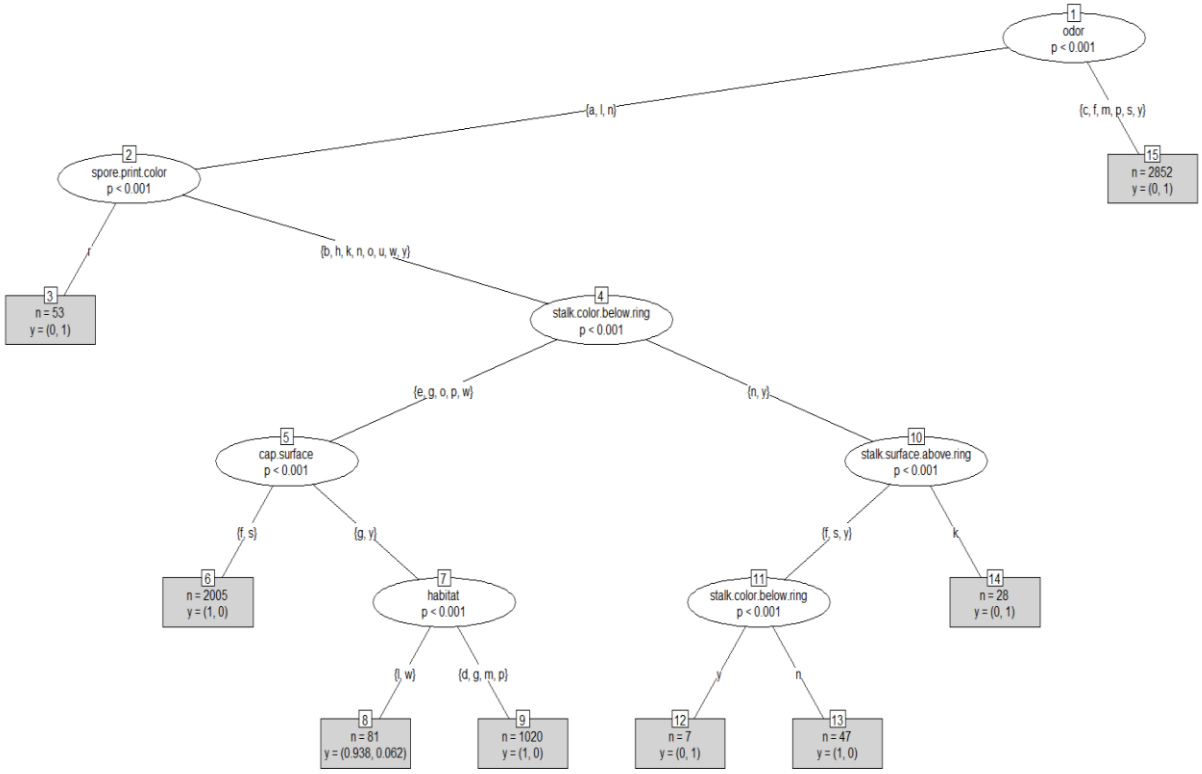
Results

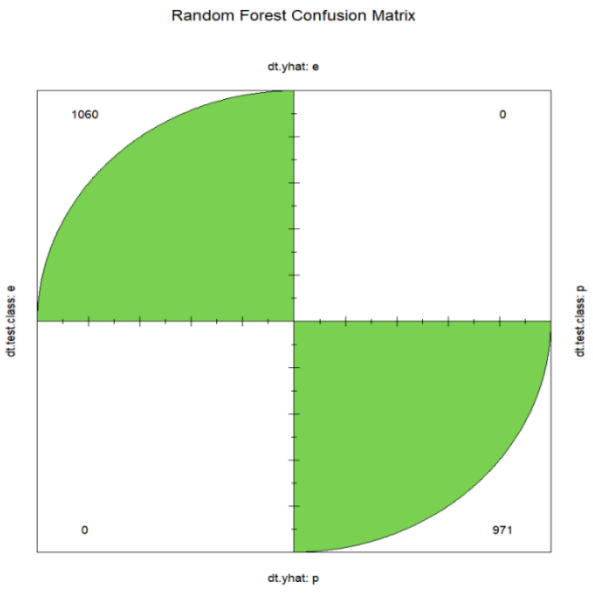
Random Forest Decision Trees

Sample Tree. Below is a sample of what one of the decision trees in the random forest could look like. This sample tree was created using the training set as opposed to using the entire data set. It is important to keep in mind that in this visualization, y(1,0) is synonymous with the class “e” and y(0,1) is synonymous with the class “p”.

Right off the bat it can be seen that there are 2852 out of 6093 mushroom samples in the training set that can be immediately classified as poisonous from odor alone. Meaning the odors that are indicative of a poisonous mushroom are creosote, fishy, foul, musty, pungent, and spicy. Unfortunately for our research question, using odor to identify an edible mushroom in the field is risky. Out of the 3241 samples that have almond, anise, or no odor, there are 88 samples that are still poisonous. Meaning that it would be unlikely that you would consume a poisonous mushroom based on odor alone, but most people would not take that chance. Fortunately, there

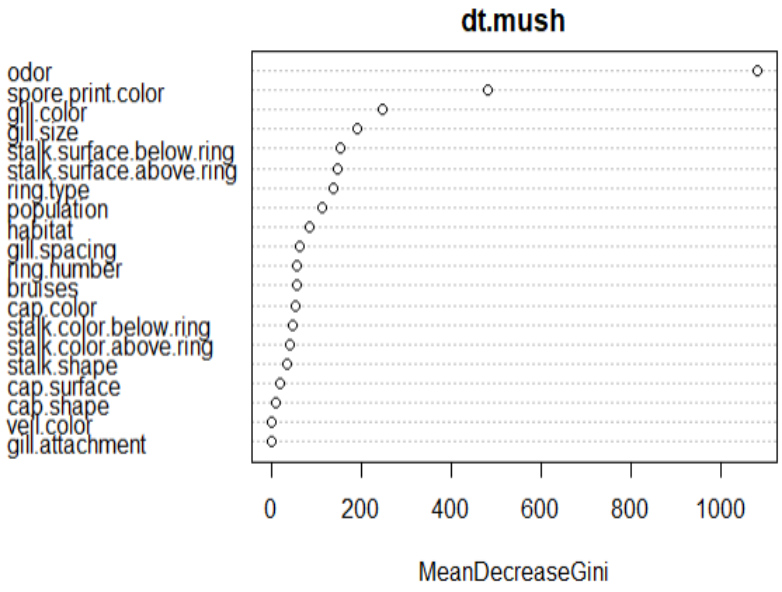
are other features indicated that can aid with spotting a poisonous mushroom. For example: a green(r) spore print color, a silky(k) above ring stalk surface, and a scaly(y) below ring stalk surface.





Confusion Matrix.

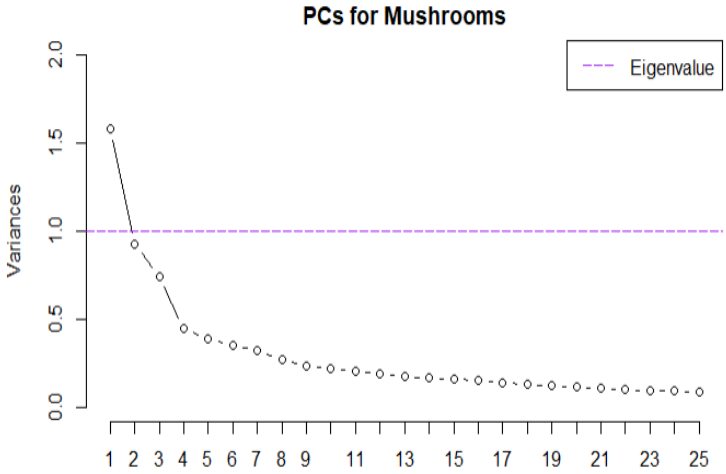
When cross validating my random forest model on the test set. I found it surprising that the model achieved 100% accuracy. So surprising, in fact, that I spent hours staring at my code trying to find what I did wrong. After being unable to find any errors in my code, I used the decision tree algorithm that I used to create the visualization above to see if there were any differences. To my surprise, the single decision tree was able to classify the test set data with 99.8% accuracy. So it is reasonable to think that a random forest of 500 trees is able to achieve 100% accuracy.

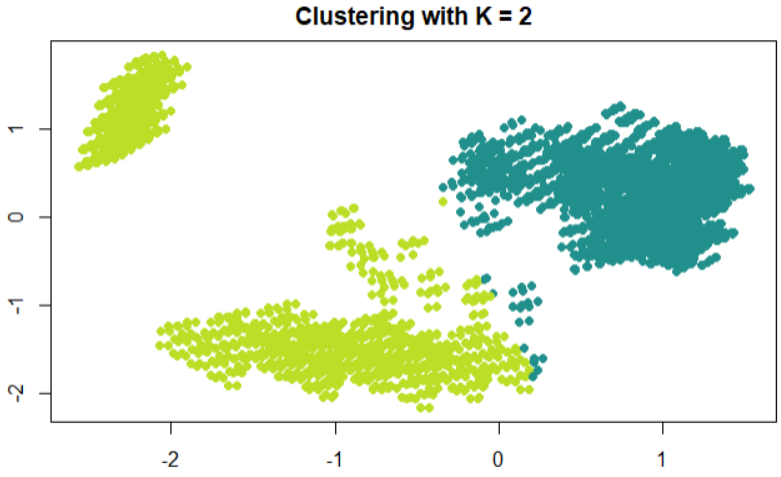


Variable Importance. To the left we see the variable importance graph. As expected, odor blows the rest of the features out of the water in terms of importance. Because odor is so prevalent, I conducted the rest of the models in this project using both odor and spore print color. I chose to include spore print color because it also has a decent level of importance.

PCA & Clustering

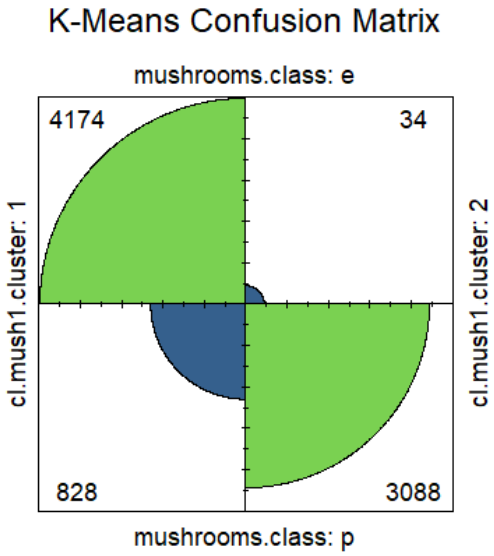
PCA. The scree plot that I obtained from my PCA suggests that the first principal component explains most of the variance of the data set. Because I want to keep 2 features to use for k-means, I decided to use the 2nd principal component as well. As mentioned before, I'm not sure how meaningful these principal components actually are due to my data set being all categorical variables.





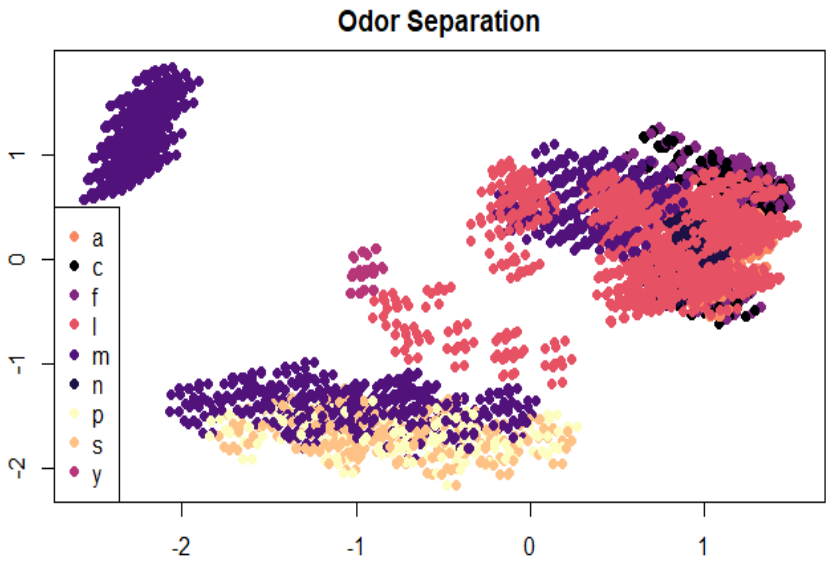
K-means. After performing k-means on the two selected principal components, I plotted the results (left) and colored the points according to which cluster they were assigned to. The light green cluster, which represents class "e" consists of 5002 data points while the dark green cluster, which represents class "p", consists of 3122 data points. In the next figure, you can see the actual class

separation of the data points. From this we can gather that k-means cannot perfectly find clusters from the first two PCs. The figures surprisingly align what seems like inversely with my random forest model, suggesting that there are many samples that are easily clustered as edible by the first two PCs but clustering poisonous samples is not as clear.



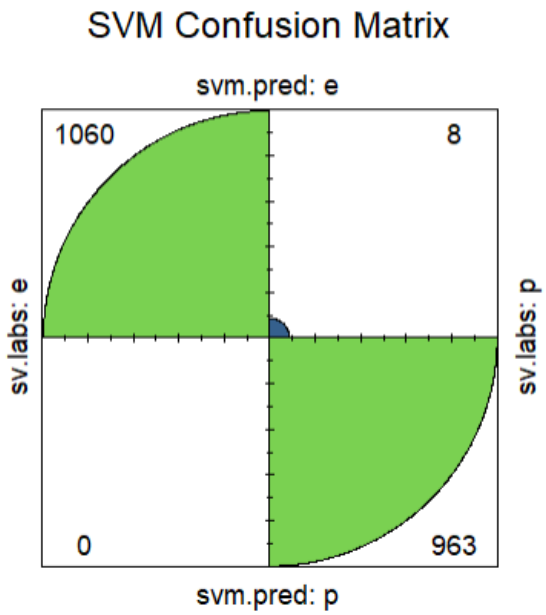
Confusion Matrix. The confusion matrix on the left indicates that k-means had more false negatives than false positives and overall had an easier time clustering edible mushrooms than poisonous mushrooms. The overall accuracy of this model was 89.4%, which is good only if we assume that analysis is meaningful. I am on the fence in whether I believe that this worked correctly. I lean more towards saying that it did not work as it was intended to due to some weird inconsistencies that I noticed. One due to the expectation that I had that poisonous mushrooms could be clustered more easily because of the evidence from my decision tree that many odors are directly indicative to poisonous mushrooms.

Odor plot. Next, I was curious to see whether coloring the point by odor would give similar results to my decision tree. Remembering from before, the odors that were said to be completely indicative to poisonous mushrooms are creosote(c), fishy(y), foul(f), musty(m), pungent(p), and spicy(s). We can see from the plot that all of the points for these 6 odors are clustered in the top left and bottom left/middle clusters. According to the clusters above, that cluster is classified as edible, which is the exact opposite of what my random forest model indicated.



K-Modes. Before moving on, I wanted to touch on one last thing that I attempted for clustering called k-modes clustering. The k-modes clustering algorithm is apparently specifically used for clustering categorical data by using modes instead of means. Because it wasn't included as one of our methods in class, I decided not to use it for a full analysis. Though, I did quickly run it to compare the results to see if it would work better than PCA and k-means and it surprisingly did substantially worse. It clustered 7324 points in the first cluster and 800 points in the second cluster, causing it to have a 52.5% accuracy.

Support Vector Machine



Confusion Matrix. Unfortunately, this confusion matrix is the only figure I can show for my support vector machine, the reason being is that while I learned that SVMs can be used for categorical data, it is impossible to plot an SVM created from categorical data. These results are from an SVM created using the odor feature and the spore print color feature, as before. The best SVM model that I used was a c-classification SVM that used the radial kernel and a cost of 1. It created 109 support vectors for my data and had an accuracy of 99.6%, which rivals that of the random forest model. Overall for this model, it is hard for me to interpret it when I can't visualize it. Unlike my k-means clustering, the only incorrectly classified samples were 8 poisonous samples that were predicted as edible samples.

Discussion

Best Performance Model

Out of the three methods used for this analysis, I wholeheartedly believe that the random forest/decision tree model performed better and is better suited for my data set. My random forest model was able to obtain 100% accuracy on my data set and while the SVM obtained similar results with an accuracy of 99.6%, it has the disadvantage of not being able to visualize categorical data. As for the PCA and k-means clustering model, even though it has an accuracy of 89.4%, I cannot trust the results due to the general consensus that PCA and k-means are not designed to be used with categorical data. Overall I have been fairly skeptical about how well the models performed on my data set because you don't usually see such high accuracy. But I have attributed this to the data set itself and not to some sort of error. I feel that the random forest model should be used for publication well before the other two.

Other Methods

Of all of the methods I have used so far for this data set, the only other method that could rival the random forest model is the KNN model from my Summer I project. Even still, I stand by the random forest model as being the best suited for this data set. In my previous report, the KNN model was very accurate in classifying the data but failed to provide any information on which features were more important for making that determination. Decision trees fix that issue due to the ability to clearly show which features were chosen and how they were split.

Research Question

In regard to the research questions proposed at the beginning of this report, it has been found that odor plays a large part in determining if a mushroom is safe to eat. Though it doesn't play a large enough part to safely base edibility on odor alone. It was found that spore print color can help improve the accuracy when used with odor to help determine edibility, but spore print color is not a feature that is easily discernable in the wild. As for the best model to be used with categorical data sets: decision trees or decision tree ensemble methods.

References

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

Max Kuhn (2020). caret: Classification and Regression Training. R package version 6.0-86. https://CRAN.R-project.org/package=caret

David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2019). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-3. https://CRAN.R-project.org/package=e1071

Revelle, W. (2019) psych: Procedures for Personality and Psychological Research, Northwestern University, Evanston, Illinois, USA, https://CRAN.R-project.org/package=psych Version = 1.9.12.

Simon Garnier (2018). viridis: Default Color Maps from 'matplotlib'. R package version 0.5.1. https://CRAN.R-project.org/package=viridis

Weihs, C., Ligges, U., Luebke, K. and Raabe, N. (2005). klaR Analyzing German Business Cycles. In Baier, D., Decker, R. and Schmidt-Thieme, L. (eds.). Data Analysis and Decision Support, 335-343, Springer-Verlag, Berlin.

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2020). dplyr: A Grammar of Data Manipulation. R package version 1.0.0. https://CRAN.R-project.org/package=dplyr

Torsten Hothorn, Kurt Hornik and Achim Zeileis (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15(3), 651--674.