John Takakjian
CS 82B, Spring 2024
Professor Seno

Final Project Summary Report

Introduction

The dataset that I am using is originally from the UCI Machine Learning Repository. I scanned through some of their more popular datasets and found this one the most interesting and novel. This is the mushroom dataset. There are several different characteristics that we have of mushrooms such as their cap color, cap size, their smell, etc as well as the most important factor, whether they are poisonous (p) or edible (e). When all was said and done, I was able to create a machine learning model that predicted the test set perfectly. The final details of the different scoring metrics used will be shown later in the report but this was quite surprising to me. Most of the models that we created in this course and that we saw in the book did well, but none so very well. And we were regularly warned about hitting performance metrics too well and the issues that are likely present when that happens. Despite that, I was able to confirm that this is a result that others have achieved regularly online as well and so with that knowledge felt it likely that there was, in fact, no issue with data leakage or anything of the sort.

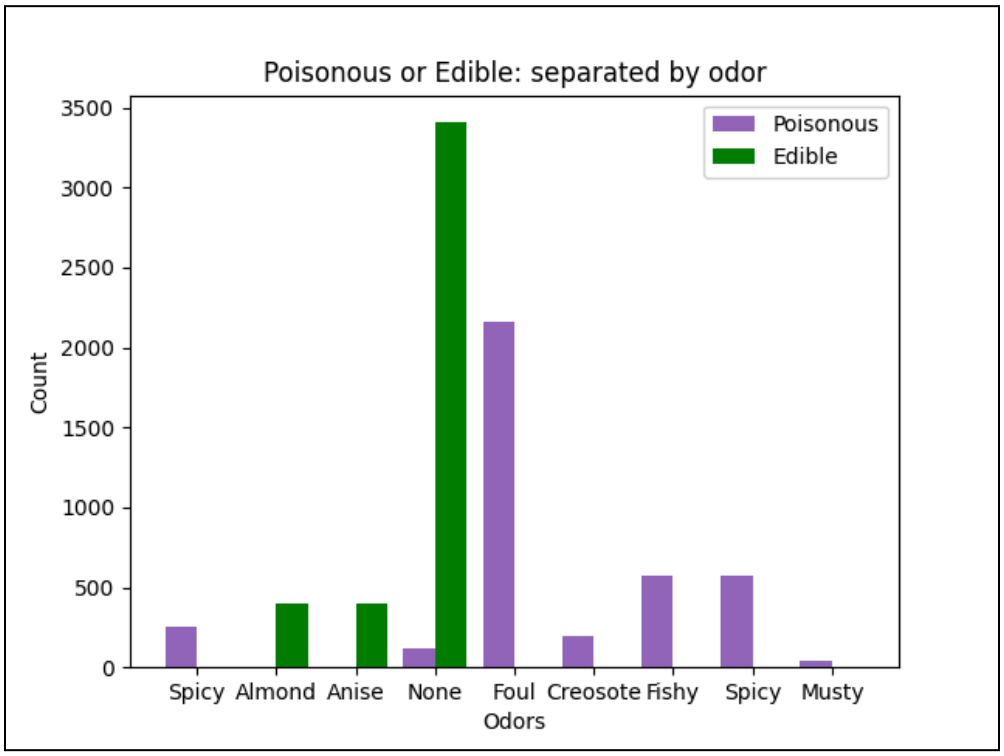
Data Preprocessing and Cleaning

This dataset downloads slightly differently than others that we have used in the past. I had to do some additional digging into the documentation of Pandas to figure out how to read the csv that UCI gave me. It ended up being fairly easy, despite it requiring the slightly tedious step of typing out all of the column names individually into an array. Once I had identified the correct file and loaded the dataset up with the correct column names and being clear that there was no header on the csv, it was time to get started.

At first I wanted to explore the dataset with df.info() as that is a good way in my experience to get a preliminary view of what you are working with and what kind of issue lay ahead. I was deceived into thinking that there was very little to be done initially. All of the columns were of the same type, objects, and there were no missing values at all. While at first, I considered moving to

another dataset that would allow me to do some imputation of values, I decided to press on and see what there was to do here.

Then I decided to try to look at a couple of variables and their relative values of poisonous versus edible mushrooms. Much to my surprise, the first variable that I tried, Odor, ended up being a very very good proxy for poisonous versus edible. This helps to explain why our machine learning model was able to perform so well. See the figure created below:



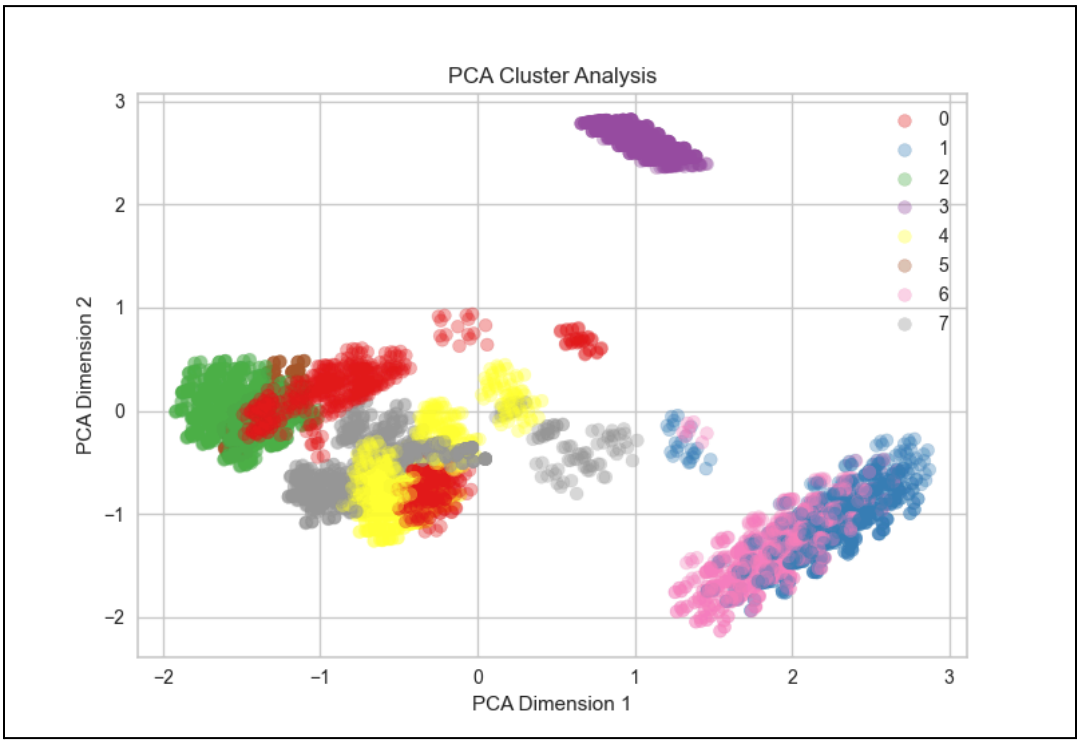
From there I created my training and test sets. I then needed to convert the data in those sets as Sci-Kit Learn likes numeric data rather than string data. Because all of the data in this set is categorical, using the one hot encoder was what I chose to go with for encoding the data.

Data Analysis

The Data Analysis and Machine Learning sections of this report overlap a bit. I chose to use the KMeans clustering algorithm from sci-kit learn to do some basic analysis in hopes of maybe finding some information about species. I trained the dataset and then both made my own elbow graph and chose to use the k elbow visualizer from yellow brick to test out using the different methods. I wouldn't say that I have a strong preference for one over the other at this point but it is

slightly easier and I do get more information from yellow brick's method so in the future I'll likely use that.

Finally, I chose to do some PCA to try to visualize our clusters on a graph. Here are our clusters:



The data in these clusters is pretty defined. There is definitely some overlap but others have their own area. It would likely be much clearer if we could view these in further dimensions but as we can't we have to assume that we've probably found some data related to species or genus.

Machine Learning

Now onto the model that I trained. As this was a classification problem, I chose to use the K Neighbors Classifier from sci-kit learn. From there, I looked into some of the hyperparameter options for this classifier, made a dictionary to run through, and then brought in grid search cv. After running through all of those options, I instantiated the best classifier, and scored it's predictions against the actual label values. The result achieved was 1.0

Conclusion

The default scoring method for this classifier is mean accuracy. I chose to also bring in precision, recall, and their combination, F1 score. The results of these were the same as mean accuracy, 1.0 meaning the model performed perfectly.