Classification of Mushrooms from the Agaricus and Lepiota Family

Aaron A. Gauthier and Pedro Uria Rodriguez

George Washington University

Classification of Mushrooms from the Agaricus and Lepiota Family

Introduction

Problem. Various guides clearly state that there is no simple rule for determining the edibility of a mushroom. Normally, one needs to identify the name of the particular mushroom to be able to tell if it is edible or not. However, this task is very difficult and time consuming for non-experts, so we thought that a machine learning approach on mushroom data could shed light on some of the features that almost guarantee a mushroom will be poisonous. This way, novices can avoid wasting time trying to identify a mushroom that will most likely end up being poisonous, and focus on mushrooms that will most likely be edible instead.

We are going to use a dataset from the UCI Machine Learning Repository, which contains 8123 mushrooms records about 22 features, classified as poisonous or edible, drawn from the *The Audubon Society Field Guide to North American Mushrooms*, for two genera (*Lepiota* and *Agaricus*) from the *Agaricaceae* family. We hope to build various machine learning algorithms that can give perfect predictions, or at least no false positives (identifying poisonous mushrooms as edible), and also identify the top features for better interpretability and generalization to other families of mushrooms.

Motivation. We realized that many people in the USA do not know the difference between an edible mushroom and a poisonous one. We hope that through this project we can help educate people on the top significant features that determine whether a mushroom is safe for ingestion or not. We hope this will help save lives and avoid tragedy, especially with children who are curious.

Domain Knowledge. While reading about mushrooms and our features, it became clear that some of them were very hard to classify. Thus, we decided to divide the project into two parts. In the first part we implemented the Machine Learning models using all the features, while on the second part we only used the features that in our opinion are easy to identify.

Proposed Methods

We are going to implement different machine learning models using *sklearn* and train them on the mushroom data in order to achieve the maximum possible accuracy on the testing data. The preprocessing is the same for all the models, and consists on getting the X (predictor vector) and y (target vector) *NumPy* arrays from our *pandas* DataFrame. We also one-hot-encode X and label encode y. Then, we make a 70% training and 30% testing split. The standardization step was omitted because all of our data is categorical, so we will only have *dummy* variables with 0s and 1s. Thus, standardizing is not necessary and would only worsen the Decision Tree's interpretability.

Logistic Regression. Due to the shear size of our dataset, we believe that Logistic Regression will provide a very accurate model, because it is a natural fit for this type of binary classification supervised learning problem. This type of classifier simply predicts the probability of a mushroom being edible or poisonous given the features we input. This probability will then be used by the model to classify the mushroom. Logistic Regression will also provide us with a way of ranking the features based on their importance.

Decision Tree. By building a Decision Tree, we will be able to provide people with a simple visual guide to tell if a mushroom is edible or not. Decision trees are human readable rules in the form of graphs that resemble a "tree-like" structure, which includes decision nodes, leaf nodes, and branches providing a clear decision path. Note however, that this tree will only be accurate for the *Lepiota* and *Agaricus* genera.

Random Forest. A Random Forest is an ensemble of various decision trees, which apart from being a very powerful classifier, provides with a numerical measure for the importance of each feature in distinguishing an edible mushroom from a poisonous one. This importance is basically the total decrease in node impurity at the node corresponding to the feature, weighted by the probability of reaching such node, and averaged across all of the trees in the forest.

Support Vector Machines. The complexity of this method is very large for non-linear kernels, because it basically transforms the feature space into higher dimensions until the problem becomes linearly separable. It will be interesting to see its performance compared to the other classifiers.

K-Nearest Neighbors. KNN is a method very different compared to the previous ones in the sense that it does not learn a discriminative function from the training data, but memorizes the training dataset instead. It uses a majority vote approach to classify a data point based on its k-closest data points.

Naive Bayes. As its name suggests, Naive Bayes is based on Bayes' Theorem. The advantage of using this model is its low time complexity (linear). It also shines on small datasets, although this is not our case.

K-Means Clustering. Finally, we will use a clustering approach by removing all the labels from the training data minus two unique ones, fitting the model on this data and using a map to assign the clusters indexes to the original labels, and finally compare with the predicted testing labels by using a combination of the clustering and this map. We already know $k = 2$, so there will be no need for the elbow method or other such approaches for determining the best value for k. We do not expect this unsupervised method to work as well as the supervised ones, but it will be interesting nonetheless.

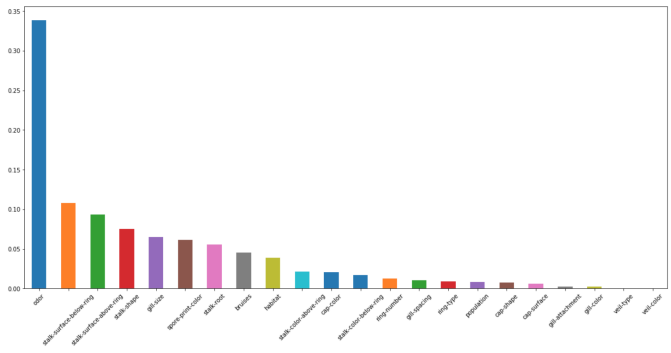
Experimental Results and Analysis

EDA. First of all, we need to get familiar with our dataset. We have 8123 rows, and each of one describes a mushroom by assigning values to different features, all of them categorical. The features consist on various characteristics of mushrooms, concerning odor, cap, gill, stalk, veil, ring, population and habitat. The main takeaway from the Exploratory Data Analysis is that we have 2840 missing values on the `stalk-root` column, and that 51.8% of the class labels are "edible", while 48.2% are "poisonous", so we have a very good balance between the two. We will take various approaches to deal with this missing data:

1. Use the dataset without the column `stalk-root`.
2. Use the raw dataset with "?" (UCI ML tag for missing values) as `np.NaN`.
3. Use the raw dataset (treating "?" as a possible value of `stalk-root`).
4. Use the dataset without rows with missing values (losing 30 % of our data).
5. Impute the missing values by mode imputation.

Using all the features. We got perfect cross-validated scores after hyperparameter tuning for Logistic Regression, Decision Tree and Random Forest for every treatment of the missing values! For the rest of the classifiers, we only used the dataset without `stalk-root`, as we found that it was not needed. Support Vector Machines and K-Nearest Neighbors also gave perfect accuracies, while Naive Bayes almost (≈ 0.999). Clustering also performed decently, giving back an accuracy ≈ 0.904. It is worth mentioning that tuning the hyperparameters for the Support Vector Machine took one whole hour!

We also used Logistic Regression and Random Forest to get the feature importances (only the ranking in the Logistic Regression case).

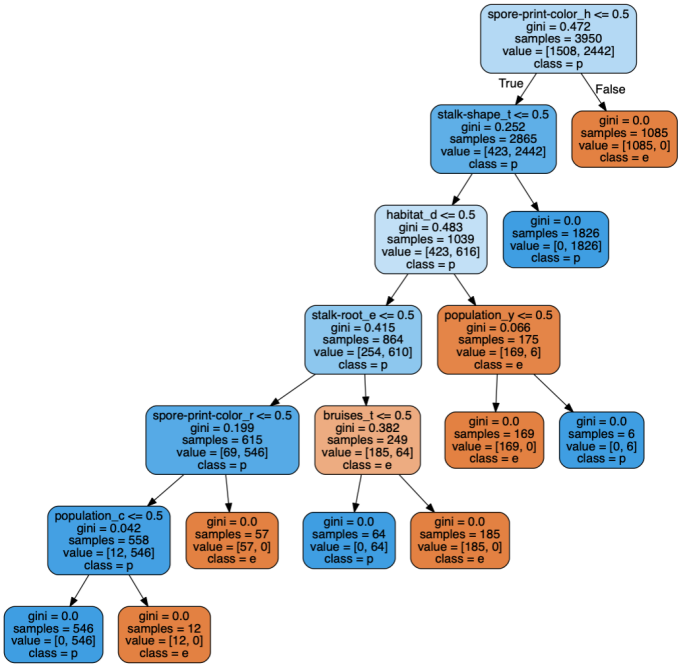


For Logistic Regression, only one feature (`odor`) out of our *hard-to-tell* features had a very relevant role (ranked first). It was also ranked first by Random Forest, but in this case stalk surfaces (above and below ring) were also relevant. The features importances computed by the Random Forest are showed on the right.

The Decision Tree was one of the most interesting models because it can give back rules to classify the mushrooms. That is, the mushroom hunters are not required to bring a phone with them and feed the features as input to a model running on it. Instead, a piece of paper with the decision tree printed on it

should be enough (given that they can identify the mushroom as one from the *Lepiota* or *Agaricus* genera). We saw that using all features with no missing values (dropping 30% of the data) was the approach that gave the simplest tree (17 nodes).

Using only the easy-to-tell features. Here things became a bit more interesting. In this case, without using *stalk-root* during Logistic Regression, we were able to drop the number of poisonous (negative) mushrooms identified as edible (positive), i.e, false positives, to only 1 out of 1175 poisonous mushrooms on the testing data using a grid search with cross-validation. This is because `stalk-root` turned out to be one of the most important features (ranked second) when we eliminated the *hard-to-tell* features.



For the rest of missing values approaches, we were able to get a perfect score once again. We could also clearly see the features moving up in the importance ranking in a natural way, filling up the places left by the *hard-to-tell* features. The Decision Tree proved to be very complex without `stalk-root` (89 nodes), and also gave 8 false positives. The rest of the approaches gave perfect scores and once again, the simplest tree was computed by using all the features and no missing values, being composed of 17 nodes (shown on the right).

The tree reads as follows: `feature_value` ≤ 0.5 means that the mushrooms do not have such value of such feature. Thus, the left branch leads to the part of the dataset with `feature` $\neq$ `value` ([False and True] = False) , and the right branch leads to `feature` = `value`. `gini` measures the gini impurity at the node. `samples` gives the total number of rows at the node, and `values` tell the number of samples with class [edible, poisonous], while `class` = most frequent class and the color is indicative of the percentage presence of each class.

For the rest of the classifiers, we used the approach 4. Random Forest performed perfectly and gave similar feature importances results as Logistic Regression for most of the features, although some of them did change a lot (*veil-color* was ranked third by Logistic Regression and last by Random Forest). Given the ways we computed the importances, we believe Random Forest to be more reliable (the *jupyter notebook* explains how we computed the importance ranking using Logistic Regression). Support Vector Machine

gave a perfect score and we also witnessed how the time complexity dropped significantly, because we were using less features (the effect is even greater because they were all converted to dummy features, as they are all categorical). K-Nearest Neighbors also performed perfectly, while Naive Bayes gave a lot of trouble (107 false positives out of 647 poisonous (negative) mushrooms). The accuracy of clustering also decreased to ≈ 0.855. One of the reasons Naive Bayes performed so poorly may have been because there is no way for us to fine tune the hyperparameters (or at least no way that we know of).

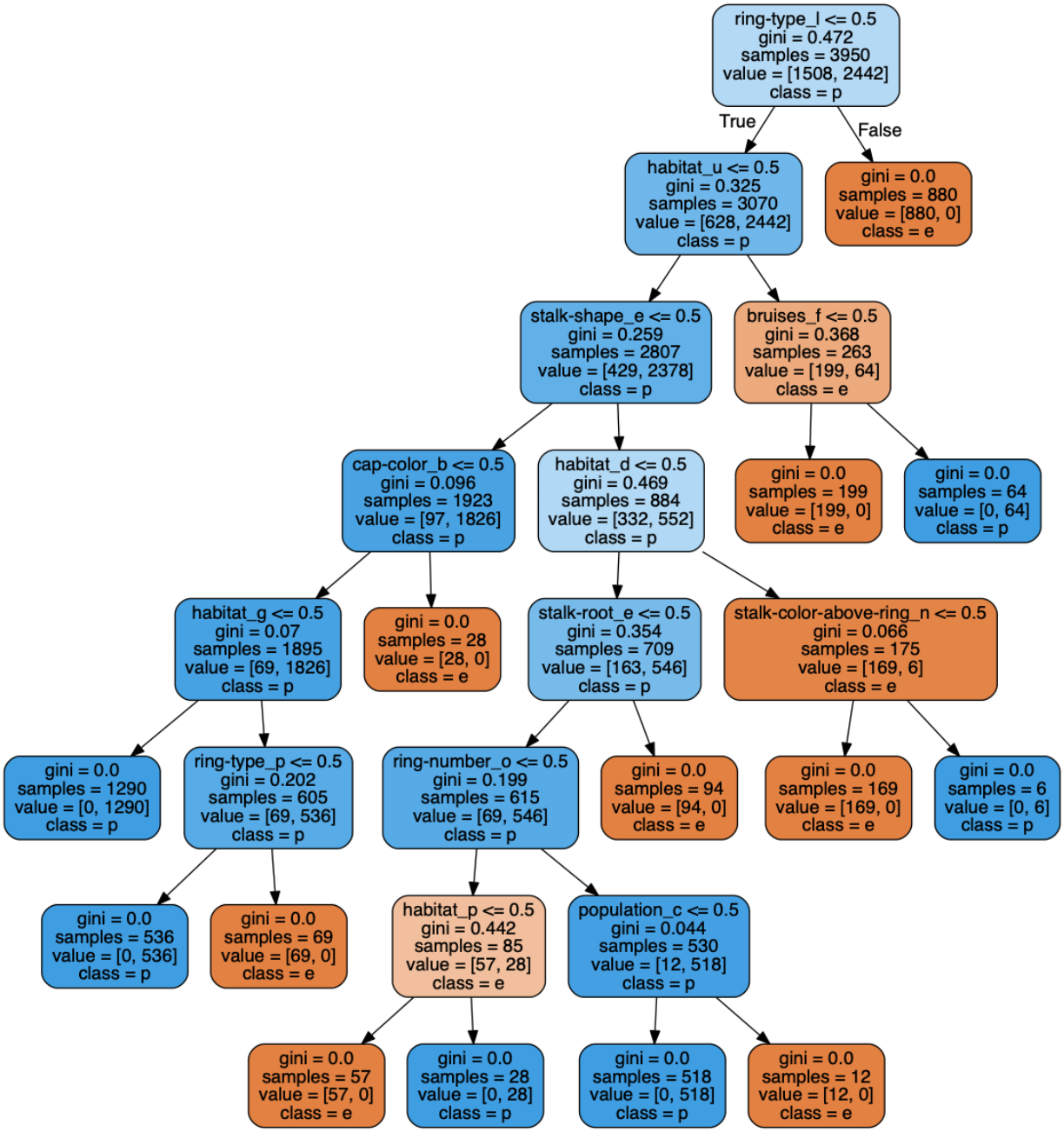
Finally, we computed the percentage of poisonous mushrooms for the top *easy-to-tell* features found by the Random Forest. We found that `spore-print-color` = chocolate, white and green are most likely poisonous, so mushroom hunters can stop wasting their time if they see this type of mushroom and move onto the next one. Well, actually, for this one this is not useful because to classify this feature the hunter needs to wait a whole night. If it does not bruise, the mushroom will also likely be poisonous. Same goes for large or no rings, for mushrooms living on leaves, cities and paths, and the stalk color above the ring being brown, buff, cinnamon or yellow. This could be done for the rest of the features, but this ones seem to be the most interesting.

Conclusions

Most of our learned models were implemented with surprising 100% cross-validated accuracy, even when using the *easy-to-tell* features. We were also able to identify the most relevant features for prediction. However, we must say once more, that these models are only valid for the two genera (*Agaricus* and *Lepiota*) the data came from. We also acknowledge that there is no substitute for domain expertise. Proof of this lies in some of the false positives that were classified in some of our models (refer to the *jupyter notebook* for more information). This is horrible when trying to classify edible versus poisonous mushrooms, because it can lead to someone being poisoned if they rely on the model. Even just one instance out of 1,175 or more is unacceptable. Therefore, a model in this instance should be a guide and not a substitute for domain expertise in classifying mushrooms.

Therefore, the models that have been derives from the *Agaricus* and *Lepiota* families are only accurate for these families of mushrooms. However, it is strongly encouraged that these models are only used to aid expertise (domain knowledge) of whether a mushroom is edible or poisonous. Expertise should almost always override data, especially when identifying potentially dangerous foods like mushrooms. Even though there is high confidence in our models, experts may have made mistakes in gathering the data, as they are human and prone to make mistakes. Thus, the authors take no responsibility on the usage of these models by other people, and always advise to consult with an expert before ingesting wild mushrooms.

On a last note, the handling of missing values proved to be quite interesting. When using all the features, dropping the problematic column `stlak-root` was fine, but when only using the *easy-to-tell* features, we did get some false positives when dropping `stlak-root`, but dropping the rows with missing values worked well, as we had a lot of data to begin with. Along the *jupyter notebook* there are lots of trees, but if we would have to pick only one to bring with us while mushroom hunting, this one would be it



References

[1] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning*. Packt, Birmingham, 2017.

[2] Machine Learning I's notes, slides, exercises and homework.

[3] Wikipedia's entries for the various classifiers.

[4] https://arxiv.org/pdf/1410.5329v3.pdf

[5] A bunch of other articles and coding questions, all referenced in the *jupyter notebook*.