Using Random Forests to Classify Edible and Poisonous Mushrooms

Mike Adams
Computer Science
University of New Mexico

Jack Ringer
Computer Science
University of New Mexico

***Abstract*—Although mushrooms can be tasty, they can also be quite dangerous. An issue faced by mushroom foragers is determining whether or not a given mushroom is poisonous or edible. Given that incorrectly classifying a poisonous mushroom as edible can lead to serious health consequences (up to and including death), an adequate procedure for classifying mushrooms should attain extremely high accuracy before being considered for real-world use. Within this report we investigate random forests for mushroom classification. Our experiments find that, after training, our random forest model is able to attain 100% accuracy on validation data and 100% accuracy on test data. The code used for this project can be found at the following GitHub repository: https://github.com/Jack-42/cs529-randomForest**

I. Introduction

The classification of wild mushrooms as either poisonous or edible is a serious problem for mushroom foragers. Within the United States accidental mushroom poisoning was responsible for more than 1,300 emergency room visits and more than 100 hospital visits from 2016-2018 [2]. Given the potential harm involved, a classification model should demonstrate extremely high reliability before being considered.

Within this work we investigate the applicability of random forests for mushroom classification. Random forests are an extension of decision trees where, rather than having a single tree learn from and classify data, an ensemble (or a forest, if you will) of decision trees are trained and then used to classify instances [4][5]. Each tree in the forest is trained on a random subset of the training data and is restricted to use a random subset of features when growing the tree. For classification, the random forest will output the class most commonly selected by its trees. The idea behind this approach is that an uncorrelated forest of trees will outperform any individual tree on a given task [5].

II. Methods

A. Dataset

The dataset used in this work consists of 7125 training instances and 999 test instances. These instances are all taken from [3]. Each instance (i.e., mushroom) in the dataset is described by a set of 22 features as well as its class (either poisonous or edible).

A subset of the training data is selected as a validation dataset. Examples in validation datasets are withheld during training and evaluated during hyperparameter tuning as discussed below.

B. The Decision Tree

Decision trees are a method used in classical machine learning. They are best suited for categorical data where features and output classes can be placed into bins. Each node represents an attribute of the dataset's examples and branches represent the values that the attribute can assume. Leaf nodes yield the final output classification for a given branch down the decision tree.

C. Measuring Information Gain

Decision trees use information gain to determine the best attribute to use for splitting a tree at a given node. Conceptually, information gain should be maximized, so less knowledge is needed to represent the dataset with its classes. We optimize for the attribute that yields the most homogeneous distribution of output classes such that tree depth is minimized [1]. For attributes with missing values, such as stalk-root, information gain is computed using the examples containing real values. The examples with the missing values are passed down the majority branch.

Three different splitting criteria are used when computing information gain: entropy, Gini Impurity, and misclassification error.

Entropy is a classic measurement in Computer Science used to model the information needed to transmit a member of a collection of items. It is usually expressed in bits using the base-2 logarithm as seen in Equation 1. Using entropy, information gain is interpreted as the number of bits no longer needed to encode the dataset after a given split with respect to the root node.

$$Entropy(S) = -\sum_{v \in vals(A)} p_i \log_2(p_i) \tag{1}$$

Gini Impurity is based on the chances of obtaining some output classes in a random sample of 2 examples. The quantity in Equation 2 represents the probability of choosing 2 examples in the data set that are not the same output class. In this case, information gain represents the reduction in probability of choosing two examples with distinct output classes from the dataset.

$$Gini(S) = 1 - \sum_{v \in vals(A)} {p_i}^2 \tag{2}$$

Lastly, misclassification error (M_E) measures impurity by computing the largest error that occurs in the training examples present at a given node. Equation 3 represents the error if the majority output class among the examples was the final classification during evaluation of the minority examples. For misclassification error, information gain represents the reduction in chance of misclassifying samples at the target node given that classification halts there.

$$M_E = 1 - \max(p_1, p_2, ..., p_k) \tag{3}$$

D. Training Models

To train a decision tree, the training dataset is examined at each node in order to choose the best attribute to split the dataset. The attribute that yields the highest information gain according to a given impurity metric is chosen. The dataset is then split among the attribute's values and recursively processed without the selected attribute.

In order to ensure the splits are significantly more homogeneous than the distribution of classes at the current node, the growth of a tree is terminated early according the the chi-squared statistic, which is given by:

$$\chi^2 = \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k} \tag{4}$$

Where E_k is the expected count and O_k is the observed count in a split for a given attribute/class k. The chi-squared statistic is compared with a critical value based on the desired confidence level. If it is determined that the split is more homogeneous than the expected proportions due to random chance, the split is continued among the attribute's branches. Otherwise, the tree stops growing at the current node. α is the area in the right tail of the distribution; smaller α values require the class proportions to be less random.

Training a random forest of decision trees involves training individual trees with different data selections. In this experiment, two bagging techniques are used in order to introduce randomness. First, random subsets of the training data are given to each tree in the hope that trees will develop different decision boundaries while still being able to generalize. Second, at each node in each tree, a random subset of the available features is made available to split criterion (information gain) so that each tree does not select the same features at each node of the tree.

During bagging, sometimes attribute values are not seen when training. To ensure any example can be classified during evaluation, a default branch is created during training according to the most common value. In these cases, the majority class from training is returned as the result.

E. Model Evaluation

To evaluate the quality of models we use classification accuracy, which is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

Where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives.

III. Experiments and Results

A. Hyperparameter Tuning

Hyperparameters are used to train each random forest. These parameters must be chosen by the programmer and can have a significant impact on model performance. Hyperparameters include 1) split criterion, 2) data and feature bagging ratios, 3) χ^2 significance (α) levels, 4) count of trees in the random forest, and 5) random seeds for bagging and splitting.

Various grid searches were performed on the random forest model using the poisonous mushroom dataset. The following parameters were explored in Table I.

Parameter	Options
Split criterion	Entropy, Gini, ME
Data bagging ratio	0.1, 0.2, 0.3, 0.4
Feature bagging ratio	0.1, 0.2, 0.3, 0.4
$\chi^2\alpha$ value	0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.99
Tree count	10, 20

TABLE I
HYPERPARAMETER COMBINATIONS EXPLORED DURING TUNING.

During hyperparameter tuning, we observed 100% accuracy on all 80/20 train/validation splits as well as in the Kaggle competition. To increase the impact of hyperparameter tuning, a much more restrictive 99/1 split was used to tune hyperparameters.

To decrease necessary computation, narrower searches of the hyperparameters were completed.

1) Split Criterion: When examining split criterion across multiple hyperparameter configurations, ranges of accuracy for each metric were: 48.1429% to 98.9509% with an average of 86.1308% for entropy, 48.1429% to 98.9509% with an average of 86.2160% for Gini index, and 47.3632% to 99.1352% with an average of 86.5073% for misclassification error.

All metrics generally perform similarly, although misclassification error has a higher maximum and average accuracy. We predict that, since the mushroom classification problem seems to be simple, penalizing errors is beneficial.

Data bag ratio	Acc. Range
0.1	77.2813% (47.4%-96.6%)
0.2	85.8328% (48.2%-98%)
0.3	90.2915% (48.1%-99.1%)
0.4	91.7331% (51.7%-99.1%)

TABLE II
VALIDATION ACCURACIES (MEAN AND MIN-MAX) FOR LEVELS OF DATA BAGGING.

2) Bagging ratios: Data and feature bagging seem to have a significant effect on model performance. As seen in Table II and III, the extremes in accuracy seem consistent among

Feat. bag ratio	Acc. Range
0.1	81.9246% (47.4%-97.5%)
0.2	85.8794% (47.4%-97.9%)
0.3	87.5890% (48.1%-99%)
0.4	89.7457% (48.1%-99.1%)

TABLE III
VALIDATION ACCURACIES (MEAN AND MIN-MAX) FOR LEVELS OF FEATURE BAGGING.

ratios, although higher ratios seem to yield a 1% to 3% bump on both ends.

More dramatic differences are seen in terms of average accuracies. With data bagging, a 14.5% improvement in accuracy is seen when exploring the 0.1 to 0.4 bagging ratios. A more modest improvement of 7.8% is seen with feature bagging.

3) χ^2 : α values: Values of α can have a significant impact on performance. As shown by Table IV, increasing the value of α led to increases in average performance, although we note that the maximum accuracy is close to 1.0 for all values of α. It's likely that the value of α only comes into play when using limiting values for other parameters (e.g., a feat. bag ratio of 0.1). We postulate that the reason increasing the value of α led to better average performance is that the mushroom classification problem is simple enough that overfitting the training data is actually beneficial in terms of validation accuracy.

α	Acc. Range
0.01	0.7499 (0.48 - 0.99)
0.05	0.8246 (0.48 - 0.99)
0.1	0.8616 (0.47 - 0.99)
0.25	0.8922 (0.59 - 0.99)
0.5	0.9016 (0.60 - 0.99)
0.75	0.9048 (0.66 - 0.99)
0.99	0.9052 (0.69 - 0.99)

TABLE IV
VALIDATION ACCURACIES (MEAN AND MIN-MAX) FOR DIFFERENT VALUES OF α.

4) Tree counts: To explore the effect of different tree counts, we restrict the search-space by fixing the feat. bag ratio to 0.4 and data bag ratio to 0.3. The α value is fixed to 0.05 (95% confidence). Trees are trained and evaluated using all three split criterion. We test tree lengths within the range $[10, 100]$ at an increment of 10. The results of these experiments are shown in Table V.

As can be seen in Table V, increasing the tree size past 30 had little impact on performance with the parameter set described above. At around 30 trees a ceiling is reached, and increasing the number of trees has little impact of performance.

IV. DISCUSSION

A. Structural Properties

The three information gain metrics produce interesting structural differences in terms of tree depth. In terms of maximum depth, entropy and Gini Impurity produce forests

Number of Trees	Acc. Range
10	0.8869% (0.88% - 0.90%)
20	0.9365% (0.93% - 0.94%)
30	0.9483% (0.94% - 0.95%)
40	0.9453% (0.94% - 0.95%)
50	0.9467% (0.94% - 0.95%)
60	0.9519% (0.94% - 0.96%)
70	0.9607% (0.95% - 0.96%)
80	0.9528% (0.94% - 0.97%)
90	0.9552% (0.95% - 0.96%)
100	0.9534% (0.94% - 0.96%)

TABLE V
VALIDATION ACCURACIES (MEAN AND MIN-MAX) FOR DIFFERENT TREE COUNTS IN THE RANDOM FOREST.

with a mean maximum depth per tree of 2.9 (2.9 for entropy, 2.89 for Gini impurity) while misclassification error produces longer trees of mean maximum depth of 3.2. This is also seen when looking at the maximum depth of a tree in a forest; entropy and Gini impurity produce trees of depth 12 while misclassification error produces depth 14 trees.

Usually, deeper trees yield worse performance because more complete trees don't generalize. Interestingly, misclassification error gives the highest average accuracy while having longer trees. As mentioned previously, we postulate that this problem is simple enough that overfitting to the training data is effective.

Accuracy also depended on the feature selected to split the data in the root node. As seen in Figure 1 in the Appendix, many attributes generated similarly performing trees in terms of average accuracy. However, gill size (gll-sz in the figure) stands out among all of the features with low deviation from the mean as well as a higher minimum accuracy compared to the other attributes. Thus, gill size must create significantly more homogeneous splits during training.

Changing the value of α led to noticeable changes in tree structure. As can be seen in Table VI, increasing the value of α led to deeper trees. This follows from the fact that a larger value of α leads to a lower confidence level needed to continue growing the tree, and so larger values of α allow each tree to fit more to the training data in comparison to trees using smaller values of α. The mean average tree depth for $\alpha = 0.01$ is below 1 because at such a high confidence level (coupled with a restricted set of features set by feat. bag ratio) many trees do not immediately find an attribute that passes the χ^2 test, and thus are not able to grow at all. These depth-0 trees will simply output the mode class (i.e., poisonous or edible) from their bagged training dataset. This highlights a potential danger of using too high of a confidence level when building decision trees, as extremely low values of α can lead to one's decision trees learning next to nothing.

α	Mean Avg. Depth	Mean Max Depth
0.01	0.57	1.48
0.05	1.00	2.06
0.1	1.28	2.38
0.25	1.80	3.23
0.5	2.11	3.76
0.75	2.25	3.99
0.99	2.27	4.02

TABLE VI
OVERVIEW OF TREE STRUCTURAL CHANGES FOR DIFFERENT VALUES OF α.

REFERENCES

[1] Mitchell, T. M. (1997). Machine learning (Vol. 1). McGraw-hill New York.

[2] Gold JA, Kiernan E, Yeh M, Jackson BR, Benedict K. Health Care Utilization and Outcomes Associated with Accidental Poisonous Mushroom Ingestions — United States, 2016–2018. MMWR Morb Mortal Wkly Rep 2021;70:337–341. DOI: http://dx.doi.org/10.15585/mmwr.mm7010a1

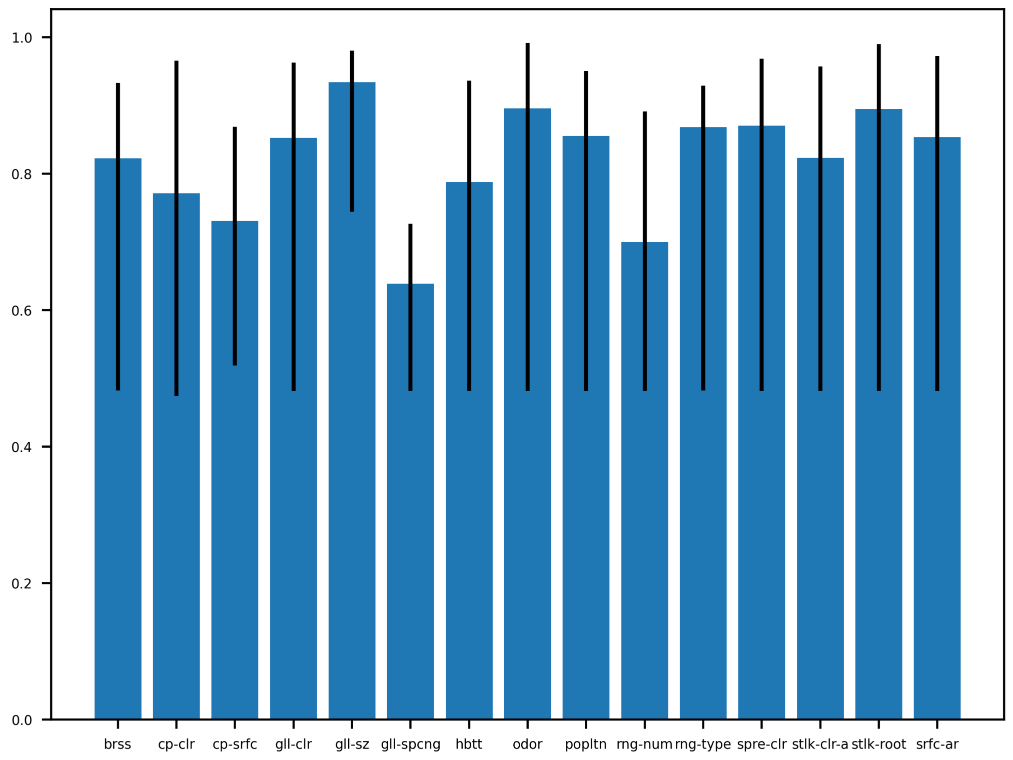
[3] The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf

[4] Tin Kam Ho, "Random decision forests," Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 1995, pp. 278-282 vol.1, doi: 10.1109/ICDAR.1995.598994.

[5] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

V. Appendix

Fig. 1. Accuracy bounds for a subset of features selected first. Confidence intervals depict the minimum and maximum accuracy when a feature is selected first.


brss
cp-clr
cp-srfc
gll-clr
gll-sz
gll-spcng
hbtt
odor
popltn
rng-num
rng-type
spre-clr
stlk-clr-a
stlk-root
srfc-ar