Classifying Mushrooms as Edible or Poisonous:

Comparing Association Rules, a Decision Tree Classifier, and a Random Forest Classifier

Brian Eager

Bellevue University

Abstract

This paper uses the mushrooms dataset from the UCI Machine Learning Repository to demonstrate three approaches to classification: (1) association rules derived from frequent itemsets, (2) a decision tree classifier, and (3) a random forest classifier. For each approach, the process of analysis is explained first, then a statement of the results obtained from its application to the mushroom dataset from the UCI Machine Learning Repository, followed by the advantages and limitations of the approach. Finally, the three approaches are compared to show that there is a trade-off in this case between the explicability, accuracy, and range of applicability of the approaches. Those that are most easily explained in terms of a set of "rules of thumb" for classifying observations correctly suffer either in terms of their accuracy or the scope of their applicability, whereas the most accurate approach is a bit of a black box that is not easily summarized in terms of "rules of thumb."

Keywords: categorical data, itemset, relative support, association rules, antecedent support, consequent support, support, confidence, generalized itemset, decision tree, decision node, leaf node, entropy, information gain, random forest, bootstrap sample, out of bag error

Classifying Mushrooms as Edible or Poisonous

Comparing Association Rules, a Decision Tree Classifier, and a Random Forest Classifier

There are numerous approaches to classifying observations in a dataset according to a target label. The reason for this is that not all approaches are right for a given situation and it depends largely on what kind of context-specific outcome one hopes to achieve. This paper demonstrates the advantages and limitations of three different approaches in the specific context of classifying mushrooms as either edible or poisonous based on their attributes.

The Mushroom Dataset

The data used in this paper all comes from the mushroom dataset of the UCI Machine Learning Repository (*UCI machine learning repository: Mushroom data set*, 1987) as represented in a CSV file posted on Kaggle (*Mushroom classification*, 2016). Each of the 8,124 rows represents data concerning one mushroom, with 22 of the columns representing categorical attributes of the mushrooms, and one column representing the target label indicating whether the mushroom is edible or poisonous (*UCI machine learning repository: Mushroom data set*, 1987). The values from the Kaggle CSV file were decoded according to the values indicated on the UCI Machine learning repository website so that each was a full English word rather than a single character. Investigation of value counts for each attribute show that about 51.8% of the mushrooms are labeled edible and 48.2% labeled poisonous. The *stalk-root* attribute is the only attribute with missing data, with the value missing for 720 edible mushrooms and 1,760 poisonous mushrooms. This means that dropping the rows with missing values would upset the balance between edible and poisonous mushrooms in the dataset, so the decision was made to drop the attribute column instead. The *veil-type* attribute has the same value for every mushroom in the dataset, so it was also dropped. Thus, the final dataset used for this paper contains 8,124

observations of mushrooms with 20 categorical attributes potentially useful for predicting the one target attribute. In order to develop some intuition about the data, a table was made listing the values for each attribute where a given value for that attribute was present for at least half of the mushrooms with a given target label (See Table 1 to the right). The *bruises*, *odor*, *gill-size*, *stalk-surface-above-ring*, and *stalk-surface-below-ring* attributes (highlighted in yellow in Table 1) stand out as the most likely to be independently useful in differentiating between edible and poisonous mushrooms. This is because for each attribute, one particular value of that attribute is found in the majority of edible and in the majority of poisonous mushrooms, with the majority value differing for edible and poisonous mushrooms. With this modified dataset that now has no missing values and with some intuition about which attributes might be the most useful for predicting the target labels, we are ready to explore our first data mining approach to this classification problem.

ATTRIBUTE	EDIBLE	POISONOUS
bruises	"yes" for 65.4%	"no" for 84.1%
odor	"none" for 81.0%	"foul" for 55.2%
gill-attachment	"free" for 95.4%	"free" for 99.5%
gill-spacing	"close" for 71.5%	"close" for 97.1%
gill-size	"broad" for 93.2%	"narrow" for 56.8%
stalk-shape	"tapering" for 56.9%	"tapering" for 51.5%
stalk-surface-above-ring	"smooth" for 86.5%	"silky" for 56.9%
stalk-surface-below-ring	"smooth" for 80.8%	"silky" for 55.2%
veil-color	"white" for 95.4%	"white" for 99.8%
ring-number	"one" for 87.4%	"one" for 97.2%

Table 1: Proportion of attribute values by target label.

Frequent Itemsets and Association Rules

An immediately intuitive way of classifying mushrooms as edible or poisonous, is to look for certain sets of qualities that are more associated with one target label than the other. One can then express these sets of qualities and their associated target labels as a sort of "rule of thumb" for classifying the mushrooms. The data mining technique for approaching the problem in this way is a two-step process: (1) find frequent sets of mushroom qualities, and (2) find and quantify the strengths of the association between these itemsets and the target labels (Abbot, 2014).

Description of the Approach

The simplest way to think about itemset mining is in terms of shopping carts, where a transaction is associated with a set of items in the cart (Zaki & Meira, 2020, p.219). When mining data for itemsets, the dataset is typically viewed in one of three ways: (1) as a binary database, where each observation represents a transaction with a Boolean attribute associated with each possible item that indicates whether or not that item is part of the transaction; (2) as a transaction database, where each transaction is mapped to the set of items in its cart; or (3) as a vertical database, where each item is mapped to the set of transactions in which it is included (Zaki & Meira, 2020, p. 221). For this paper, the mushroom dataset was converted to a binary database, where each row represented a mushroom, and each column was a Boolean representing whether a particular attribute value belonged to that mushroom or not. For example, the *bruises* attribute has two possible values in the original mushroom dataset: "yes" or "no". These possible attribute values are each independently represented by their own Boolean columns in the binary database, *yes_bruises* and *no_bruises*, where the value is True if a the mushroom represented by that row has that value for its *bruises* attribute. The same conversion was done for all 20 attributes so that each transaction represented in the binary database corresponds to a mushroom in the original dataset, and each value of each attribute in the original database is represented as a possible item to be included in an itemset associated with that transaction.

Using the *mlxtend* Python library's *apriori* function (*Apriori—Mlxtend*, n.d.), this binary database was mined for frequent itemsets with a minimum relative support of 0.3, which is to say that the resulting sets of attribute values occur in at least 30% of the mushrooms in the original dataset (Zaki & Meira, 2020, p.221). The resulting frequent itemsets were then mined for association rules using the *association_rule* function from *mlxtend* (*Association rules—Mlxtend*,

n.d.). Each association rule is of the form {antecedent itemset} → {consequent itemset}, with both the antecedent and the consequent being frequent itemsets mined in the first step (Zaki & Meira, 2020, p.237). These rules were filtered down to only those that have either {edible} or {poisonous} as their consequent. The minimal antecedent sets were then found for rules with a given confidence threshold. The resulting rules were then sorted in decreasing order by confidence, which is the conditional probability of the consequent given the antecedent (Zaki & Meira, 2020, p.222), and in increasing order by antecedent length. The reason for this sorting order is to list the rules with the highest confidence first and to break ties by putting the most memorable (i.e., shortest) antecedents first.

Results

Antecedent	Consequent	Ant. Sup.	Support	Confidence
odor=none, stalk-shape=tapered	edible	0.307	0.307	1.000
odor=none, ring-number=one, gill-size=broad	edible	0.331	0.331	1.000
bruises=no, gill-attachment=free, population=several, gill-spacing=close	poisonous	0.311	0.302	0.972
bruises=no, ring-number=one, gill-attachment=free, gill-spacing=close	poisonous	0.406	0.388	0.956
bruises=no, gill-attachment=free, population=several	poisonous	0.323	0.308	0.954
bruises=no, gill-attachment=free, gill-spacing=close	poisonous	0.410	0.390	0.952
bruises=no, gill-spacing=close, population=several	poisonous	0.323	0.302	0.936
bruises=no, population=several	poisonous	0.335	0.308	0.921
bruises=no, ring-number=one, gill-spacing=close	poisonous	0.429	0.388	0.904
bruises=no, gill-spacing=close	poisonous	0.436	0.392	0.901

Table 2: Some of the association rules mined from the mushroom binary database.

All observations were used in the itemset and association rule mining process, so the results must be viewed as descriptive statistics rather than as rules to be used for predictions. Still, some of the rules have a 100% confidence, which indicates that those rules would have predictive value if the approach had not been descriptive. Thus, one could say that there are at least two “rules of thumb” that indicate a mushroom is edible: (1) It is odorless and has a tapered stalk, and (2) It is odorless, has one ring, and a broad gill size. The caveat is that these rules only

apply to about one third of mushrooms, with the antecedent and association supports being only about 0.307 and 0.331, indicating that these rules are only useful for about one third of mushrooms in the dataset. In fact, the most widely applicable rules in Table 2 are still only applicable to fewer than half of the mushrooms in the dataset. Looking at the elements of the antecedent itemsets, it looks like the two most important attributes are likely *odor* and *bruises*. In fact, the analysis in this project showed that the simple association rule {has no bruises} → {poisonous} has a confidence of about 70%!

Advantages and Limitations of this Approach

The main advantage of this approach is that it is extremely intuitive, with the results being directly formatted as rules. This makes the results easy to explain in terms of memorable "rules of thumb" associated with a particular value of the target label. The main limitation of this approach, even if it is used in a predictive way, is that the results do not allow us to draw conclusions about the negation of the antecedent. For example, our results indicate that mushrooms with no odor and a tapering stalk shape are always edible in this dataset, but we are not thus able to draw any conclusions about the mushrooms that have an odor or a stalk shape that is not tapering. Even though this rule has 100% confidence, it can truly only be used to provide information about one third of mushrooms in the dataset.

Decision Tree Classifier

Description of the Approach

A decision tree classifier is a machine learning algorithm that uses a series of Boolean tests in succession in order to predict the target label of a given unlabeled observation. The implementation used for this paper follows Algorithm 19.1 from the second edition of *Data Mining and Machine Learning* (Zaki & Meira, 2020, p.488) using entropy and information gain

as its metric. It uses both the presence of a particular attribute and its absence as useful information to classify a given observation.

When the decision tree is being fit to a training set, each value of each attribute is evaluated to decide which one results in the most information gain. Once a split point is found, a decision node is added with the Boolean condition *observation[attribute] == split_value*, with a child for the subset of the training set where this condition evaluates to False and a child for the subset where it evaluates to True. If the resulting subset meets a minimum purity threshold—purity being the percentage of observations in the subset with the most frequent value of the target label in that subset (Zaki & Meira, 2020, p.486)—or if the subset's size falls below a certain threshold, then a leaf node is created. Otherwise, another decision node is created. This process continues recursively until every branch of the decision tree terminates in a leaf node. When the tree is predicting the value of the target label for an unlabeled observation, it works its way from the root node through the decision nodes based on the Boolean conditions applied to the attributes of the unlabeled observation until it arrives at a leaf node, at which point the prediction is the most frequent target label in the subset of the training data represented by that leaf node.

Results

Two different decision trees were created for the purposes of this analysis: a descriptive tree that was allowed to perfectly fit the entire dataset and a predictive tree that was fit to a training set without overfitting and tested on a separate set of observations.

```
Decision: odor = none, Gain= 0.529
    If True, Decision: spore-print-color = green, Gain= 0.111
        If True, Leaf: label= poisonous, purity= 1.0, size= 72
        If False, Decision: stalk-surface-below-ring = scaly, Gain= 0.068
            If True, Decision: gill-size = narrow, Gain= 0.863
                If True, Leaf: label= poisonous, purity= 1.0, size= 40
                If False, Leaf: label= edible, purity= 1.0, size= 16
            If False, Decision: cap-surface = grooves, Gain= 0.011
                If True, Leaf: label= poisonous, purity= 1.0, size= 4
                If False, Decision: gill-size = broad, Gain= 0.005
                    If True, Leaf: label= edible, purity= 1.0, size= 3200
                    If False, Decision: bruises = no, Gain= 0.144
                        If True, Leaf: label= edible, purity= 1.0, size= 192
                        If False, Leaf: label= poisonous, purity= 1.0, size= 4
    If False, Decision: bruises = yes, Gain= 0.382
        If True, Decision: odor = foul, Gain= 0.346
            If True, Leaf: label= poisonous, purity= 1.0, size= 288
            If False, Decision: odor = pungent, Gain= 0.799
                If True, Leaf: label= poisonous, purity= 1.0, size= 256
                If False, Leaf: label= edible, purity= 1.0, size= 800
        If False, Leaf: label= poisonous, purity= 1.0, size= 3252
```

Figure 1: Descriptive decision tree for the mushroom dataset.

Descriptive Tree. When allowed to perfectly fit the entire mushroom dataset, the resulting decision tree (represented in Figure 1 on the previous page) has eleven leaf nodes, meaning that there are eleven different combinations of True or False for the various attributes that were chosen. The results of this descriptive analysis are similar in some ways to the results of our earlier statistical analysis and itemset mining in that the two most important attributes appear to be *odor* and *bruises*.

Predictive Tree. The mushroom dataset with 20 attributes was split into two subsets such that one contained 75% of the observations and the other contained the remaining 25%. A decision tree was trained on the larger subset using a minimum leaf size of 5 and a purity threshold of 0.95. These threshold parameters ensure that leaf nodes were created earlier than in the descriptive tree, preventing overfitting and making the tree easier to explain. Each leaf node can be thought of as a "rule of thumb", something like an association rule with a generalized itemset specifying either the presence or absence of certain attribute values as its antecedent. From the top of the tree to the bottom, the leaf nodes can be summarized as the following "rules of thumb":

```
Decision: odor = none, Gain= 0.537
    If True, Leaf: label= edible, purity= 0.966, size= 2644
    If False, Decision: bruises = no, Gain= 0.37
        If True, Leaf: label= poisonous, purity= 1.0, size= 2448
        If False, Decision: odor = foul, Gain= 0.356
            If True, Leaf: label= poisonous, purity= 1.0, size= 227
            If False, Decision: odor = pungent, Gain= 0.808
                If True, Leaf: label= poisonous, purity= 1.0, size= 192
                If False, Leaf: label= edible, purity= 1.0, size= 582
```

Figure 3: A predictive decision tree for the mushroom dataset.

- Rule 1: If it's odorless, it's probably edible. (Leaf 1).
- Rule 2: If it has no bruises and it has a noticeable odor, it's probably poisonous. (Leaf 2)
- Rule 3: If it has bruises and it has a foul or pungent odor, it's probably poisonous. (Leaves 3 and 4).
- Rule 4: If it has bruises and it had a noticeable odor that is neither foul nor pungent, it's probably edible. (Leaf 5).

When applied to the test set, the accuracy rates for these rules (summarized in Figure 4) are very high.

		Actual	
		edible	poisonous
Predicted	edible	1071	31
	poisonous	0	929

Overall Accuracy	0.984737
Overall Failure	0.015263
Edible Accuracy	0.971869
Edible Failure	0.028131
Poisonous Accuracy	1.000000
Poisonous Failure	0.000000

Figure 4: Accuracy rates for the predictive decision tree.

When the rules suggested that a mushroom is poisonous, they were always correct. When the

rules suggested that a mushroom was edible, they were accurate 97.2% of the time. While the accuracy rates are high, not all misclassifications are equal, and it would have been better to be wrong about a mushroom being poisonous than to be wrong about a mushroom being edible. In the former case, following the rules would lead to someone failing to eat an edible mushroom. In the latter case, following the rules would lead to eating a poisonous mushroom that could lead to either sickness or death! One thus wishes there were a way to reduce that 2.8% failure rate.

Advantages and Limitations of this Approach

A significant advantage of this approach is its applicability to all observations, which os due to the lack of a particular attribute value being treated as useful information. It is also fairly simple to explain because each decision node can be summarized as a "rule of thumb" concerning which attributes values are required to be either present or absent to result in a particular prediction for the target label. In this case, the rules are memorable and useful even without using a computer, but it would be great to have a predictive model that never recommended a poisonous mushroom as edible.

Random Forest Classifier

A random forest classifier is an ensemble classifier that bags multiple decision tree classifiers and asks each of its individual trees for its prediction. The majority vote of these trees is the prediction made by the forest, which prediction is frequently more accurate than that of one decision tree classifier. Random forest classifiers are also less prone to overfitting due to the effects of bootstrap sampling and a slightly altered decision tree algorithm, as described below.

Description of the Approach

The Random Forest Algorithm. The implementation for this paper follows Algorithm 22.5 in the second edition of *Data Mining and Machine Learning* (Zaki & Meira, 2020, p.576).

As with the decision tree classifier, the random forest was trained on 75% of the observation and tested for accuracy on the remaining 25%. Each tree in the forest was trained on a randomly selected bootstrap sample of the training set (of the same size as the training set and with replacement), such that each individual tree's training set has duplicates of some observations from the training set and does not contain others. The training examples that are not in an individual tree's bootstrap training set are considered "out of bag" observations for that tree (Zaki & Meira, 2020, p.577). The algorithm for building an individual tree is the same as for the decision tree classifier with the exception that each decision node searches for the optimal split value across a randomly selected subset of the attributes rather than across all the attributes (Zaki & Meira, 2020, p.576-577). It occasionally happened that there was no information gain possible from any of the four randomly selected features, in which case the trees were instructed to make a leaf node. The overall result of each tree in the forest having only seen a random bootstrap sample of the training examples, each decision node of each tree having only considered a random subset of the available features, and each tree being allowed to overfit its training data is that each tree consists of a random set of "rules of thumb" that provide highly accurate predictions for the observations that it was trained on. When the forest predicts a label for an observation, it asks each tree to predict the label using its random "rules of thumb". Whichever label receives the most votes from the trees is the label predicted by the forest.

Values Chosen for the Hyperparameters. There are four hyperparameters for the random forest model: (1) the number of decision trees in the forest, (2) the minimum leaf size for the trees, (3) the minimum leaf purity for the trees, and (4) the number of features randomly selected for consideration by each decision node (Zaki & Meira, 2020, p.576). For this paper, the number of trees in the forest was chosen with reference to the out of bag error, which is the

error rate of the random forest's predictions of the label of each of the training examples using only the trees in the forest for which that sample was out of bag (Zaki & Meira, 2020, p.577). This out of bag error can serve as an approximation of the model's actual error rate. By plotting the number of trees along the x-axis and the corresponding out of bag error along the y-axis, we get a graph approximating a decreasing exponential function (shown in Figure 5). A good choice for the number of trees in the model will be where the graph substantially flattens out because that is the point at which the computational time for the additional trees is resulting in less of a reduction in the out of bag error rate. In this case, that seemed to be around 15 trees, which was the number chosen for the hyperparameter. The number of features randomly selected f or consideration by each decision node was chosen according to the frequently-used rule $floor(\sqrt{d})$, where d is the total number of features (Zaki & Meira, 2020, p.576), resulting in $floor(\sqrt{20}) = 4$. The minimum leaf size and minimum leaf purity were each set to 1 to allow each tree to fit its bootstrap sample as completely as possible.

Num_Trees	OOB_Error
1	0.617594
5	0.110619
11	0.005580
15	0.000821
21	0.000164
31	0.000000
41	0.000000
51	0.000000

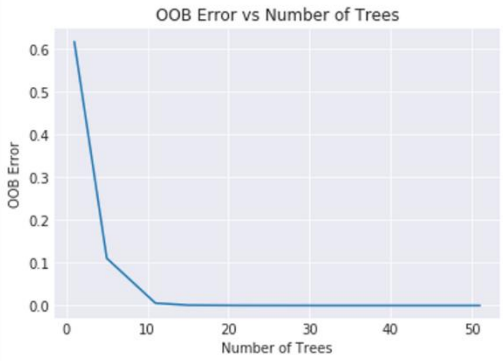

Figure 5: Out of Bag Error vs Number of Trees in the Forest

Results

The random forest described above was able to classify the 25% of observations that none of its trees had ever seen in training with perfect accuracy, as shown in Figure 6.

		Actual	
		edible	poisonous
Predicted	edible	1071	0
	poisonous	0	960

Overall Accuracy	1.0
Overall Failure	0.0
Edible Accuracy	1.0
Edible Failure	0.0
Poisonous Accuracy	1.0
Poisonous Failure	0.0

Figure 6: Accuracy rates for the random forest classifier on the mushroom dataset.

Advantages and Limitations of this Approach

The main advantages of this approach are that it is able to so accurately classify the observations and is not prone to overfitting. The main disadvantage of the approach is that it requires a lengthy explanation in order to understand how the model is making its predictions. It also cannot be easily summarized as memorable rules of thumb and requires a computer to be used to make predictions about a given mushroom. This might be fine if someone has a computer with them and is able to enter the value for each of its 20 attributes as they consider whether a mushroom is edible or poisonous, but this random forest model is completely useless without a computer.

Conclusion: Comparison of the Approaches

The association rules approach produces "rules of thumb" that are easy to remember, explain, and apply without a computer. They have a high degree of confidence; however, these rules only apply to a subset of the observations because they do not use the absence of a given attribute value as information. One could view these as "play-it-safe" rules. The decision tree classifier produces highly accurate results that can be summarized a set of "rules of thumb" and applied without a computer. Furthermore, they allow for the classification of any mushroom because even the absence of an attribute value is used as information in the classification process. The random forest classifier produces the most accurate predictions, but it cannot be readily summarized as "rules of thumb" and cannot be applied without a computer. As we can see, there is a trade-off in this situation between the explicability, applicability, and accuracy of a model. The most accurate and universally applicable model—the random forest—suffers in terms of explicability. The model with the narrows range of applicability—the itemset association rules—excels in accuracy and explicability. The decision tree is perhaps the happy

medium between these two, being universal applicable and highly accurate. Which approach is considered the best depends very much on the goal of the model. If the goal is to find memorable, safe, and nearly certain rule(s) that a given mushroom is edible even if it only applies to some mushrooms, then the association rules approach is the best approach. If the goal is to find a set of memorable rules that can be applied to any mushroom with a high degree of accuracy and without the use of a computer, then the decision tree classifier is the best approach. If the goal is simply to have the most accurate predictions, without regard for the explicability of the model or the technological requirements for its use, then the random forest classifier is the best approach.

References

Abbott, D. (2014). *Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst*. John Wiley & Sons, Inc.

Apriori—Mlxtend. (n.d.). Retrieved August 4, 2020, from http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/

Association rules—Mlxtend. (n.d.). Retrieved August 4, 2020, from http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

Mushroom classification. (2016). https://kaggle.com/uciml/mushroom-classification

UCI machine learning repository: Mushroom data set. (1987, April 27). Retrieved June 20, 2020, from https://archive.ics.uci.edu/ml/datasets/Mushroom

Zaki, M. J., & Meira, Jr., W. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (Second Edition). Cambridge University Press.