Distinguish Wild Mushrooms with Decision Tree

Shiqin Yan
Introduction

Mushroom poisoning, which also known as mycetism, refers to harmful effects from ingestion of toxic substances present in the mushroom. Its symptoms varies from slight gastrointestinal discomfort to death. Although many edible mushrooms has distinctive features that could be easily differentiated from poisonous mushrooms, experienced wild mushroom collectors are still potentially eating poisonous mushrooms despite of well awareness of the risk. Most of this is caused by close resemblance in terms of color and general morphology of the toxic mushrooms with edible species. In the meantime, it would be difficult for collectors and botanists to differentiate newly discovered mushrooms. They have to have a sophisticated understanding and experience with mushrooms and would still possibly mis-identify the mushroom in the rare occasion.

As the knowledge of the mushroom, whether they are edible or poisonous, turns out to be very important to botanists and explorer, the objective of this project is to extract the most informative features that could facilitate this process of determining the poisonous mushrooms and to predict the poisonous of the mushroom that unseen, thus preventing potentially adverse effect and death caused by mushroom poisoning. In order to realize this objective, this project will utilize the existing database recorded by the Audubon Society Field Guide to North American Mushrooms and results of the research done by Wlodzislaw Duch, Department of Computer Methods, Nicholas Copernicus University, for comparison purpose.

DataSet and Method

Analyzing the Data and Parsing the Data

The data set that is being used contains 8124 of instances with 22 attributes. All of those features are nominally valued. There are 2480 missing attribute value marked “?” for attribute 11. 4208 out of 8124 instances are categorized as edible, which is around 51.8%.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>cap-shape</td>
<td>bell=b, conical=c, convex=x, flat=f, knobbly=k, pubescent=p, sulcate=s</td>
</tr>
<tr>
<td>cap-surface</td>
<td>fibrous=f, grooves=g, scaly=y, smooth=s</td>
</tr>
<tr>
<td>cap-color</td>
<td>brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y</td>
</tr>
<tr>
<td>bruises</td>
<td>bruisable=y, none=n</td>
</tr>
<tr>
<td>odor</td>
<td>almond=a, anise=l, creosote=o, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s</td>
</tr>
<tr>
<td>gill-attachment</td>
<td>attached=a, descending=d, free=f, notched=n</td>
</tr>
<tr>
<td>gill-spacing</td>
<td>close=c, crowded=w, distant=d</td>
</tr>
<tr>
<td>gill-size</td>
<td>broad=b, narrow=n</td>
</tr>
<tr>
<td>gill-color</td>
<td>black=k, brown=n, buff=b, chocolate=b, gray=g, green=g, orange=o, pink=p, purple=u, red=e, white=w, yellow=y</td>
</tr>
<tr>
<td>stalk-shape</td>
<td>enlarging=e, tapering=t</td>
</tr>
<tr>
<td>stalk-root</td>
<td>bulbous=b, club=c, cup=c, equal=e, rhizomorphs=r, rooted=r, missing=m</td>
</tr>
<tr>
<td>stalk-surface-above-ring</td>
<td>ibrous=f, scaly=y, silky=s, smooth=s</td>
</tr>
<tr>
<td>stalk-surface-below-ring</td>
<td>ibrous=f, scaly=y, silky=k, smooth=k</td>
</tr>
<tr>
<td>stalk-color-above-ring</td>
<td>brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y</td>
</tr>
<tr>
<td>stalk-color-below-ring</td>
<td>brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y</td>
</tr>
<tr>
<td>veil-type</td>
<td>partial=p, universal=u</td>
</tr>
<tr>
<td>veil-color</td>
<td>brown=n, orange=o, white=w, yellow=y</td>
</tr>
<tr>
<td>ring-number</td>
<td>none=n, one=o, two=t</td>
</tr>
<tr>
<td>ring-type</td>
<td>cobwebby=c, evanescentes, flaring=f, large=l, none=n, pendant=p, sheathing=s, zoned=z</td>
</tr>
<tr>
<td>spore-print-color</td>
<td>black=b, brown=b, buff=b, chocolate=b, green=r, orange=o, purple=p, white=w, yellow=y</td>
</tr>
<tr>
<td>population</td>
<td>abundant=a, clustered=c, numerous=n, scattered=s, several=s, solitary=s</td>
</tr>
<tr>
<td>habitat</td>
<td>grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d</td>
</tr>
</tbody>
</table>
As the data are categorized by characters instead of numerical values, the decision tree is used to learn the data set so “?” is just a special value of attribute 11 and there is no need to eliminate the instance that contain missing value.

The data set was preprocessed with emacs to eliminate all the commas. Then BufferedReader was used to read all the data into the object DataSet in java. The data set is then split into 3 parts without randomness and used for 3-fold cross-validation where each validation set is used once to evaluate the training result. It should be noted that the data set in this project is not randomly partitioned into k equal size subsamples due to the time constraint.

For loop is used to evaluate the result of the attempts of the cross-validation and store all the accuracy in a double array. For the 3 subsamples, each is used twice as part of the training set and once as the testing set.

**Methodology**

The critical part of this project is to figure out a way to build the decision tree based on the training data and thus splitting the data into different branch of the tree according to their feature values. The method used in this project is mutual information.

\[
H(Y|X = v) = \sum_{i=1}^{k} - \Pr(Y = y_i|X = v) \log_2 \Pr(Y = y_i|X = v)
\]

\[
H(Y|X) = \sum_{v, \text{values of } x} \Pr(X = v) H(Y|X = v)
\]

Mutual information: \(I(Y; X) = H(Y) - H(Y|X)\)

Y: label, X: feature

The above are the equations that used to evaluate the mutual information of attributes. For each layer of the decision tree, we always choose the feature that maximize the mutual information to split the tree. In this way, we can get as many pure leaf nodes as possible. For the leaf node that is not pure, majority vote is used to assign the class label for that node. In the case that there is a tie in the majority vote, edible is always assigned to that leaf node.

This is the abstract decision tree algorithm used in this project:

```java
function DECISION-TREE-LEARNING(examples, attributes, parent_examples) returns a tree

if examples is empty then return PLURALITY-VALUE(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return PLURALITY-VALUE(examples)
else
  A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
  tree ← a new decision tree with root test A
  for each value v_k of A do
    exs ← \{ e : e ∈ examples and e.A = v_k \}
    subtree ← DECISION-TREE-LEARNING(exs, attributes − A, examples)
    add a branch to tree with label (A = v_k) and subtree subtree
  return tree
```
Because there are around 8124 instances and each instance has 22 attributes, pruning is also implemented in this project in order to avoid over-fitting. It should be noted that pruning is not used in the 3-fold cross-validation as we can see in the later results that pruning turns out to be unnecessary and the data set is well fitted in the decision tree.

The pruning method is not hard to implement, which just iterates through the tree nodes and prune each node, then use the tuning set to evaluate the pruned tree result. If after the pruning the accuracy stays the same or even increase, then we replace the tree with the pruned tree. We recursively iterate through each node until the accuracy cannot be improved anymore. When we are doing the pruning, we always start from the top of the tree and iterate to the bottom of the tree.

This is the abstract pruning algorithm used in this project:

```
Prune(tree T, TUNE set)
  1. Compute T’s accuracy on TUNE, call it A(T)
  2. For every internal node N in T:
     a) New tree T_N = copy of T, but prune (delete) the subtree under N.
     b) N becomes a leaf node in T_N. The label is the majority vote of TRAIN examples reaching N.
     c) A(T_N) = T_N’s accuracy on TUNE
  3. Let T* be the tree (among the T_N’s and T) with the largest A(). Set T ← T* /* prune */
  4. Repeat from step 1 until no more improvement available.
  Return T.
```

Results

- Decision Tree without Cross-validation

```
ROOT (odor?)
  a (e)
  l (e)
  c (p)
  y (p)
  f (p)
  m (e)
  n {spore-print-color?}
    k (e)
    n (e)
    b (e)
    n (e)
    r (p)
    e (e)
    u (e)
    w {habitat?}
      g (e)
        l {population?}
          c (e)
          c (e)
          n (e)
          s (e)
          v (e)
          y (e)
          n (e)
          p (e)
          u (e)
          w (e)
          d (p)
          y (e)
          p (p)
          s (p)
Prediction accuracy on the test set is: 0.96898
```
In this attempt, the data is split into 3 subsets: Training: 5416, Tuning: 1354, Testing: 1354. The decision tree achieved around 97% percent accuracy on the test set. The most informative features extracted from this decision tree are odor, spore-print-color, habitat, and population.

- **Decision tree with 3-fold cross-validation**
  (Note: the test data is the entire data set, i.e., 8124 instances)

Cross-Validation Set 1:

Cross-Validation Set 2 (Final selected decision tree):

Features extracted here are the more or less the same with the previous decision tree and it only achieves around 90% accuracy, which is not very impressive.
Impressive!!! 100% accuracy on the entire data set which has 8124 instances!!!
The most informative features extracted from this decision tree are:
  odor, spore-print-color, cap-color, stalk-surface-above-ring, habitat

Cross-Validation Set 3:

The result is still very remarkable, though not a perfect classification.

It achieves 99.458% accuracy on the entire data set!

This is actually the same decision tree which this project attempted at the very beginning without cross-validation.
Prior Research Results

Several attempts to classify mushrooms has been made before. They all achieved very satisfactory results. Schlimmer, J.S. analyzed the mushroom data set in his doctoral dissertation --- Concept Acquisition Through Representational Adjustment (Technical Report 87 – 19). Iba, W., Wogulis, J., and Langley, P. developed a HILLARY algorithm. Most of them are able to achieve around 95% classification accuracy after reviewing certain number of instances. The most noteworthy work is done by Wlodzislaw Duch. He summarized all the logical rules that could be used to classify mushrooms and achieved very impressive accuracy. His work is used as benchmark here to be compared with.

Logical rules for the mushroom data sets.

Logical rules given below seem to be the simplest possible for the mushroom dataset and therefore should be treated as benchmark results.

Disjunctive rules for poisonous mushrooms, from most general to most specific:

P_1) odor=NOT(almond.OR.anise.OR.none)
120 poisonous cases missed, 98.52% accuracy

P_2) spore-print-color=green
43 cases missed, 99.41% accuracy

P_3) odor=none.AND.stalk-surface-below-ring=scaly.AND.
(stalk-color-above-ring=NOT.brown)
8 cases missed, 99.90% accuracy

P_4) habitat=leaves.AND.cap-color=white
100% accuracy

Rule P_4) may also be

P_4') population=clustered.AND.cap_color=white

These rule involve 6 attributes (out of 22). Rules for edible mushrooms are obtained as negation of the rules given above, for example the rule:

odor=(almond.OR.anise.OR.none).AND.spore-print-color=NOT.green

gives 48 errors, or 99.41% accuracy on the whole dataset.

Several slightly more complex variations on these rules exist, involving other attributes, such as gill_size, gill_spacing, stalk_surface_above_ring, but the rules given above are the simplest we have found.

Above are the logical rules summarized by Wlodzislaw Duch. From his rule, we can extracted the most informative features to distinguish mushrooms. They are: odor, spore-print-color, stalk-surface-below-ring, stalk-surface-above-ring, habitat, cap-color, and population. He uses these rules and achieves over 99% accuracy on determining whether the mushroom is poisonous or edible.

From 3-fold cross-validation implemented in this project, the most informative features I find are: odor, spore-print-color, cap-color, stalk-surface-above-ring, habitat, and population. As we can see, the results are more or less the same. There are more complex rules, such as gill size, gill spacing as suggested by Wlodzislaw Duch, existed but not revealed by this project, which requires further exploration.
Discussion and Conclusion

The results obtained based on the decision tree is extraordinary promising. It proves that poisonous mushrooms have their distinct characteristics from edible species which could be used for identification. The tree printed out by the decision tree could be very convenient for determining the poisonous mushroom in a quickly manner. People with even no prior knowledge could easily distinguish mushrooms by simply tracking down the decision tree. The program could also be used for scientist to analyze large amount of data of the mushrooms at once. The cross-validation is critical to this project, as we can see it improves the accuracy from 96% to 100%. While we also need to note that the cross-validation implemented in this project is still far from perfect. It just simply separated the data set in certain order instead of using random partitioning. We also need to note that, in contrast to KNN, which has to calculate the distances between instances based on 22 attributes, decision tree saves plenty of computational time by using the mutual information.

Reference


