CS 5350/6350, DS 4350: Machine Learning Spring 2024

Homework 1

Handed out: January 16, 2024
Due date: January 30, 2024

1 Decision Trees

1. [25 points] Mark loves mangoes. Unfortunately he is lousy at picking ripe mangoes at the grocery. He needs your help. You need to build a decision tree that will help Mark decide if a mango is ripe or not. You need to make the decision based on four features described below:

   (a) **Variety** (*Alphonso, Keitt or Haden*): Describes the variety of the mango.

   (b) **Color** (*Red, Yellow or Green*): Describes the color of the mango.

   (c) **Smell** (*Sweet or None*): Describes the smell of the mango.

   (d) **Time** (*One or Two*): Number of weeks since the mango was plucked.

   You are given the following dataset which contains data for 8 different mangoes. For each mango, the values of the above four features are listed. The label of whether the mango was ripe or not is also provided.

Variety	Color	Smell	Time	Ripe?
Alphonso	Red	None	Two	False
Keitt	Red	None	One	True
Alphonso	Yellow	Sweet	Two	True
Keitt	Green	None	Two	False
Haden	Green	Sweet	One	True
Alphonso	Yellow	None	Two	False
Keitt	Yellow	Sweet	One	False
Alphonso	Red	Sweet	Two	True

Table 1: Training data for the mango prediction problem.

   (a) [5 points] How many possible functions are there to map these four features to a boolean decision? How many functions are consistent with the given training dataset?

Ans) Since there are 4 attributes: Variety which can take 3 values, Colour which can take 3 values, Smell which can take 2 values, Time which can take 2 values. So the total new rows would be $3 \times 3 \times 2 \times 2 = 36$. We know the value of 8 rows out of the 36 rows. Therefore, consistent functions would be $2^{36-8} = 2^{28}$.

(b) [3 points] What is the entropy of the labels in this data? When calculating entropy, the base of the logarithm should be base 2.
Ans) Since there are 4 Truth values and 4 False values,
E(S) = $\sum_{i=1}^{c} -p_i \log_2 p_i$.
E(S)=$-\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$

(c) [4 points] Compute the information gain of each feature and enter it into Table 2. Specify upto 3 decimal places.
Ans) Let us calculate for the attribute Variety.
For variety=Alphonso ($\frac{4}{8}$): Truth=$\frac{2}{4}$ and False=$\frac{2}{4}$
Therefore $S = 1$
For variety=Keitt ($\frac{3}{8}$): Truth=$\frac{1}{3}$ and False=$\frac{2}{3}$
$S = -\frac{1}{3}\log_2(0.333) - \frac{2}{3}\log_2(0.666)$
$S = 0.528 + 0.390 = 0.918$
For variety=Haden ($\frac{1}{8}$): $S = 0$
$\Sigma S = 1 \times \frac{1}{2} + \frac{3}{8} \times 0.918 + \frac{1}{8} \times 0 = 0.5 + 0.344 = 0.844$
Gain = 1 - 0.844 = 0.156
We will do this for all other attributes as well.

Feature	Information Gain
Variety	0.156
Color	0.062
Smell	0.189
Time	0.050

Table 2: Information gain for each feature.

(d) [1 points] Which attribute will you use to construct the root of the tree using the information gain heuristic of the ID3 algorithm?
Ans) Smell attribute is used to construct the Decision tree.

(e) [8 points] Using the root that you selected in the previous question, construct a decision tree that represents the data. You do not have to use the ID3 algorithm here, you can show any tree with the chosen root. **Ans)** Image might have been shifted downwards.

(f) [4 points] Suppose you are given three more examples, listed in Table 3. Use your decision tree to predict the label for each example. Also report the accuracy of the classifier that you have learned.
Ans) According to the tree, First two rows are correct, last row is incorrect.
Hence accuracy is $2/3 * 100$
accuracy=66.666

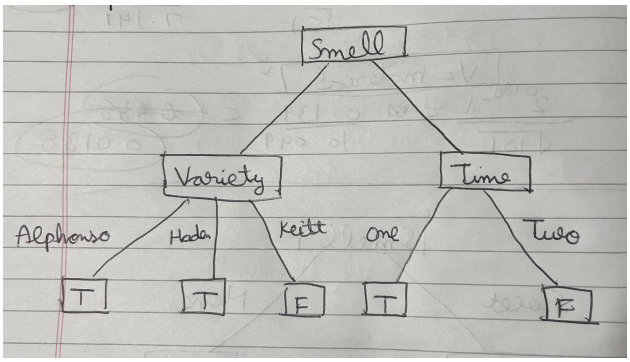

Figure 1: Decision Tree

Variety	Color	Smell	Time	Ripe?
Alphonso	Green	Sweet	Two	True
Keitt	Red	Sweet	One	False
Haden	Yellow	None	Two	True

Table 3: Test data for mango prediction problem

2. [10 points] Recall that in the ID3 algorithm, we want to identify the best attribute that splits the examples that are relatively pure in one label. Aside from entropy, which we saw in class and you used in the previous question, there are other methods to measure impurity.

   We will now develop a variant of the ID3 algorithm that does not use entropy. If, at some node, we stopped growing the tree and assign the most common label of the remaining examples at that node, then the empirical error on the training set at that node will be

   $$MajorityError = 1 - \max_i p_i$$

   where, p_i is the fraction of examples that are labeled with the i^{th} label.

   (a) [2 points] Notice that $MajorityError$ can be thought of as a measure of impurity just like entropy. Just like we used entropy to define information gain, we can define a new version of information gain that uses $MajorityError$ in place of entropy. Write down an expression that defines a new version of information gain that uses $MajorityError$ in place of entropy.

   Ans) Majority Error: $1 - \max(p, 1-p)$
   Majority Error(S): $1 - \max(1, 1-p) = 1 - \max\left(\frac{1}{2}, \frac{1}{2}\right) = 1 - \frac{1}{2} = \frac{1}{2}$
   Gain = Majority Error(S) - $\sum$Majority Error(attribute)

   (b) [6 points] Calculate the value of your newly defined information gain from the previous question for the four features in the mango dataset from 1. Use 3 significant digits. Enter the information gain into Table 4.

   Ans) Let us calculate majority error for the Variety.

   For Alphonso ($\frac{4}{8}$): Truth=$\frac{1}{2}$ and False=$\frac{1}{2}$ Majority error=$\frac{1}{2}$

For Keitt ($\frac{3}{8}$): Majority error=1-max(0.333, 0.666)=0.333
For Haden ($\frac{1}{8}$): Majority error=0
Sum majority error: $\frac{1}{2} \times \frac{1}{2} + 0.333 \times \frac{3}{8} + 0 = 0.25 + 0.124 = 0.374$
Gain=0.5 - 0.374 = 0.126
We will do like this for every feature.

Feature	Information Gain (using majority error)
Variety	0.126
Color	0.127
Smell	0.25
Time	0.126

Table 4: Information gain for each feature.

(c) [2 points] According to your results in the last question, which attribute should be the root for the decision tree? Do these two measures (entropy and majority error) lead to the same tree?
Ans) Since using both Entropy and Majority Error we can see that the feature Smell has the highest value, the decision tree using both metric would be same as it would have the same root feature.

2 Experiments

This problem uses a modified version of the famous the Mushroom Dataset from the UCI machine learning repository. Each data point has 21 features indicating different characteristics of a mushroom. You can find definitions of each feature in the file `information.txt`. The labels are either `p` or `e`, which stand for poisonous and edible respectively.

Your task is to implement the ID3 algorithm and build a decision tree using the training data `data/train.csv`. The goal of the decision tree is to distinguish between poisonous and edible mushrooms. We also provide the test set `data/test.csv` to evaluate how your decision tree classifier is performing.

Programming notes. You may use any programming language for your implementation, but as discussed in class, we prefer Python. The graders should be able to execute your code on the CADE machines without having to install any libraries. Remember that you are not allowed to use any machine learning libraries (ex: `scikit-learn`, `tensorflow`, `pytorch`, etc.), but can use libraries with mathematical functions like `numpy` and data management libraries like `pandas`.

We have provided a file `data.py`, which is code that can help you with the data loading and manipulation if you choose to use Python. We have included a Jupyter notebook `datademo.ipynb` that demonstrates how to use the file and what functionality it provides. Using the code is not required, although we think it should help you with the implementation of the algorithm. (If you are familiar with a library such as `pandas`, you may find that to be helpful instead of using the provided code.)

Cross-Validation

The depth of the tree is a *hyper-parameter* to the decision tree algorithm that helps reduce overfitting. By depth, we refer to the maximum path length from the root to any leaf. That is, a tree with just a single node has depth 0, a tree with a root attribute directly leading to labels in one step has depth 1 and so on. You will see later in the semester that many machine learning algorithm (SVM, logistic-regression, etc.) require choosing hyper-parameters before training commences, and this choice can make a big difference in the performance of the learners. One way to determine a good hyper-parameter values to use a technique called *cross-validation*.

As usual, we have a training set and a test set. Our goal is to discover good hyperparameters using the training set *only*. Suppose we have a hyperparameter (e.g. the depth of a decision tree) and we wish to ascertain whether it is a good choice or not. To do so, we can set aside some of the training data into a subset called the *validation* set and train on the rest of the training data. When training is finished, we can test the resulting classifier on the validation data. This allows us to get an idea of how well the particular choice of hyper-parameters does.

However, since we did not train on the whole dataset, we may have introduced a statistical bias in the classifier caused by the choice of the validation set. To correct for this, we will need to repeat this process multiple times for different choices of the validation set. That is, we need train many classifiers with different subsets of the training data removed and average out the accuracy across these trials.

For problems with small data sets, a popular method is the leave-one-out approach. For each example, a classifier is trained on the rest of the data and the chosen example is then evaluated. The performance of the classifier is the average accuracy on all the examples. The downside to this method is for a data set with n examples we must train n different classifiers. Of course, this is not practical in general, so we will hold out subsets of the data many times instead.

Specifically, for this problem, you should implement k-fold cross validation.

The general approach for k-fold cross validation is the following: Suppose we want to evaluate how good a particular hyper-parameter is. We randomly split the training data into k equal sized parts. Now, we will train the model on all but one part with the chosen hyper-parameter and evaluate the trained model on the remaining part. We should repeat this k times, choosing a different part for evaluation each time. This will give we k values of accuracy. Their average cross-validation accuracy gives we an idea of how good this choice of the hyper-parameter is. To find the best value of the hyper-parameter, we will need to repeat this procedure for different choices of the hyper-parameter. Once we find the best value of the hyper-parameter, we can use the value to retrain we classifier using the entire training set.

Growing decision trees

For this problem, your should use the data in `data` folder. This folder contains two files: `train.csv` and `test.csv`. You should train your algorithm on the training file. Remember

that you should not look at or use your testing file until your training is complete.

1. [6 points] **Baseline**

   First, find the most common label in the training data. What is the training and test accuracy of a classifier that always predicts this label?

   Ans) e has the most common label. Accuracy for predicting e on the training set: 0.51791 or 51.791%. Accuracy for predicting e on the testing set: 0.51819 or 51.819%.

2. **Implementation: Full trees**

   In the first set of decision tree experiments, run the ID3 algorithm we saw in class without any depth restrictions. (That is, there are no hyperparameters for this setting.)

   [6 points] Implement the decision tree data structure and the ID3 algorithm for your decision tree (Remember that the decision tree need not be a binary tree!). For debugging your implementation, you can use the previous toy examples like the toy data from Table 1. Discuss what approaches and design choices you had to make for your implementation and what data structures you used. Also explain the organization of your code.

   Your report should include the following information

   (a) [2 points] The root feature that is selected by your algorithm
   Ans) According to my algorithm, the root feature is *spore-print-color*. These functions were used to compute that:

   1) `calc_total_entropy`: This function calculates the total entropy of the dataset, which is needed as a reference for calculating information gain.

   2) `calc_info_gain`: This function calculates the information gain for each feature in the dataset. The feature with the highest information gain will be selected as the root feature.

   3) `find_most_informative_feature`: This function iterates over all features in the dataset and selects the one with the highest information gain as the root feature.

   (b) [2 point] Information gain for the root feature
   Ans) 0.48515 is the information gain for the root spore-print-color. It has used the same functions as mentioned above.

   (c) [2 points] Maximum depth of the tree that your implementation
   Ans) According to my algorithm, the maximum depth is 12. I think it is not correct as it might be taking the value of attributes as nodes as well. The answer should be 6. Functions used are:

   `get_max_depth`: This function recursively traverses the decision tree structure and calculates the depth of each node. It returns the maximum depth found in the tree. The function utilizes recursion to traverse each branch of the tree and incrementally calculate the depth.

   Here's a summary of how the `get_max_depth` function works:

- If the current node is a leaf node (i.e., it's not a dictionary), the function returns 0, indicating a depth of 0 for that node.
- If the current node is a decision node (i.e., it's a dictionary), the function recursively calculates the depth of each branch by calling itself on each branch.
- The function returns the maximum depth among all branches, plus 1 to account for the current node.

This process continues recursively until the entire tree is traversed, and the maximum depth is determined.

This function is essential for understanding the complexity and depth of the decision tree, which can help in assessing its performance and potential for overfitting.

(d) [3 points] Accuracy on the training set
Ans) According to my algorithm, accuracy on the training set is 1.0 or 100%. Functions used are:

`evaluate`: This function evaluates the performance of the decision tree model by predicting labels for each instance in the training dataset and comparing them to the actual labels. It calculates the number of correct predictions and the number of incorrect predictions to compute the accuracy.

Here's a summary of how the `evaluate` function works:

- It iterates over each row in the training dataset.
- For each row, it uses the `predict` function to predict the label based on the decision tree model.
- It compares the predicted label with the actual label in the dataset.
- It counts the number of correct predictions and incorrect predictions.
- Finally, it calculates the accuracy by dividing the number of correct predictions by the total number of predictions.

The `evaluate` function is responsible for quantifying how well the decision tree model performs on the training dataset by providing an accuracy score. This score indicates the proportion of correctly classified instances out of the total number of instances in the dataset.

Also note that fixing the missing values with the most common attribute had no effect. Data was missing in the training dataset.

(e) [5 points] Accuracy on the test set
Ans) According to my algorithm, accuracy on the training set is 1.0 or 100%. Same functions as before.

3. **Limiting Depth**

Next, you will perform 5-fold cross-validation to limit the depth of your decision tree, effectively pruning the tree to avoid overfitting. We have already randomly split the training data into five splits. You should use the 5 cross-validation files for this section, titled `data/CVfolds/foldX.csv` where `X` is a number between 1 and 5 (inclusive).

(a) [15 points] Run 5-fold cross-validation using the specified files. Experiment with depths in the set $\{1, 2, 3, 4, 5, 10, 15\}$, reporting the average cross-validation accuracy and standard deviation for each depth. Explicitly specify which depth should be chosen as the best, and explain why. If a certain depth is not feasible for any reason, your report should explain why.

Ans) As we can see, depth 15 has the best accuracy, which is 95.895%, therefore we would select it as the best depth.

Table 5: Depth vs Accuracy with Standard Deviation

Depth	Accuracy (%)	Standard Deviation
1	0.0	0.0
2	0.0	0.0
3	2.557	2.424
4	2.557	2.424
5	67.151	18.815
10	94.686	5.541
15	95.895	4.872

Replacing Missing Values in Dataset:

The function `replace_missing_stalk_root` reads a CSV file into a DataFrame using pandas. It identifies missing values (denoted as '?') in the 'stalk-root' column and replaces them with the most common value found in that column. The modified DataFrame is then saved back to the original CSV file, effectively replacing the missing values.

Calculating Information Gain:

Several functions (`calc_total_entropy`, `calc_entropy`, `calc_info_gain`, `find_most_informati` are defined to calculate information gain for decision tree construction using the ID3 algorithm. These functions compute entropy and information gain based on the distribution of classes in the dataset.

Constructing Decision Tree:

The functions `generate_sub_tree` and `make_tree` are responsible for constructing the decision tree recursively. They iteratively find the most informative feature, split the dataset based on its values, and create subtrees for each branch.

Determining Maximum Depth:

The function `get_max_depth` calculates the maximum depth of the decision tree. It recursively traverses the tree structure to determine the depth, considering each branch as a level.

Pruning the Tree:

The function `prune_tree` prunes the decision tree to limit its maximum depth. It replaces subtrees beyond the specified maximum depth with a placeholder, effectively reducing the complexity of the tree.

Predicting and Evaluating:

The function `predict` predicts the class label for a given instance using the constructed decision tree. The function `evaluate` evaluates the accuracy of the decision tree model on a test dataset by comparing predicted labels with actual labels.

Performing Cross-Validation:

The code performs 5-fold cross-validation with different maximum depths specified in the `max_depths` list. For each maximum depth value, it trains decision tree models on training data from different folds and evaluates their performance on respective test data. It calculates the average accuracy and standard deviation of cross-validation accuracy scores for each maximum depth.

Even though we are getting 15 as the best depth, the issue is it might not be feasible as the maximum depth we got was 12.

(b) [4 points] Explain briefly how you implemented the depth limit functionality in your code.

Ans) Determining Maximum Depth:

The function `get_max_depth` calculates the maximum depth of the decision tree. It recursively traverses the tree structure to determine the depth, considering each branch as a level.

(c) [15 points] Using the depth with the greatest cross-validation accuracy from your experiments: train your decision tree on the `data/train.csv` file. Report the accuracy of your decision tree on the `data/test.csv` file.

Ans) According to my algorithm, accuracy for the decision tree with depth limited at 15 is 100% when trained on the training data and when tested on the testing and training data.

Information Gain Calculation:

Functions `calc_total_entropy`, `calc_entropy`, and `calc_info_gain` are defined to calculate the entropy and information gain of features in the dataset. These calculations are crucial for determining the best features to split the data on during the construction of the decision tree.

Finding the Most Informative Feature:

The function `find_most_informative_feature` identifies the feature that provides the highest information gain, i.e., the most informative feature for splitting the dataset.

Generating Subtrees:

The function `generate_sub_tree` creates subtrees based on the selected feature, splitting the dataset into subsets corresponding to different feature values.

Constructing the Decision Tree:

The function `make_tree` recursively constructs the decision tree by selecting features, generating subtrees, and assigning them to appropriate nodes based on the information gain.

ID3 Algorithm:

The function `id3` implements the ID3 algorithm, which builds a decision tree based on the provided training data, label, and optionally, a maximum depth

parameter for tree pruning. It returns the constructed tree, the root feature of the tree, the information gain at the root, and the maximum depth of the tree.

Tree Pruning:

The functions `get_max_depth` and `prune_tree` are used for tree pruning. `get_max_depth` calculates the maximum depth of the tree, while `prune_tree` prunes the tree to the specified maximum depth by replacing subtrees with placeholders beyond that depth.

Prediction and Evaluation:

The function `predict` predicts the class label for a given instance using the constructed decision tree. The function `evaluate` evaluates the accuracy of the decision tree model on test data by comparing predicted labels with actual labels.

Cross-Validation and Performance Evaluation:

The code performs 5-fold cross-validation with a fixed depth specified for the decision tree. It evaluates the model's accuracy on both test and training data and calculates the average accuracy and standard deviation of cross-validation accuracy scores for both datasets.

(d) [5 points] Discuss the performance of the depth limited tree as compared to the full decision tree. Do you think limiting depth is a good idea? Why?
Ans) According to the accuracies, a full decision tree is better than depth-limiting trees. Generally, a full decision tree tends to have higher accuracy on the training data compared to a depth-limited tree. This is because a full decision tree can perfectly fit the training data, potentially resulting in overfitting. In contrast, a depth-limited tree may sacrifice some accuracy on the training data but might generalize better to unseen data due to its simpler structure.

But this is just one metric being used to define if it is better or not. However, depth-limiting trees are less prone to overfitting compared to full decision trees.

Depth limitation helps prevent overfitting by restricting the tree's ability to memorize noise or outliers in the training data. It encourages the model to capture more generalizable patterns rather than fitting the training data too closely.

Experiment Submission Guidelines

1. The report should detail your experiments. For each step, explain in no more than a paragraph or so how your implementation works. You may provide the results for the final step as a table or a graph.

2. *Your code should run on the CADE machines.* You should include a shell script, **run.sh**, that will execute your code in the CADE environment. For each experiment, your code must print your results in the following order:

   (a) Entropy of the data

   (b) Best feature and its information gain

   (c) Cross-validation accuracies for each fold (for limiting depth setting)

(d) Best depth (for the limiting depth setting)

(e) Accuracy of the trained classifier on the training set (For the limiting depth setting, this would be for the tree with the best depth)

(f) Accuracy of the trained classifier on the test set (For the limiting depth setting, this would be for the tree with the best depth)

Without these details, you will lose points for your implementation.

You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.

3. Please do *not* hand in binary files! We will not grade binary submissions.

3 CS 6350 only: Decision Trees with Attribute Costs

[10 points] Sometimes, we may encounter situations where the features in our learning problem are associated with costs. For example, if we are building a classifier in a medical scenario, features may correspond to the results of different tests that are performed on a patient. Some tests may be inexpensive (or inflict no harm), such as measuring the patient's body temperature or weight. Some other tests may be expensive (or may cause discomfort to the patient), such as blood tests or radiographs.

In this question, we will explore the problem of learning decision trees in such a scenario. We prefer decision trees that use features associated with low costs at the top of the tree and only use higher cost features if needed at the bottom of the trees. In order to impose this preference, we can modify the information gain heuristic that selects attributes at the root of a tree to penalize costly attributes.

In this question, we will explore different such variations. Suppose we denote $Gain(S, A)$ as the information gain of an attribute A for a dataset S (using the original version of information from ID3). Let $Cost(A)$ denote the cost of the attribute A. We can define two cost-sensitive information gain criteria for attributes as:

1. $Gain_T(S, A) = \frac{Gain(S,A)^2}{Cost(A)}$
2. $Gain_N(S, A) = \frac{2^{Gain(S,A)}-1}{\sqrt{Cost(A)+1}}$

In both cases, note that attributes with higher costs are penalized and so will get chosen only if the information gain is really high.

To evaluate these two methods for root selection, we will use the following training set:

Shape	Color	Size	Material	Label
square	red	big	metal	+
square	blue	small	plastic	+
triangle	yellow	medium	metal	+
triangle	pink	big	leather	-
square	pink	medium	leather	-
circle	red	small	plastic	-
circle	blue	small	metal	-
ellipse	yellow	small	plastic	-
ellipse	blue	big	leather	+
ellipse	pink	medium	wood	+
circle	blue	big	wood	+
triangle	blue	medium	plastic	+

Suppose we know the following costs of the attributes:

Attribute	Cost
Shape	10
Color	30
Size	50
Material	100

1. [8 points] Compute the modified gains $Gain_T$ and $Gain_S$ for each attribute using these costs. Fill in your results in the table below. (upto 3 decimal places)
   Ans) Entropy of whole data: $T = \frac{7}{12}$ and $F = \frac{5}{12}$
   $S = -\frac{7}{12}\log_2\left(\frac{7}{12}\right) - \frac{5}{12}\log_2\left(\frac{5}{12}\right) = 0.583(0.778) + 0.416(1.265) = 0.453 + 0.526 = 0.979$

   Let us calculate Gain for attribute Shape

   For attribute value square ($\frac{3}{12}$): $T = \frac{2}{3}$ and $F = \frac{1}{3}$
   Entropy(S)$=-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) = 0.918$

   For attribute value triangle ($\frac{3}{12}$): $T = \frac{2}{3}$ and $F = \frac{1}{3}$
   Entropy(S)$=-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) = 0.918$

   For attribute value circle ($\frac{3}{12}$): $T = \frac{2}{3}$ and $F = \frac{1}{3}$
   Entropy(S)$=-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) = 0.918$

   For attribute value ellipse ($\frac{3}{12}$): $T = \frac{2}{3}$ and $F = \frac{1}{3}$
   Entropy(S)$=-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) = 0.918$

   $\Sigma S = \frac{1}{4} \times 0.918 \times 4 = 0.918$

   Gain $= 0.979 - 0.918 = 0.061$
   $Gain(S, A)_T = \frac{0.061^2}{10} = 0.00037$

   $Gain(S, A)_S = \frac{2^{0.061}-1}{\sqrt{11}} = \frac{1.043-1}{\sqrt{11}} = \frac{0.043}{3.316} = 0.012$

Attribute	$Gain_T$	$Gain_S$
Shape	0.00037	0.012
Color	0.00047	0.015
Size	0.00056	0.017
Material	0.00035	0.0138

2. [2 points] For each variant of gain, which feature would you choose as the root?
   Ans) For $Gain_T$, we will use Size as the root attribute, for $Gain_S$ we will use Size as the root attribute as well since both have the highest value respectively in each set.

(You may modify your code from the experiments to calculate these values instead of doing so by hand.)