Decision Trees

TJ Machine Learning[*]

October 2020

1 Introduction

Decision trees are powerful and interpretable classifiers that mirror human decisions unlike many other classifiers in supervised machine learning and are the building blocks of random forests.

2 Definition

In essence, decision trees ask a series of true/false questions to narrow down what class a particular sample belongs to. Here is an example of a decision tree one might use in real life to decide upon an activity on a given day:

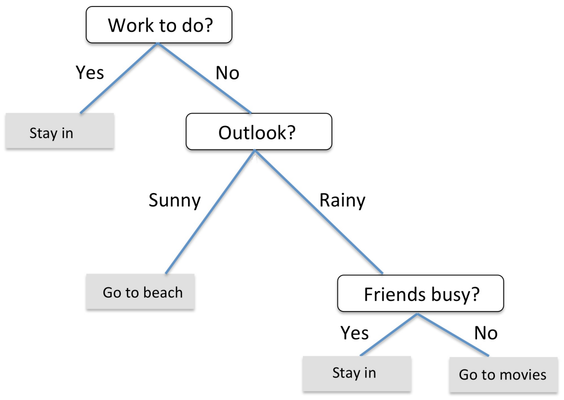


Figure 1: Real Life Decision Tree

Although this figure asks categorical variable-based questions, we can ask numerical-based questions like "$x_1 < 5$?" when the features are continuous. To build our tree, we start at the root node and ask a question that splits the data

[*]Adapted from lectures of Nikhil Sardana Vinay Bhaip

based on a feature such that the information gain is maximized. We continuously do this for each node until the decision tree can classify all the training data. (Note that in practice this leads to overfitting, so the tree is usually pruned, i.e. a limit on the depth of the tree is set.)

2.1 Information Gain

We split each node on the feature and threshold that yields the most information gain. The formula for information gain in a binary decision tree is as follows:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

D_p is the dataset of the parent node (the node which we are splitting), f is the feature of the dataset which we are splitting on, N_p is the total number of samples in the parent node, N_{left} and N_{right} are the number of samples in the datasets of the left child node and right child node respectively, and I is the impurity measure. A node is pure if all samples in its dataset belong to the same class and is most impure when an equal number of samples belong to each class. Essentially, information gain calculates the difference between the impurity of the parent node and the impurity of the child nodes.

One commonly used measure of impurity is Gini impurity:

$$I_G(i) = 1 - \sum_{k=1}^{c} p(k|i)^2$$

$p(k|i)$ is the proportion of samples of class k to the total number of samples in the dataset of the i^{th} node. The impurity is maximized when the classes of the node are perfectly mixed (for this example, consider a situation in which there are 2 classes, meaning c = 2):

$$1 - \sum_{k=1}^{c} 0.5^2 = 0.5$$

An alternative impurity measure is entropy, which is defined as:

$$-\sum_{k=1}^{c} p(k|i) log_2 p(k|i)$$

Note that this function has a maximum of 1.0, not 0.5. In practice, Gini impurity and entropy yield similar results, so it is more useful to test different pruning cut-offs rather than to evaluate trees with different impurity criteria.

To decide on a split for a specific node, we will search for the feature and the threshold (e.g. “petal length < 2.45 cm” for a flower classifier) that maximizes the information gain. We can choose the best threshold for a feature from the feature values in the training data or from the averages of every pair of feature values in the training set. Another method is to select the best threshold from the quartiles (20%, 40%, 60%, and 80% values) of the feature set.

Here is the pseudocode for determining the best split:

Algorithm 1 Best Split

```
1:  $IG \leftarrow 0$
2:  for each feature do
3:      for each threshold do
4:          $pot\_left,\ pot\_right \leftarrow$ split$(parent,\ feature,\ threshold)$
5:          $pot\_ig \leftarrow$ information_gain$(parent,\ pot\_left,\ pot\_right)$
6:          if $pot\_ig > IG$ then
7:              $left \leftarrow pot\_left$
8:              $right \leftarrow pot\_right$
9:              $IG \leftarrow pot\_ig$
10:         end if
11:     end for
12: end for
13: return $left,\ right$
```

Algorithm 2 Split

```
1:  function SPLIT($dataset,\ feature,\ threshold$)
2:      Initialize $left$ and $right$ lists
3:      for each data point in dataset do
4:          if $feature$ of data point $<$ $threshold$ then
5:              append data point to $left$
6:          else
7:              append data point to $right$
8:          end if
9:      end for
10:     return $left,\ right$
11: end function
```

Algorithm 3 Gini Impurity

```
1: function SPLIT($dataset$)
2:     $sum \leftarrow 0$
3:     for each class label $c$ do
4:         $ratio \leftarrow$ (number of class labels $c$)/size of dataset
5:         $sum \leftarrow sum + ratio * ratio$
6:     end for
7:     return 1 - $sum$
8: end function
```

3 Practice Problems

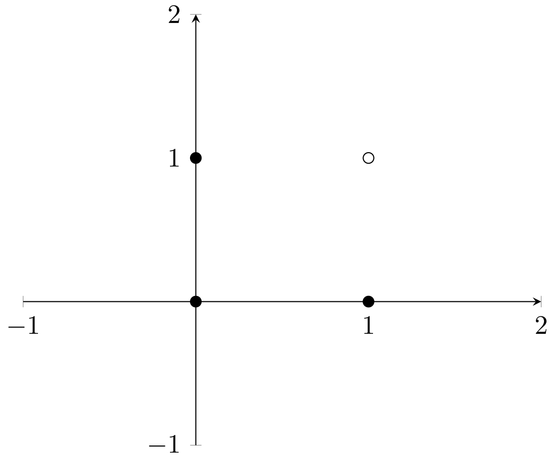
Consider the following dataset:

x1	x2	x3	y
0	1	0	-1
1	0	0	+1
0	1	1	+1
0	0	1	-1

1. What feature will we split on at the root of our decision tree, and what will our information gain be from splitting on that feature using the Gini impurity measure?
2. Build a decision tree using the dataset. What is the depth of the tree?
3. What will the decision tree classify a data point with the features x1 = 0, x2 = 0, and x3 = 0 as (y = -1 or y = +1)?

x1	x2	y
0	0	+1
0	1	+1
1	0	+1
1	1	-1

It's often helpful to visualize your data, even if it seems simple. Let the ● = +1 and ○ = −1.



4. What will the information gain be after the first split in the above data set with the Gini impurity measure? With entropy as the impurity measure?
5. What is the depth of the final decision tree?

4 Example

Decision trees are created by splitting on a threshold of a feature which results in the most information gain, and recursively continuing this process for each of the children.

Let us review Information Gain with an example. Remember, the equation is:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

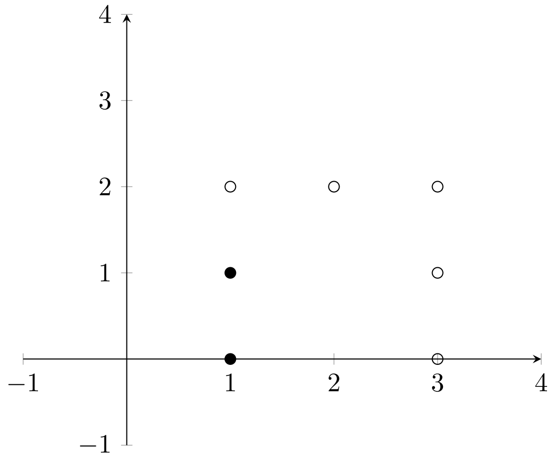
We will use the Gini Impurity measure, just like last time:

$$I(i) = 1 - \sum_{k=1}^{c} p(k|i)^2$$

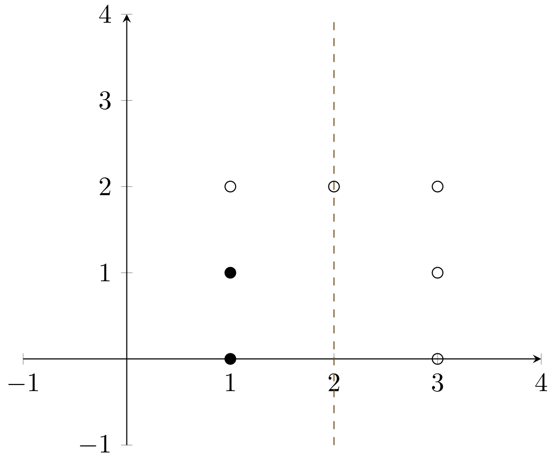
Consider the following data. We have two features, $x1$ and $x2$. Our label is y.

x1	x2	y
1	0	0
1	1	0
1	2	1
2	2	1
3	0	1
3	1	1
3	2	1

Let's graph this data. Let the ● = 0 and ○ = 1.

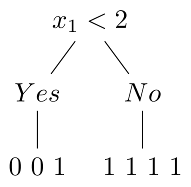


We want to draw a vertical or horizontal line that best separates the white dots from the black dots (ie has the greatest information gain). Since we are restricted to the coordinates of data points, a human will look at the graph and choose:



Since we are classifying things based on whether they are less than the threshold ($x < 2$), the point in the middle goes with the right side.

The decision tree that goes with this is:



We can calculate the information gain.

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

$$N_{left} = 3$$

$$N_{right} = 4$$

$$N_p = 7$$

Now we need the impurities.

$$I(i) = 1 - \sum_{k=1}^{c} p(k|i)^2$$

For the parent, we calculate the impurity of the entire dataset, which is:

$$I(D_p) = 1 - \left(\frac{2}{7}\right)^2 - \left(\frac{5}{7}\right)^2 = 1 - \frac{4}{49} - \frac{25}{49} = \frac{49 - 4 - 25}{49} = \frac{20}{49}$$

And for the children:

$$I(D_{left}) = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 1 - \frac{4}{9} - \frac{1}{9} = \frac{4}{9}$$

$$I(D_{right}) = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2 = 1 - 1 = 0$$

Note that the most pure data $I(i) = 0$ is all one type.

Finally, we get the information gain:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

$$IG(D_p, f) = \frac{20}{49} - \frac{3}{7} * \frac{4}{9} - \frac{4}{7} * 0 = \frac{20}{49} - \frac{12}{63} = \frac{1}{7} * \left(\frac{20}{7} - \frac{12}{9}\right) = \frac{1}{7} * \left(\frac{180}{63} - \frac{84}{63}\right)$$

$$IG(D_p, f) = \frac{1}{7} * \frac{96}{63} = \frac{96}{441}$$

5 Data Structure

Let's go back to the example from Section 2.1.

x1	x2	y
1	0	0
1	1	0
1	2	1
2	2	1
3	0	1
3	1	1
3	2	1

How will we represent this data in our code? It's simple! We'll use a list of lists, with each column a feature.

$$\begin{bmatrix} [1,0,0] \\ [1,1,0] \\ [1,2,1] \\ [2,2,1] \\ [3,0,1] \\ [3,1,1] \\ [3,2,1] \end{bmatrix}$$

Alternatively, you can make the labels their own list. Our sample I/O code has it this way:

$$Features = \begin{bmatrix} [1,0] \\ [1,1] \\ [1,2] \\ [2,2] \\ [3,0] \\ [3,1] \\ [3,2] \end{bmatrix} \qquad Labels = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

To find the best information gain, we loop through each of the features (columns), and each of the thresholds, splitting the matrix and calculating the information gain. For example, in the previous example, we split on $x1 < 2$, so our left child matrix would be:

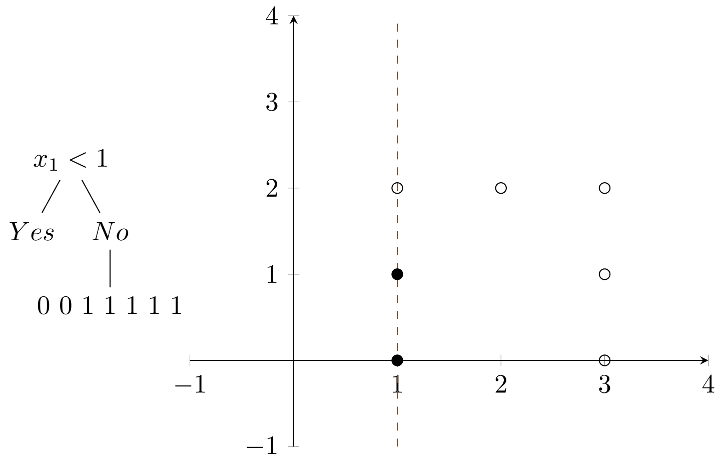
$$\begin{bmatrix} [1,0,0] \\ [1,1,0] \\ [1,2,1] \end{bmatrix}$$

and the right child matrix:

$$\begin{bmatrix} [2,2,1] \\ [3,0,1] \\ [3,1,1] \\ [3,2,1] \end{bmatrix}$$

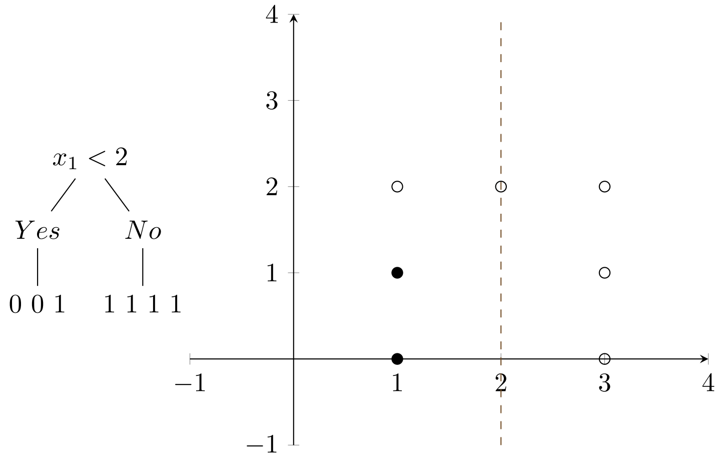
6 Summary

To ensure complete understanding and summarize the topics we've covered, lets loop through all the features and each of the possible thresholds, splitting the matrix for each one.



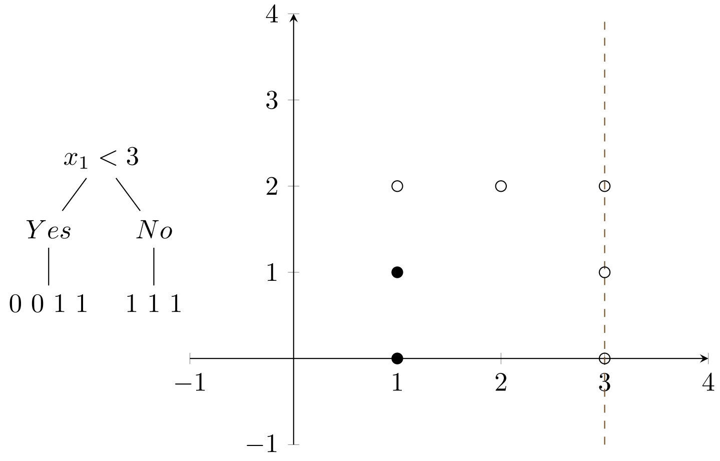
$$D_{left} = []$$

$$D_{right} = \begin{bmatrix}[1,0,0]\\ [1,1,0]\\ [1,2,1]\\ [2,2,1]\\ [3,0,1]\\ [3,1,1]\\ [3,2,1]\end{bmatrix}$$



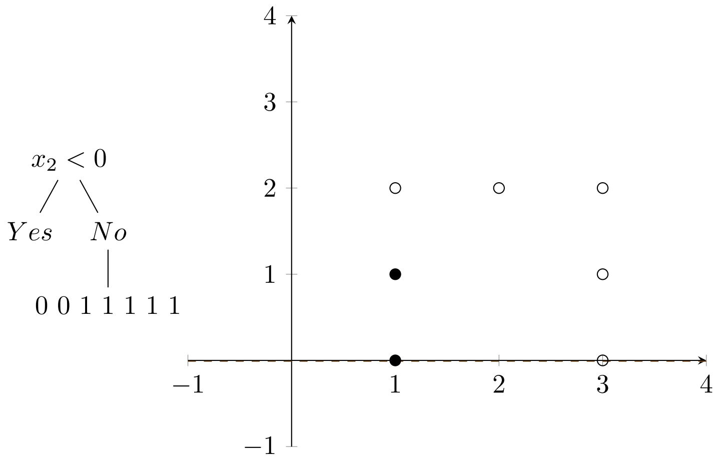
$$D_{left} = \begin{bmatrix}[1,0,0]\\ [1,1,0]\\ [1,2,1]\end{bmatrix}$$

$$D_{right} = \begin{bmatrix}[2,2,1]\\ [3,0,1]\\ [3,1,1]\\ [3,2,1]\end{bmatrix}$$



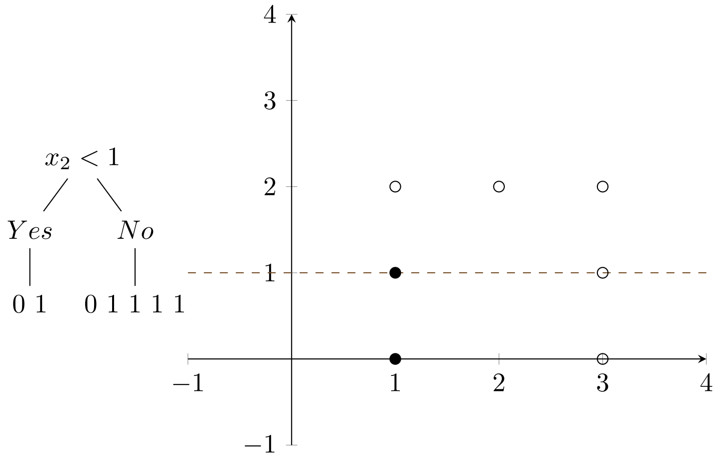
$$D_{left} = \begin{bmatrix}[1,0,0]\\ [1,1,0]\\ [1,2,1]\\ [2,2,1]\end{bmatrix}$$

$$D_{right} = \begin{bmatrix}[3,0,1]\\ [3,1,1]\\ [3,2,1]\end{bmatrix}$$



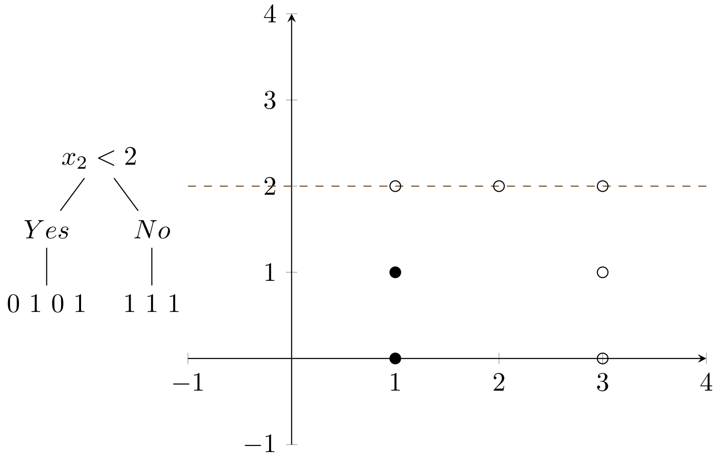
$$D_{left} = []$$

$$D_{right} = \begin{bmatrix} [1,0,0] \\ [1,1,0] \\ [1,2,1] \\ [2,2,1] \\ [3,0,1] \\ [3,1,1] \\ [3,2,1] \end{bmatrix}$$



$$D_{left} = \begin{bmatrix} [1,0,0] \\ [3,0,1] \end{bmatrix}$$

$$D_{right} = \begin{bmatrix} [1,1,0] \\ [1,2,1] \\ [2,2,1] \\ [3,1,1] \\ [3,2,1] \end{bmatrix}$$



$$D_{left} = \begin{bmatrix} [1,0,0] \\ [3,0,1] \\ [1,1,0] \\ [3,1,1] \end{bmatrix}$$

$$D_{right} = \begin{bmatrix} [1,2,1] \\ [2,2,1] \\ [3,2,1] \end{bmatrix}$$

7 The Full Tree

Let's build an entire decision tree. First, consider the following data.

x1	x2	y
0	0	0
1	0	0
1	1	0
0	1	0
1	3	1
0	3	1
2	2	1
4	1	1
3	1	1
3	2	1

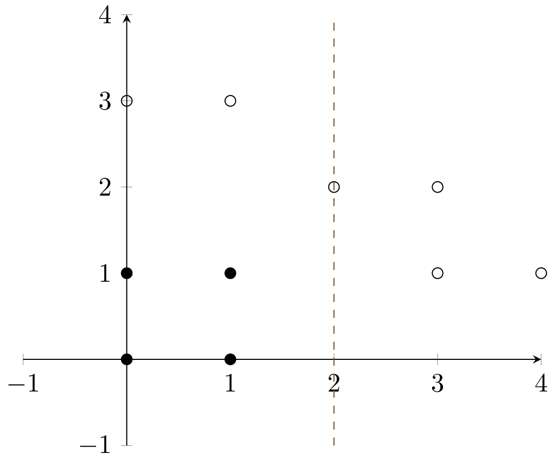
We graph it and calculate that $x1 < 2$ gives us the greatest information gain.

$$I(D_p) = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2 = 1 - \frac{16}{100} - \frac{36}{100} = \frac{100-52}{100} = \frac{48}{100} = \frac{12}{25}$$

$$I(D_{left}) = 1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2 = 1 - \frac{16}{36} - \frac{4}{36} = \frac{16}{36} = \frac{4}{9}$$

$$I(D_{right}) = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2 = 1 - 1 = 0$$

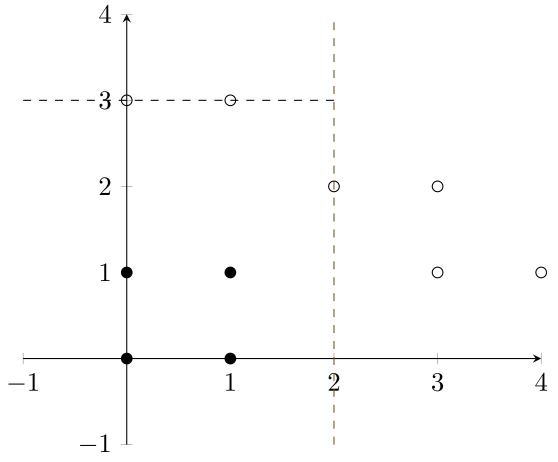
$$IG(D_p, f) = \frac{12}{25} - \frac{6}{10} * \frac{4}{9} - \frac{4}{10} * 0 = \frac{12}{25} - \frac{4}{15} = \frac{36}{75} - \frac{20}{75} = \frac{16}{75}$$



The dataset is split:

$$D_{left} = \begin{bmatrix} [0,0,0] \\ [1,0,0] \\ [1,1,0] \\ [0,1,0] \\ [0,3,1] \\ [1,3,1] \end{bmatrix} \qquad D_{right} = \begin{bmatrix} [2,2,1] \\ [3,1,1] \\ [3,2,1] \\ [4,1,1] \end{bmatrix}$$

Now we make our next decision. We start with the left child, and it is obvious that $x2 < 3$ generates the greatest information gain. (Again, our thresholds are restricted to coordinates in our dataset).



The data is now split

$$D_{left} = \begin{bmatrix} [0,0,0] \\ [1,0,0] \\ [1,1,0] \\ [0,1,0] \end{bmatrix}$$

$$D_{right} = \begin{bmatrix} [0,3,1] \\ [1,3,1] \end{bmatrix}$$

We can now calculate the information gain. We know the impurity of the parent dataset, which was the impurity of the left child on the last split.

$$I(D_p) = \frac{4}{9}$$

Now, we calculate the children impurities to find our information gain:

$$I(D_{left}) = 0$$

$$I(D_{right}) = 0$$

$$IG(D_p, f) = \frac{4}{9} - \frac{4}{6} * 0 - \frac{2}{6} * 0 = \frac{4}{9}$$

All our data has been classified correctly, so the decision tree is done. We will leave it as an exercise to the reader to draw the decision tree.

8 Code

Here's how we recommend you structure your decision tree code for the competition. It should be noted: do not split recursively until all data is pure. You will overfit the testing data. Overfitting means that we train a model that memorizes the training data, rather than finding the generalized patterns embedded. When you overfit your data, the training accuracy will be high, but the testing accuracy will be low. Instead, set a baseline for your information gain. If the maximum information gain is less than, say, 0.1, then stop the recursion and make it a leaf of the tree. Play with this baseline to see how your training and testing accuracy change. Remember: your goal is the greatest testing accuracy.

```
def calculateImpurity(matrix):
    #do calculations
    return impurity

def splitmatrix(matrix, feature, threshold):

    #split matrix into leftchild, rightchild
    return leftchild, rightchild

def informationGain(parent, leftchild, rightchild):
    #do calculations
    return infoGain

def bestsplit(parent, depth):
    for each feature in parent:
        for each threshold in feature:
            lchild, rchild = splitmatrix(parent, feature, threshold)
            igain = informationGain(parent, lchild, rchild)

            #if igain is the greatest:
            #    save the leftchild, rightchild, feature, threshold

    #print this node (feature, threshold) of the decision tree
    # hint: use depth to indent
    #recur on leftchild
    #recur on rightchild
```