Introduction to Data Science

DECISION TREES

BRIAN D'ALESSANDRO

TARGETED EXPLORATION

If we have a Target variable in our dataset, we often wish to:

1. Determine whether a variable contains important information about the target variable (in other words, does a given variable reduce the uncertainty about the target variable).

2. Obtain a selection of the variables that are best suited for predicting the target variables.

3. Rank each variable on its ability to predict the target variable.

INFORMATION THEORY

We'll run targeted exploration using some fundamental tools from information theory, and then build these up as the foundation for Decision Trees.

1. **Entropy** – the average amount of information that is encoded by a random variable X.

2. **Conditional Entropy** – Given we know X, how much extra information is needed to encode Y.

3. **Information Gain** – How much new information do we gain on Y by conditioning on X.

ENTROPY

This comes from Information Theory, and is a measure of the unpredictability or uncertainty of information content.

Certainty => Closer to 0% or 100% sure that something will happen.

Entropy is fundamental concept used for Decision Trees and Feature Selection/Importance, and is a tool that can satisfy the demands stated on previous slide.

ENTROPY

More formally…

X is a random variable with {x1, x2 … xN} possible values, the entropy is the expected "information" of an event, where "information" is defined as –log(P(X)) and is measured in 'bits.' Intuitively, Entropy is one way to express the variance, or spread of a discrete random variable.

$$H(X) = E[I(X)] = E[-\ln(P(X))].$$

$$H(X) = \sum_i P(x_i)\, I(x_i) = -\sum_i P(x_i) \log_b P(x_i)$$

Equation source:
http://en.wikipedia.org/wiki/Entropy_(information_theory)

ENTROPY

Example: Coin Flip, X is a random variable with possible values {H,T}

Entropy of a coin: **- [P(H)*log2(P(H)) + P(T)*log2(P(T))]**

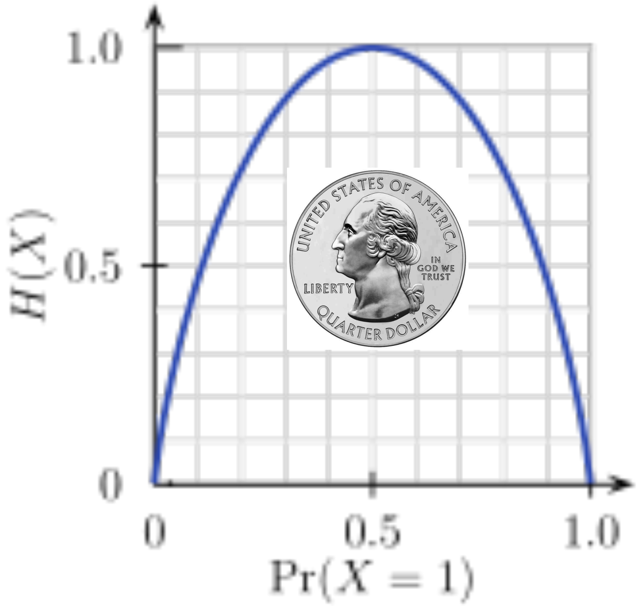


Image source: http://en.wikipedia.org/wiki/Entropy_(information_theory)

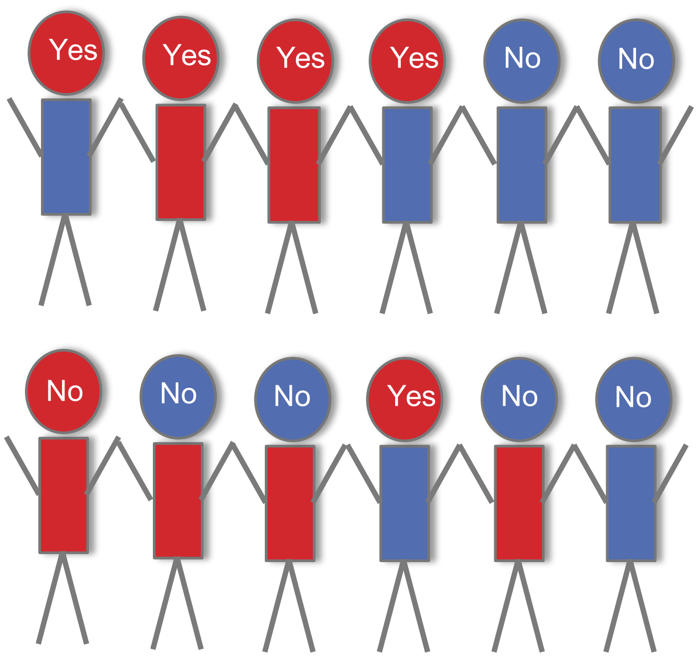
Fair Coin:
H(X)=-[0.5*Log2(0.5)+0.5*Log2(0.5)] = 1

Single Outcome Coin:
H(X)=-[1*Log2(1)] = 0

Weighted Coin (P(H)=0.2):
H(X)=-[0.2*Log2(0.2)+0.8*Log2(0.8)] = .72

CONDITIONAL ENTROPY

We want to explore whether the attributes head and body colors give us more information about our target variable (yes/no). In information theory, the Conditional entropy expresses how much additional information we need to encode Y if we know X.



We use the **Conditional Entropy**

$$H(Y|X) \equiv \sum_{x \in \mathcal{X}} p(x) H(Y|X=x)$$

Where,

- Y is binary target variable
- X is the attribute
- x is a value of an attribute
- p(x) is the likelihood X=x
- H(Y|X=x) is the entropy of Y where X=x

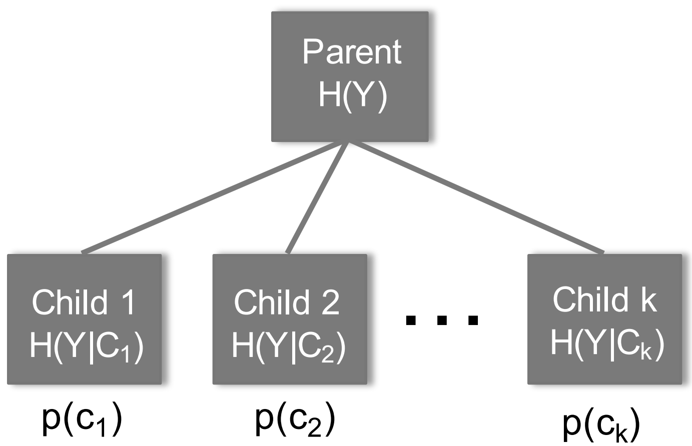
INFORMATION GAIN

The Information Gain of an attribute measures the change in conditional entropy if we split the data by a given attribute.

$$IG = H(Y|Parent) - \sum_{c \in C} p(c) H(Y|c)$$

where $$H(Y|c) = -\sum_{y \in Y} p(y|c) \log_2 p(y|c)$$

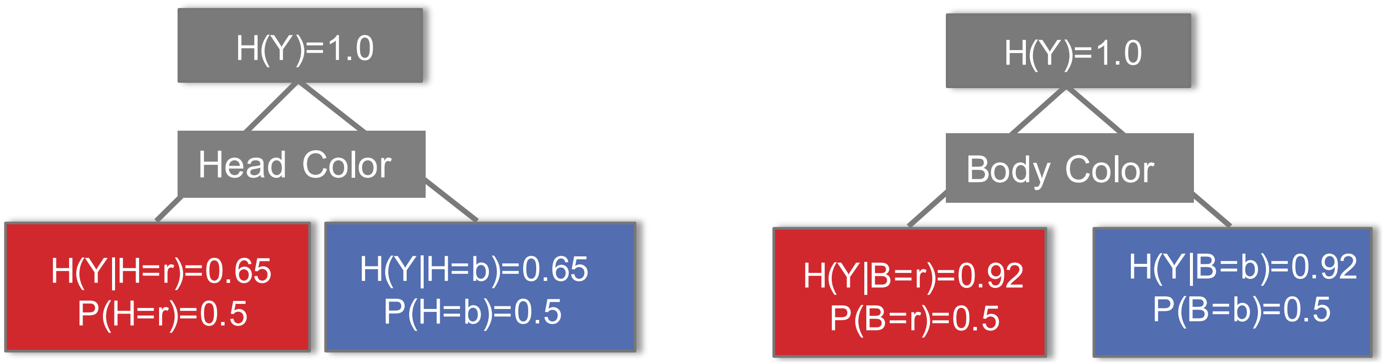
Information Gain tells us how pure can we make segments with respect to the target variable if we split the population by a certain attribute.



Each child node *c* is the result of splitting the data by the values of a certain attribute, and has its own associated Entropy H(Y|c) as well as a split proportion p(c).

USING INFORMATION GAIN

With Information Gain we can now define a way to compare features on their ability to segment the data in a way that reduces the uncertainty of our target variable.



$$IG(Head)=1-(0.5*0.65+0.5*0.65)=0.35$$

$$IG(Body)=1-(0.5*0.92+0.5*0.92)=0.08$$

We get a higher IG when splitting on Head

RECAP

With Entropy, Conditional Entropy and Information gain,
we now have tools that can help us:

1. Quantify the purity of a segment with respect to a Target variable

2. Measure and rank individual features by their ability to split the data in a way that reduces uncertainty about the target variable.

Next we'll use these tools to build a classifier

C4.5 – TOWARDS A DECISION TREE

Several algorithms exist that return a decision tree structure, but we'll focus on C4.5, which utilizes the information theoretic tools learned so far.

Given a dataset $D=\langle X,Y\rangle$, where Y is a target variable and $X=\langle X_1,X_2,..X_k\rangle$ is a k-dimensional vector of predictor variables.

C4.5 pseudocode

1. For each feature x_i in X:
- Compute split that produces highest Information Gain
- Record Information Gain

2. Let X^{max} be the feature with the highest Information Gain

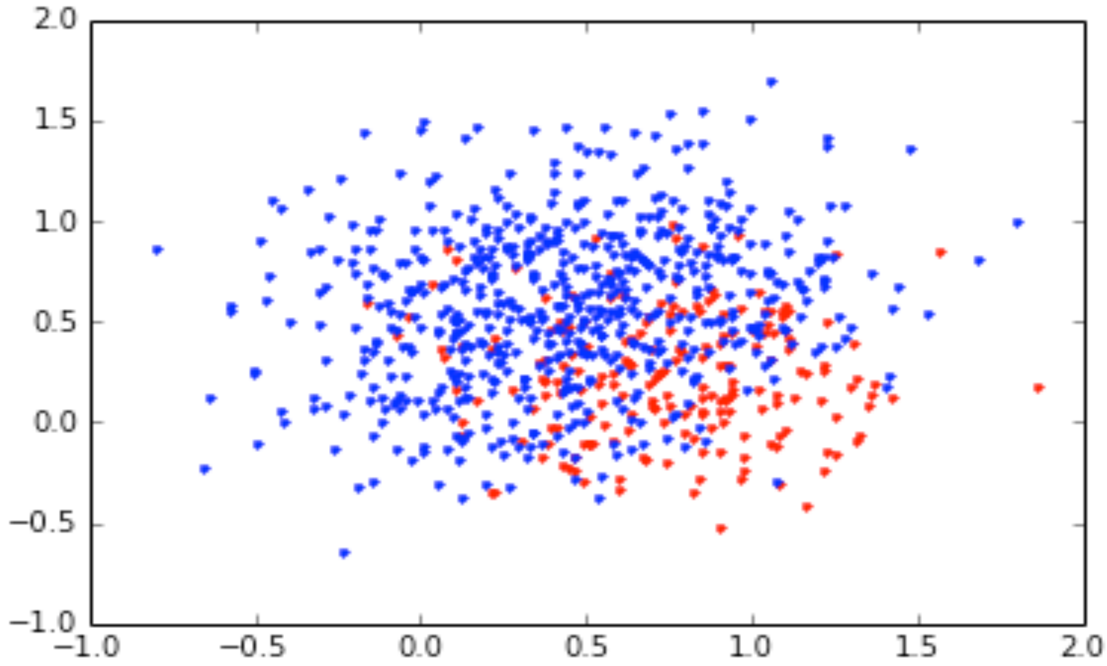
3. Create child nodes that split on the optimal splitting of X^{max}

4. Recurse on each child node from step 3 on the remaining features.

LETS BUILD A TREE

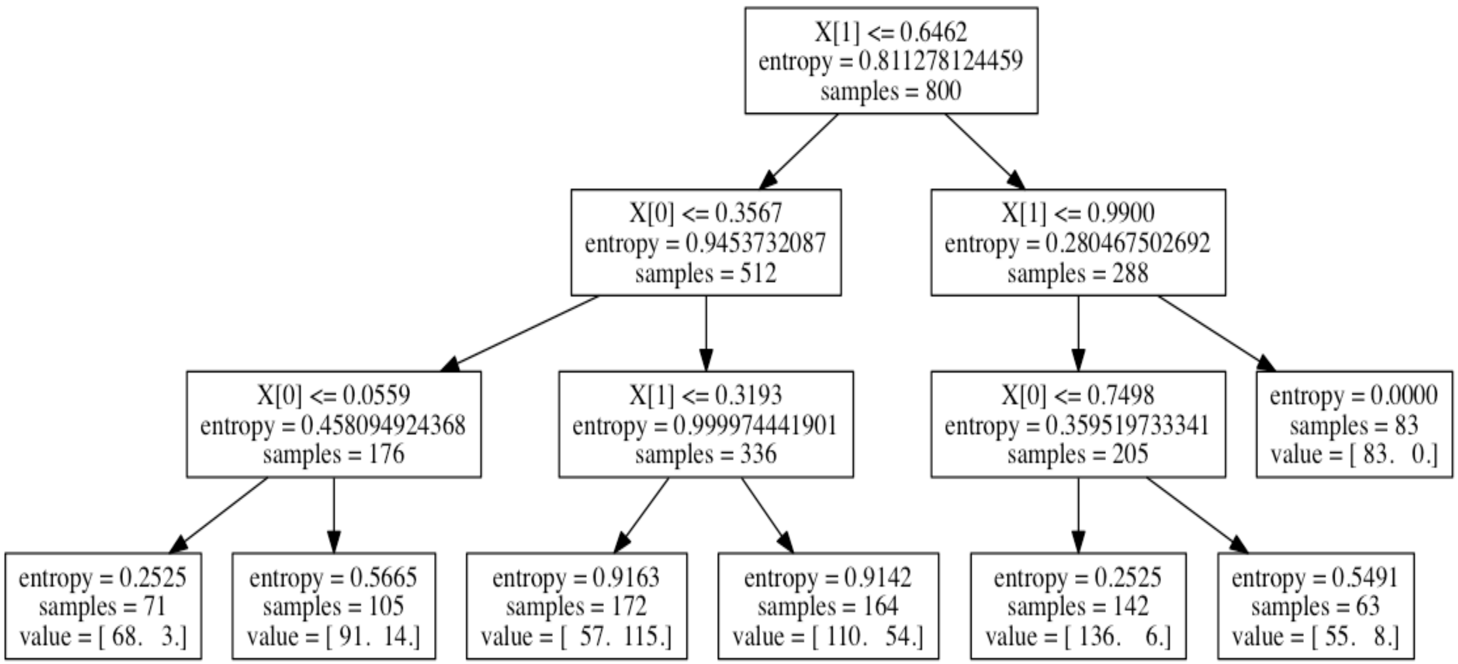
We have data generated as a mixture of 4 different bivariate distributions with means: mu=[[0.25,0.75],[0.75,0.75],[0.75,0.25],[0.25,0.25]], and all with the same covariance structure.

The distribution with mean [0.75,0.25] was given a 'red' label.



TREES WITH SCI-KIT LEARN

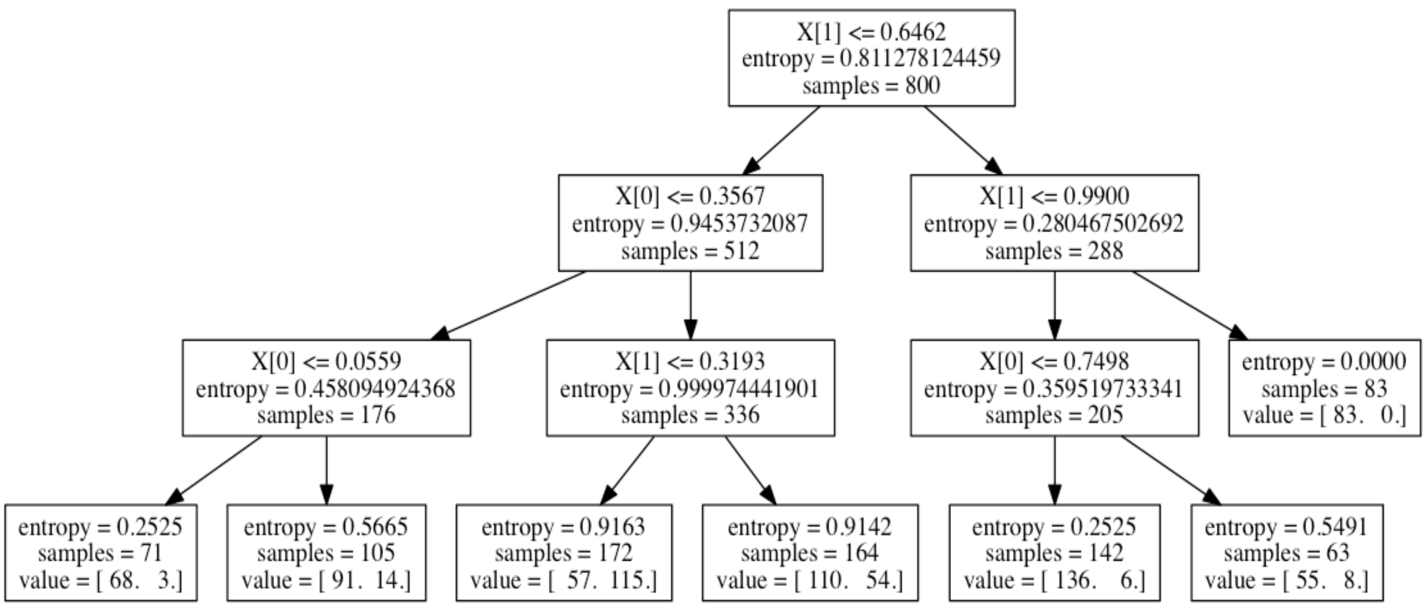
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(args)
clf = clf.fit(X,y)
```



Note: with a lot of features and a lot of data, trees can be incredibly large. Its not always worth it to try and visualize them.

UNDERSTANDING THE TREE

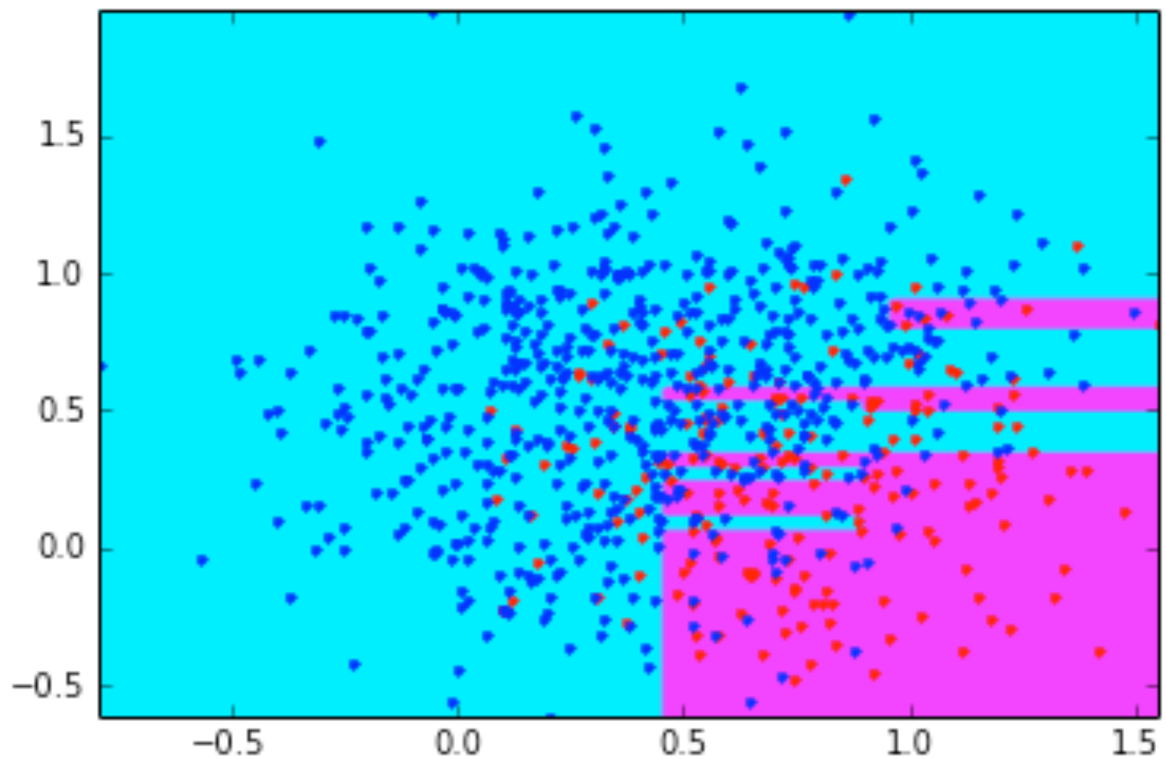
- Each parent node references the feature chosen by the splitting algorithm.
- The node specifies a boolean condition
  - If True, then move to the right
  - Else move to the left
- The final child nodes (leaves) show the distribution of the target variable at that node
- The prediction by averaging the target variable at each leaf.



PARTITIONING X SPACE

We can think of the Decision Tree as a rectangular partitioning of the feature space. Again, this is done with a series of boolean conditionals.

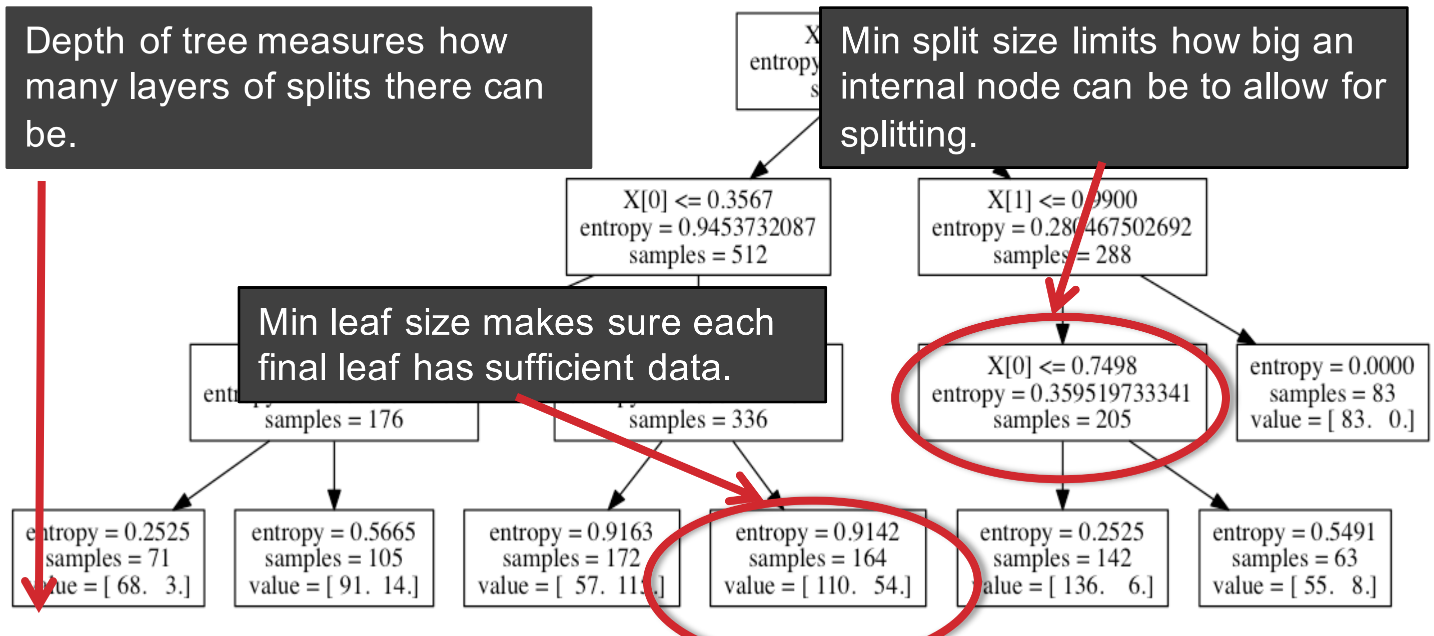
For classification tasks, the partition determines the predicted class. This chart shows the decision boundaries with the training data overlayed.



CONTROLLING COMPLEXITY

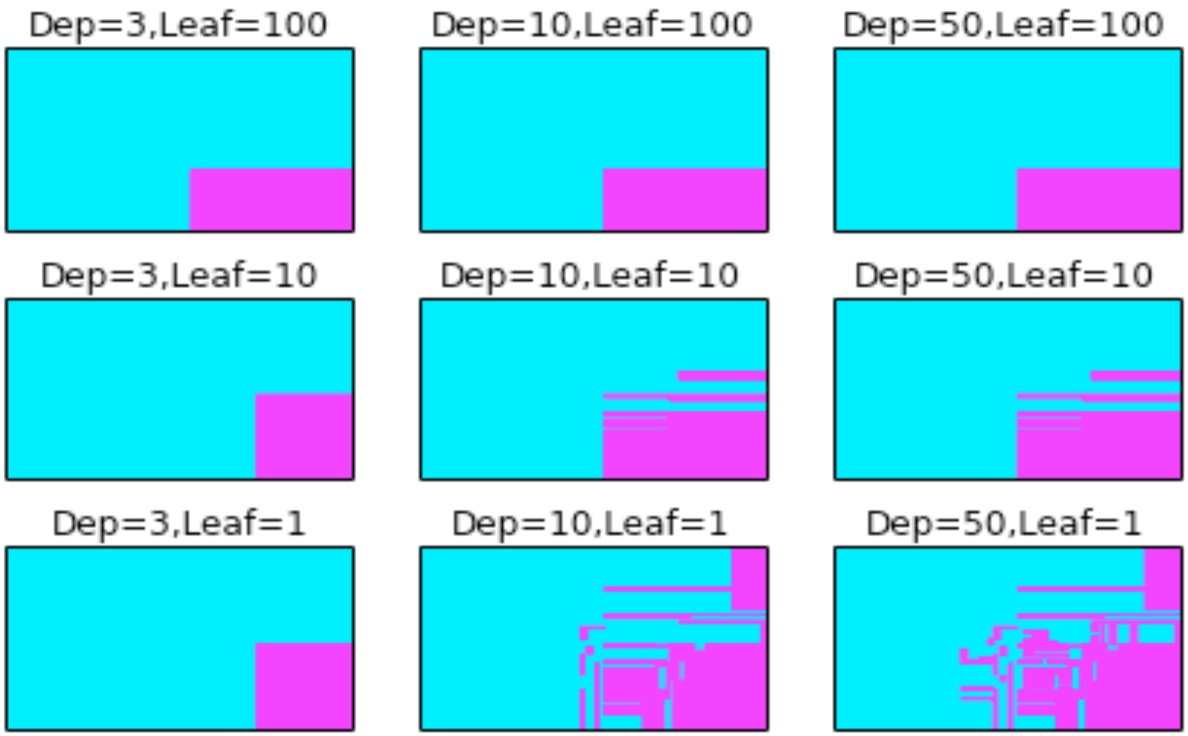
Like any classifier, when the ratio of features to data points shrinks, we risk overfitting.

Three waysfor controlling tree complexity are to limit the depth of the tree, limit the size of an internal node that can be split, and to specify the minimum number of instances that can be in a leaf.



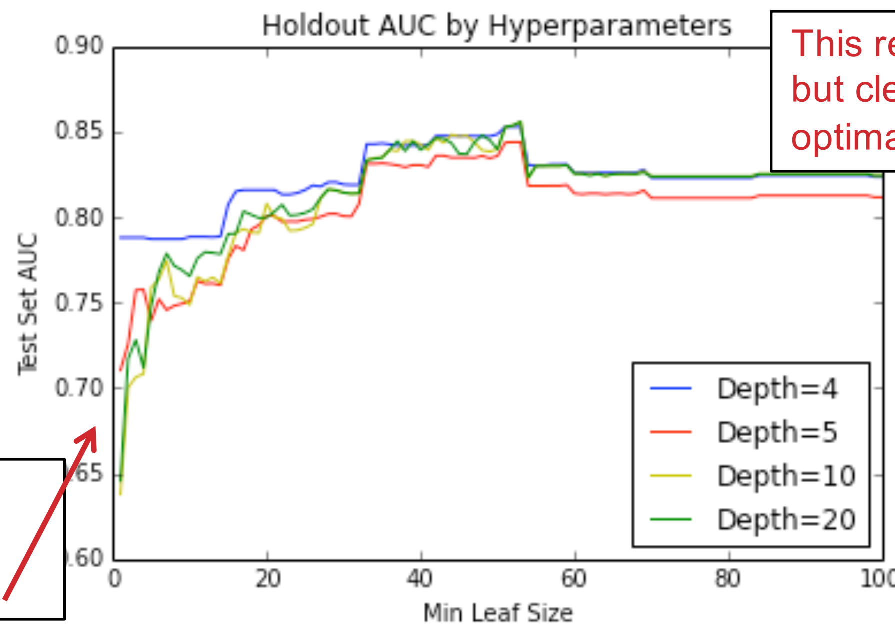
CONTROLLING COMPLEXITY

Visualizing more complex trees is difficult, but we can see how the decision boundary changes once we let trees grow arbitrarily complex. Overfitting is likely if we don't set any limits on the complexity.



LEARNING OPTIMAL TREE SIZE

We built a DT for the “Ads” dataset presented in HW. In this case we have unbalanced class distribution so we choose Area Under ROC curve as our evaluation metric*



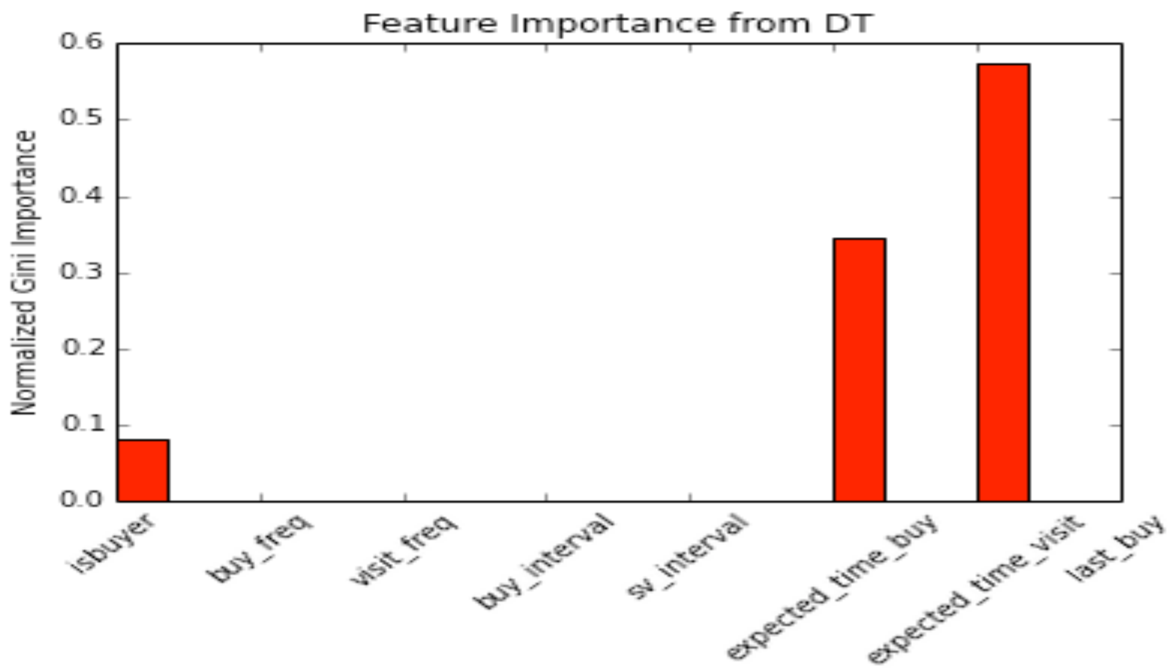
This region is safe, but clearly not optimal.

For most depths, we overfit when the leaves get too small

Note: We haven’t taught this yet, but suffice it to say that AUC is better than accuracy when the classes are skewed.

FEATURE EXPLORATION

Often times the DT is useful as tool for testing for feature interactions as well as ranking features by their ability to predict the target variable. Scikit learn's DecisionTree fit function automatically returns normalized information gain for each feature (also called the Gini Importance)



DECISION TREES IN PRACTICE

Advantages

- Easy to interpret (though cumbersome to visualize if large)
- Easy to implement – just a series of if-then rules
- Prediction/scoring is cheap – total operations = depth of tree
- No feature engineering necessary
  - Can often handle categorical/text features as is (software dependent)
  - Automatically detect non-linearities and interactions

DECISION TREES IN PRACTICE

<u>Disadvantages/Issues</u>

- Easy to overfit – flexibility of algorithm requires careful tuning of parameters and leaf pruning
- DT algorithms are greedy – not very stable and small changes in data can give very different solutions
- Difficulty learning in skewed target distributions (think of how entropy might cause this).
- Not well suited for problems as # of samples shrinks and # of features grows