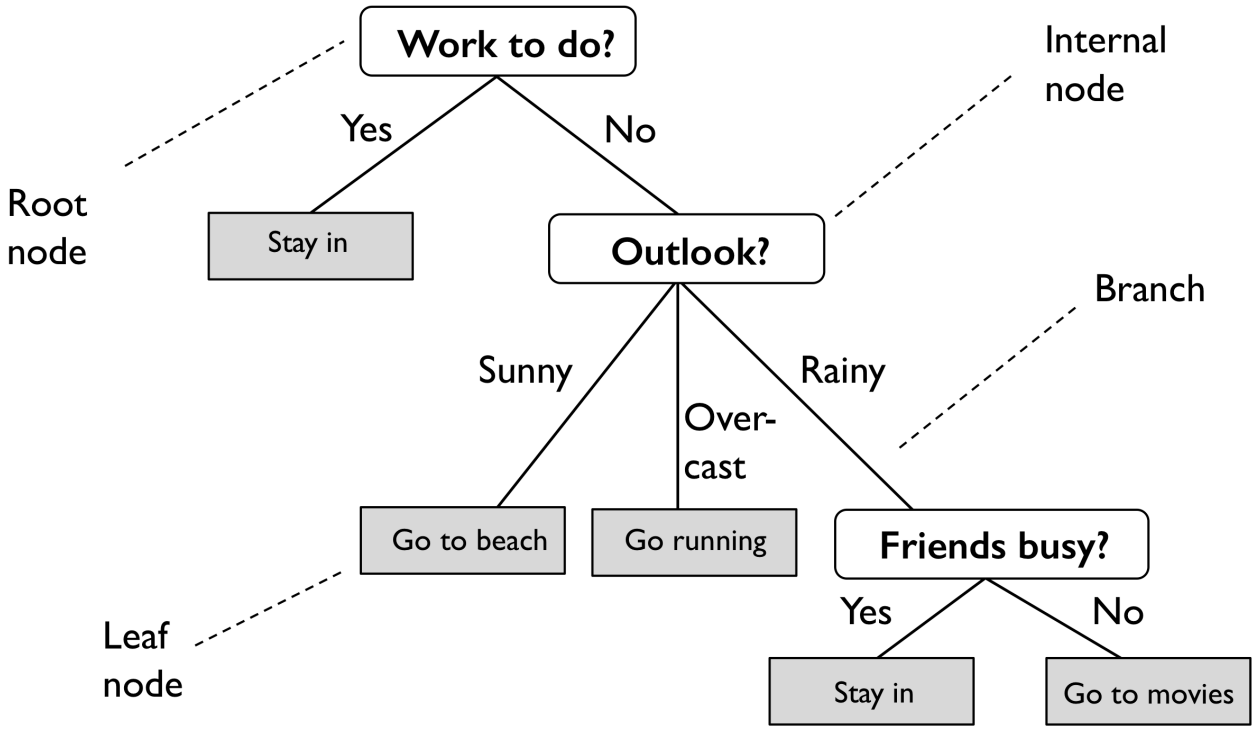
Introduction

- Iterative, top-down construction of the hypothesis
- Can represent any Boolean (binary) function
    - Hypothesis space being searched is the entire space of Boolean functions
        - question in ML is whether the algorithm is able to learn/find the "right" function or a good approximation
    - Size of the hypothesis space determined by the dataset
    - At each node, 2^m potential splits to evaluate given m features
    - Hypothesis space is searched greedily, not exhaustively, because of its exponential nature
    - greedy search over all possible trees
    - in greedy search, we make a series of locally optimal choices, which may not lead to the global optimum
- Order of features in the tree matters
- trees don't require feature normalization

Terminology

- root node: no incoming edge, zero or more outgoing edges
- internal node: one incoming edge two (or more) outgoing edges

- leaf node: each leaf node is a assigned a class label (if nodes are pure; otherwise majority vote)
- parent and child nodes



ML Categories

- Supervised learning algorithm: classification and regression
- Optimization method: combinatorial -> greedy search
- Eager learning algorithm
- Batch learning algorithm
- Nonparametric model
- Deterministic (vs. stochastic)

Relationship to rule-based learning

- Think of nested if-else rules

Rules as conjunctions of conditions:

$rule1 = \text{if } x = 1 \cap \text{if } y = 2 \cap \ldots$

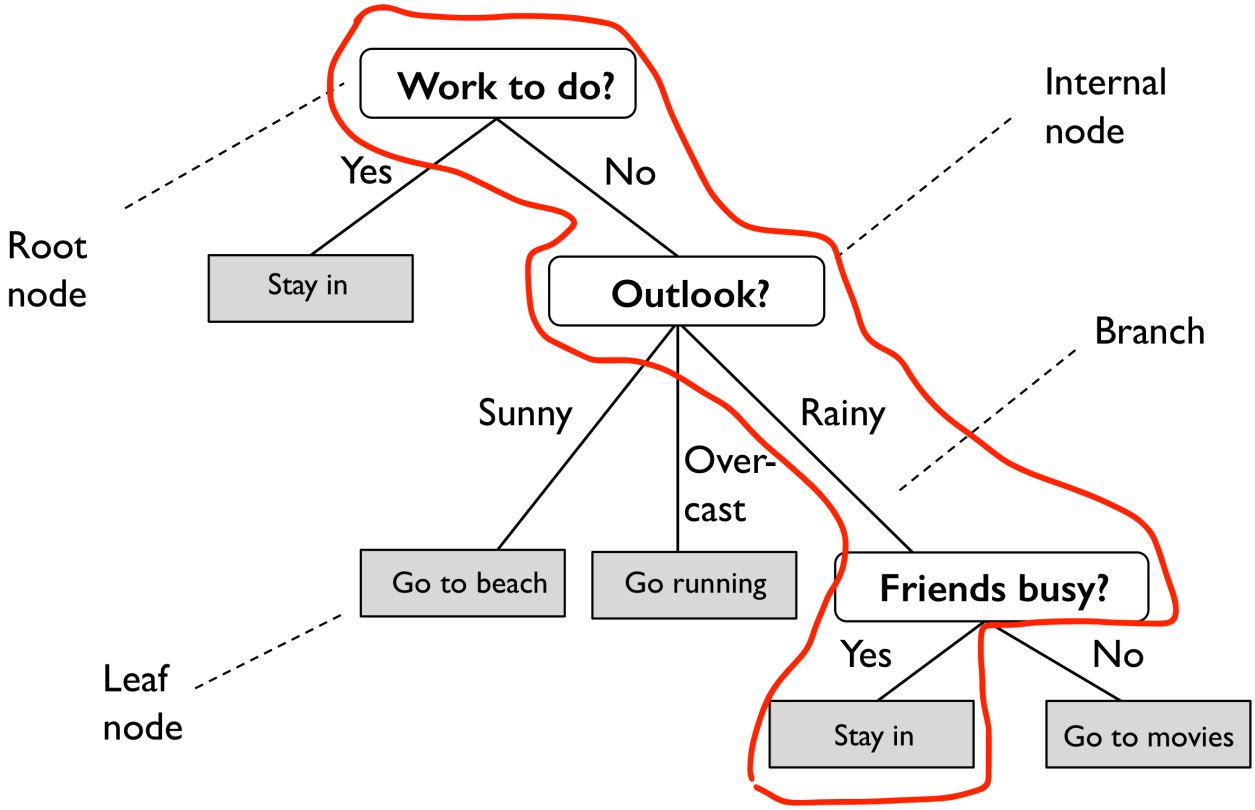
Multiple rules can then be joined into a set of rules, which can be applied to a training example or test instance:

$class1 = \text{if } rule1 = True \cup rule2 = True \cup \ldots$

class 0 is true whenever we don't have a rule to match class 1

In the example below, the rules for the the leaf node in the circled region would be

$$(\text{Work to do?} = \text{False}) \cap (\text{Outlook} = \text{Rainy?}) \cap (\text{Friends busy?} = \text{Ye}$$



Decision rule for class "Stay In"

$$(\text{Work to do?} = \text{False}) \cap (\text{Outlook} = \text{Rainy?}) \cap (\text{Friends busy?} = \text{Ye}$$
$$\cup (\text{Work to do?} = \text{True})$$

- rules can be constructed from decision trees easily: each leaf node is a rule
- we can also construct a decision tree from a set of rules, but it requires more effort (especially if rules were pruned, e.g., where would you place the root of the tree?)
- in contrast to a decision tree, rules only use each attribute once (cheaper to build a rule)
- evaluating a rule set is much more expensive than evaluating a tree, where we only have to go one single branch
- rulesets can have multiple answers if we are not careful

- rules are more prone to overfitting, because they have a larger hypothesis space than decision trees
- rules are very interpretable

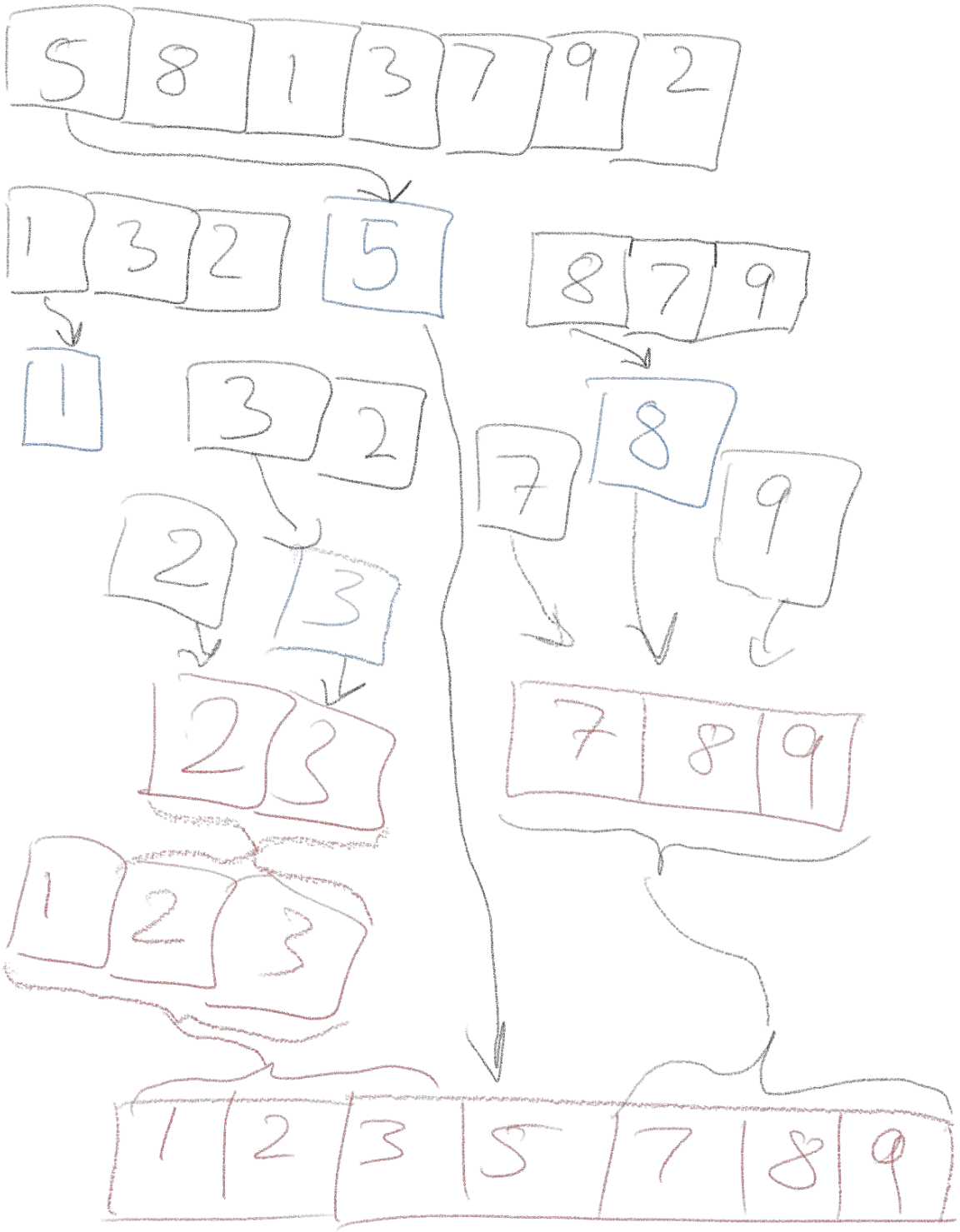
Divide-and-Conquer and Recursion

- Simple recursive algorithms to compute the length of an array

```
def array_len(x):
    if x == []:
        return 0
    else:
        return 1 + array_len(x[1:])
```

- Divide-and-conquer is a concept in computer science where we divide a problem into subproblems of the same type.
- Usually, divide-and-conquer can be implemented using recursion

```
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```



Hunts Algorithm

- basis for CART, ID3, C4.5
- The process of growing a decision tree can be expressed as a recursive algorithm

1) Pick the feature that when parent node\footnote{in first iteration, this is the root node} is split, it results in the largest information gain\footnote{Stop if information gain is not positive.}

2) Stop if child nodes are pure

3) Go back to step 1 for each of the two child nodes

- Majority class/plurality voting if no enough features to make child nodes pure

More formal:

GenerateTree($\mathcal{D}$):

- if $y = 1 \; \forall \; \langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{D}$ or $y = 0 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$:
    - return Tree
- else:
    - Pick best feature x_j:
        - $\mathcal{D}_0$ at Child$_0$: $x_j = 0 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$
        - $\mathcal{D}_1$ at Child$_1$: $x_j = 1 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$

      return Node(x_j, GenerateTree($\mathcal{D}_0$), GenerateTree($\mathcal{D}_1$))

Issues

- empty child node (no training example fits the criterion)
    - use majority vote on parent
- if no attribute can split the tree further, stop and use parent node for majority voting

Design choices

- How to split
    - what measurement/criterion as measure of goodness
    - binary vs multi-category split
        - multi-way split can be expensive
            - need to evaluate all intervals in continuous variables
        - think zip codes; if each training example has a unique zip code, perfect seperation and 0% error on training set. but not a good tree
- when to stop
    - if leaf nodes contain only examples of the same class
    - attribute values are all the same for all examples
    - statistical significance test

Time Complexity

- depth of a binary tree is $\log_2 m$ (if each feature is used once)

- $O(\log m)$ for inference ($\log_2 m = \frac{\log_b m}{\log_b 2}$)
- tree building $O(m \cdot n \log n)$ if features are only sorted once, because dominated by search; can be shown that optimal split is on boundary between adjacent examples (similar feature value) with different class labels \footcite{fayyad1992induction}

ID3, C4.5, CART

General Differences

- Splitting criterion: information gain, statistical tests, objective function (entropy, Gini impurity, misclassification error), etc.
- binary split vs multi-way split
- discrete vs. continuous variables

ID3 -- Iterative Dichotomizer 3

- one of the earlier/earliest decision tree algorithms, by Ross Quinlan
- cannot handle numeric features
- no pruning, prone to overfitting
- short and wide trees (compared to CART)
- (Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.)
- maximizing information gain/minimizing entropy
- discrete features, binary and multi-category features

C4.5

- continuous and discrete features
- chi2 test
- continuous is very expensive, because must consider all possible ranges
- Ross Quinlan 1993
- handles missing attributes (ignores them in gain compute)
- post-pruning (bottom-up pruning)
- Gain Ratio

CART

- "*Classification And Regression Trees*" by Leo Breiman
- continuous and discrete features
- strictly binary splits
- binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret; k-attributes has a 2^{k-1} -1 ways to create a binary partitioning
- numerical and categorical variables
- variance reduction in regression trees
- gini impurity, twoing criteria in classification trees
- cost complexity pruning

Others

- CHAID (CHi-squared Automatic Interaction Detector) \footcite{Kass, G. V. (1980). "An exploratory technique for investigating large quantities of categorical data". *Applied Statistics*. 29 (2): 119–127.}
- MARS (Multivariate adaptive regression splines) \footcite{Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". *The Annals*

of Statistics. 19: 1}
- C5.0 (patented)

Information Gain

- Mutual information:
    - reduction of entropy of one variable by knowing the other
- here: knowing class label by knowing feature value
- want to maximize mutual information when defining a splitting criteria
- i.e., we define the criterion at a note such that it maximizes information gain

$$GAIN(\mathcal{D}, x_j) = H(\mathcal{D}) - \sum_{v \in Values(x_j)} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} H(\mathcal{D}_v)$$

##

Information Theory and Entropy

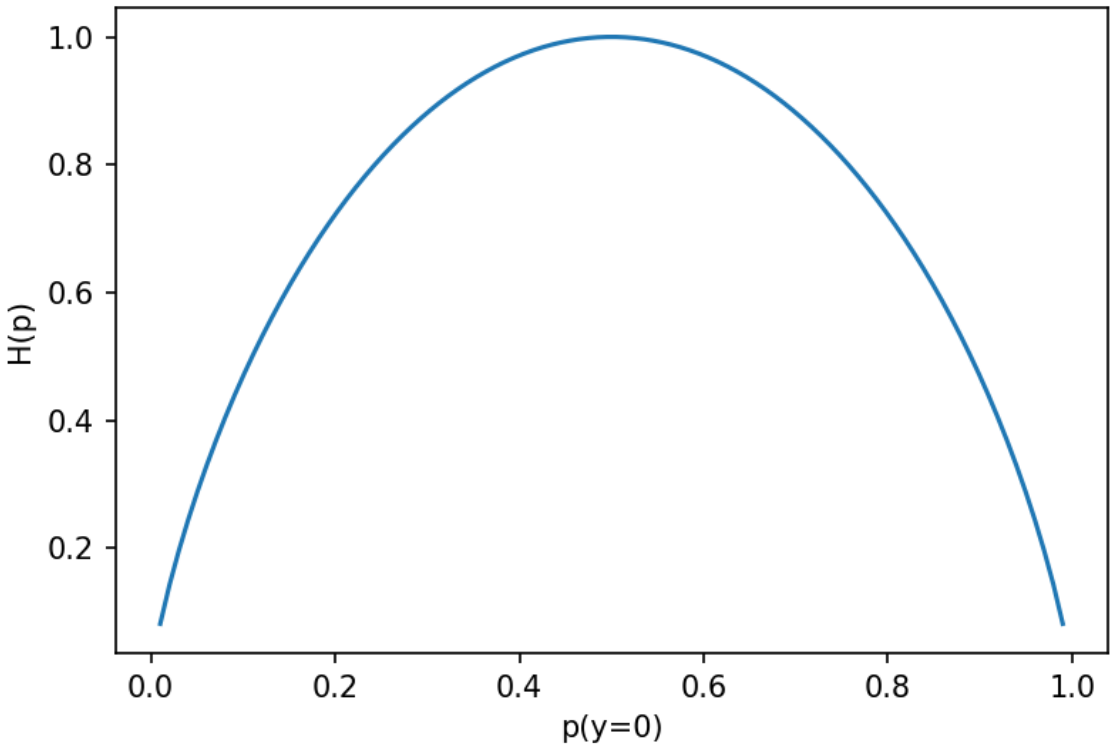
- In ID3, we use Shannon Entropy to measure improvement in a decision tree (instead of misclassification error); i.e., as a optimization metric (impurity measure)
- Shannon Entropy was originally proposed as a way to encode digital information in the form of Bits (0s or 1s).

- Consider entropy as a measure of the amount of information of a discrete random variable (two outcomes, Bernoulli distribution)
- Shannon information:
  - Shannon defined information as the number of bits to encode a number $\frac{1}{p}$, where p is the probability that an event is true (i.e., $\frac{1}{1-p}$ is the uncertainty)
  - The number of bits for encoding $\frac{1}{p}$is $\log_2(1/p)$
  - note $\log_2(1/p) = \log_2(1) - \log_2(p) = -\log_2(p)$
  - $-\log_2(p) \rightarrow [\infty, 0]$; e.g., if we are 100% certain about an event, we gain 0 information
  - E.g., assume 2 soccer teams, team 1 and team 2, both with a win probability 50%
    - If the information "team 1 wins" is transmitted, we transmitted 1 bit: $\log_2(1/p) = \log_2(2) = 1$
- Shannon entropy is the average information
  - $H(p) = \sum_i p_i \log_2(1/p_i)$
  - note $\log_2(1/p) = \log_2(1) - \log_2(p) = -\log_2(p)$
  - $H(p) = -\sum_i p_i \log_2(p_i)$
  - e.g., assume soccer team 1 and team 2 have win probabilities 75% and 25%, respectively

$$
\begin{aligned}
H(p) &= \frac{1}{2}\left(-0.75 \times \log_2(0.75) - 0.25 \times \log_2(0.25) \right) \\
&= (-0.75 \times 0.41 - 0.25 \times 2)/2 \\
&= 0.81
\end{aligned}
$$

- consider in a real application, we can't send fractional bits
  - say our soccer game has three outcomes:

        - (1) team 1 wins, (2) team 2 wins, (3) draw
        - these are 3 states.
        - 2 bits encodes 2 states (2^1 = 2) , 2 bits encode 4 states (2^2 = 4)
        - we wasted 1 bit
- The key idea behind Shannon entropy is that we can send information more efficiently (saving bits) depending on how certain we are



- If we have i class labels (i.e., i different values for y), then the entropy can be as large as $\log_2 i$.
    - E.g., for 10 classes: $10 \times (-(0.1 \times \log_2(0.1))) = \log_2(10) = 3.32$

class i given feature x_j

$$H(p) = -\sum_i p(i|x_j) \log_2 (p(i|x_j))$$

- In the context of decision trees, think of entropy as the minimum number of bits that are required to encode the classification of data points. For instance, if we have p=1 we need to send 1 bit on average (most expensive) to classify a data point.

Why Growing Decision Trees via Entropy or Gini Impurity instead of Misclassification Error?

Consider the more general formula for measuring information gain,

$$GAIN(\mathcal{D}, x_j) = I(\mathcal{D}) - \sum_{v \in Values(x_j)} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} I(\mathcal{D}_v),$$

where I is a function that measures the impurity of a given node. If I defined as the Entropy measure (H) we defined earlier, this equation is equal to the information gain equation used in classification trees.

Instead of using Entropy as an impurity measure, the misclassification error ERR seems to be another reasonable choice, where

$$ERR(\mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}^{[i]}, y^{[i]}),$$

with the 0-1 Loss,

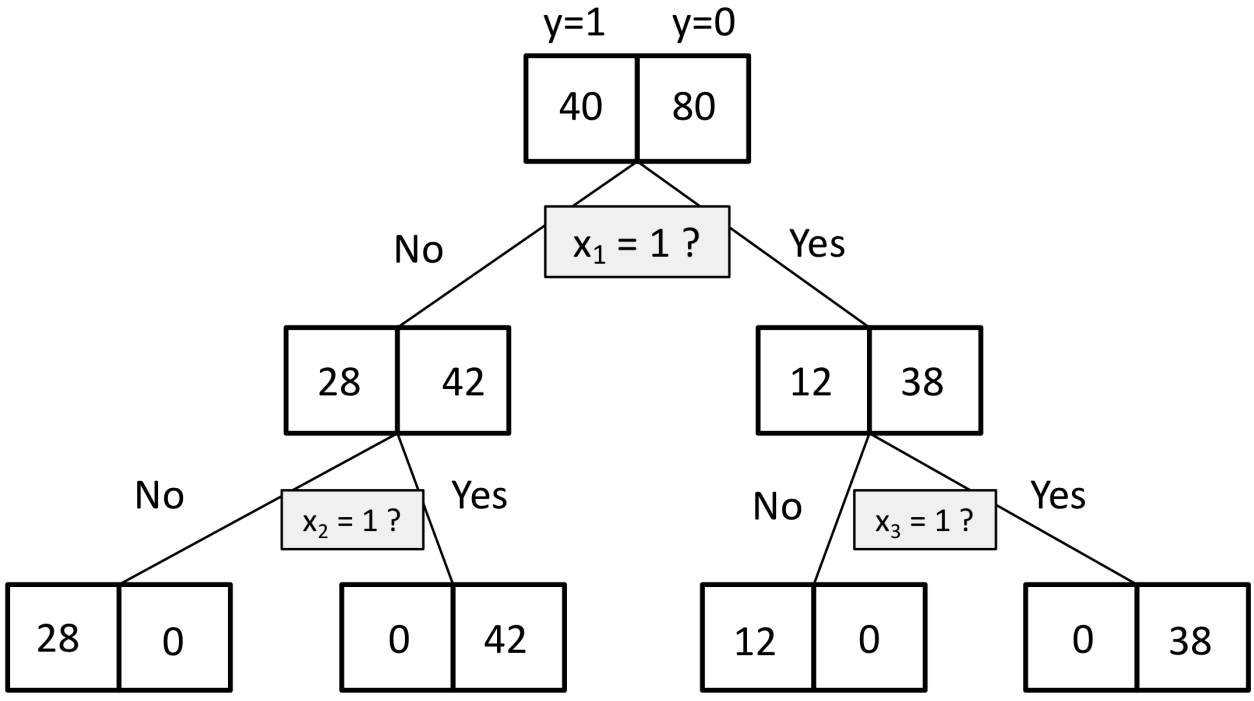
$$L(\hat{y}, y) = \begin{cases} 0 \text{ if } \hat{y} = y, \\ 1 \text{ otherwise.} \end{cases}$$

This, in case of the training set, is equal to

$$ERR(p) = 1 - \max_i(p(i|x_j))$$

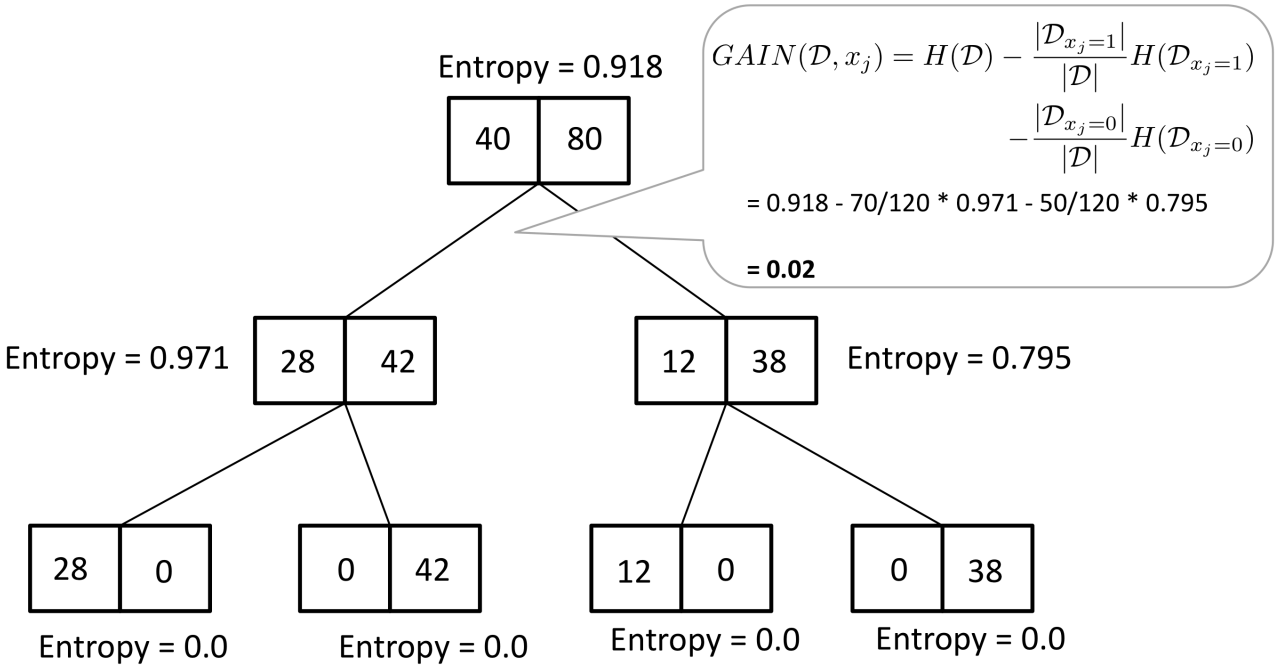
for a given node if we use majority voting at this node.

Now, to see the difference between using the misclassification error as opposed to the entropy measure for computing the information gain upon growing a decision tree, we will take a look at an example. Consider the following binary decision tree for a binary class problem with binary features:



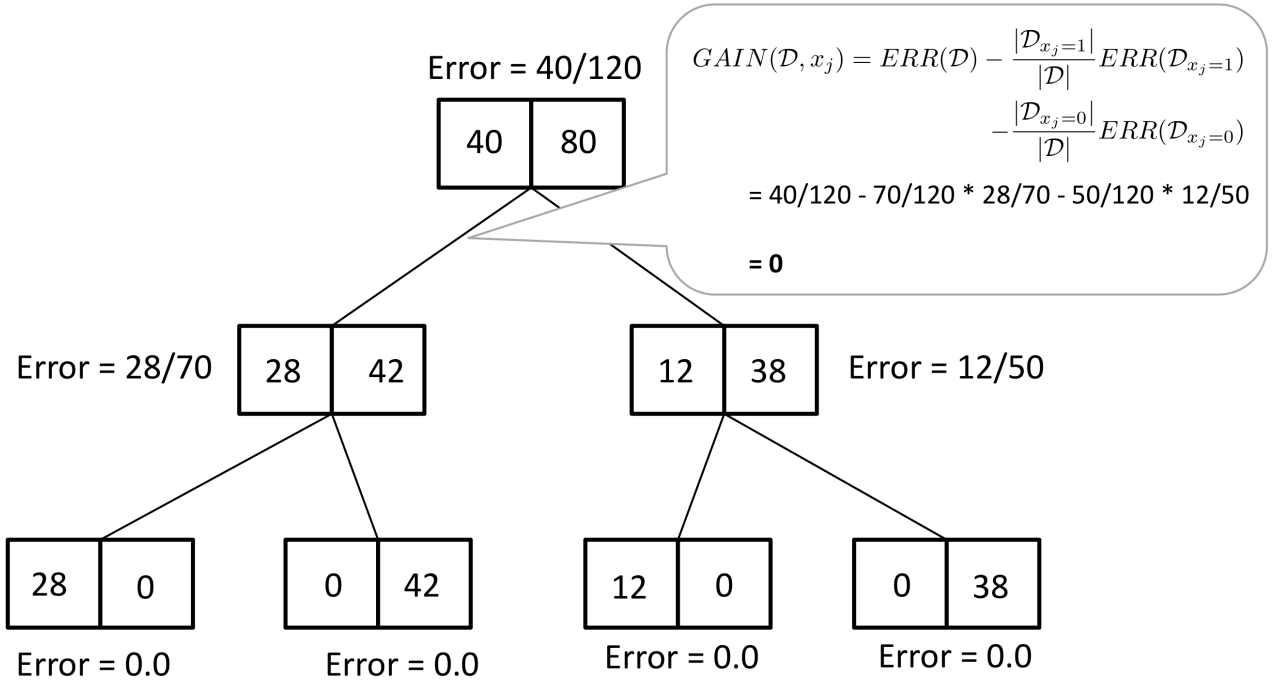
Note that this is a toy example where we assume that there exists three features $x_1, x_2, x_3 \in \{0, 1\}$ that result in a perfect class separation if split as illustrated in the preceeding figure.

We split the tree using entropy as an information or impurity measure first, and the entropy values of the child nodes of the root node are provided in the figure below.



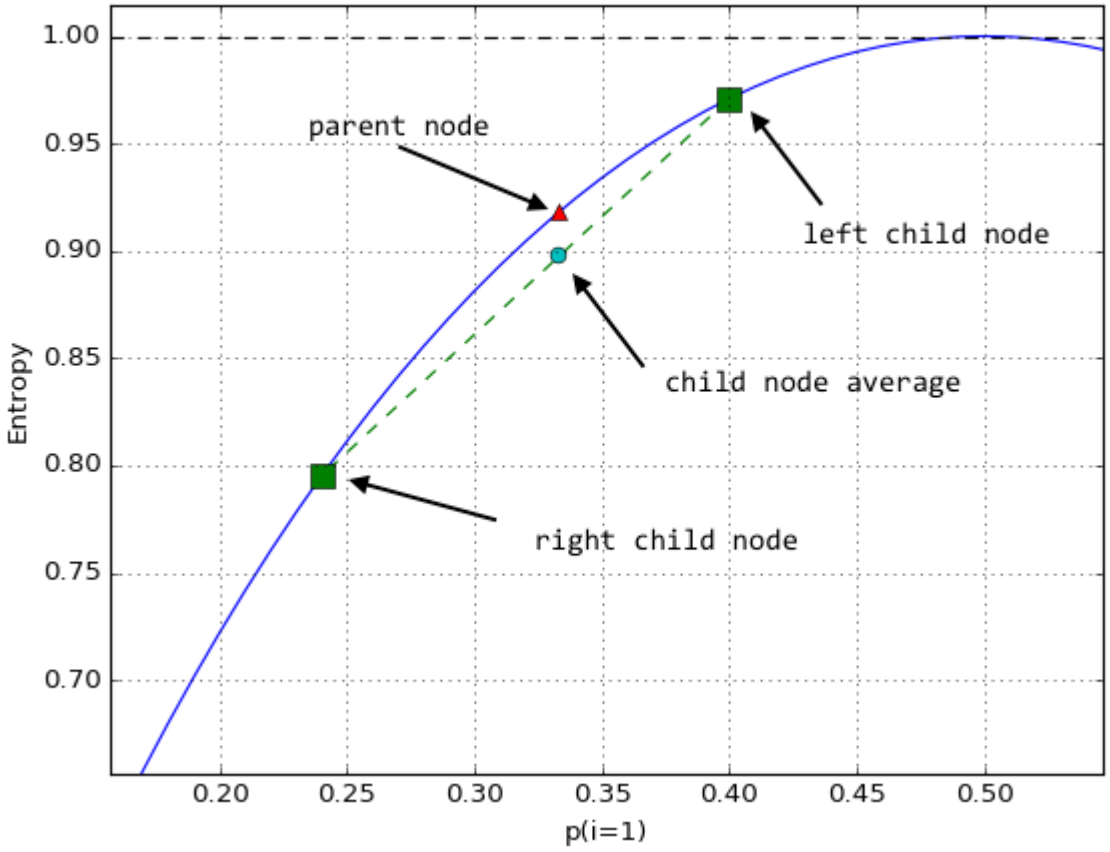
We note that splitting the root node into the two children nodes results in entropy values 0.971 and 0.795, respectively. The information gain of this split is 0.02. In the toy example, the splits that follow this first split will seperate the classes perfectly, as indicated in the figure.

Next, consider the same decision tree in a scenario where the misclassification error was used as an impurity metric, as shown in the next figure.



As shown in the previous figure, the information gain upon splitting the root node using the misclassification error as impurity metric is 0, which means that performing this split actually did not result in an improvement compared to the original state (i.e., the root node). Hence, according to the decision tree algorithm(s) we defined earlier, we would stop growing the tree if we do not make further improvement(s) as measured via information gain.

To provide an intuitive explanation as to why this happens, consider the next figure, which shows the entropy values of the root node plotted along with the two child nodes.



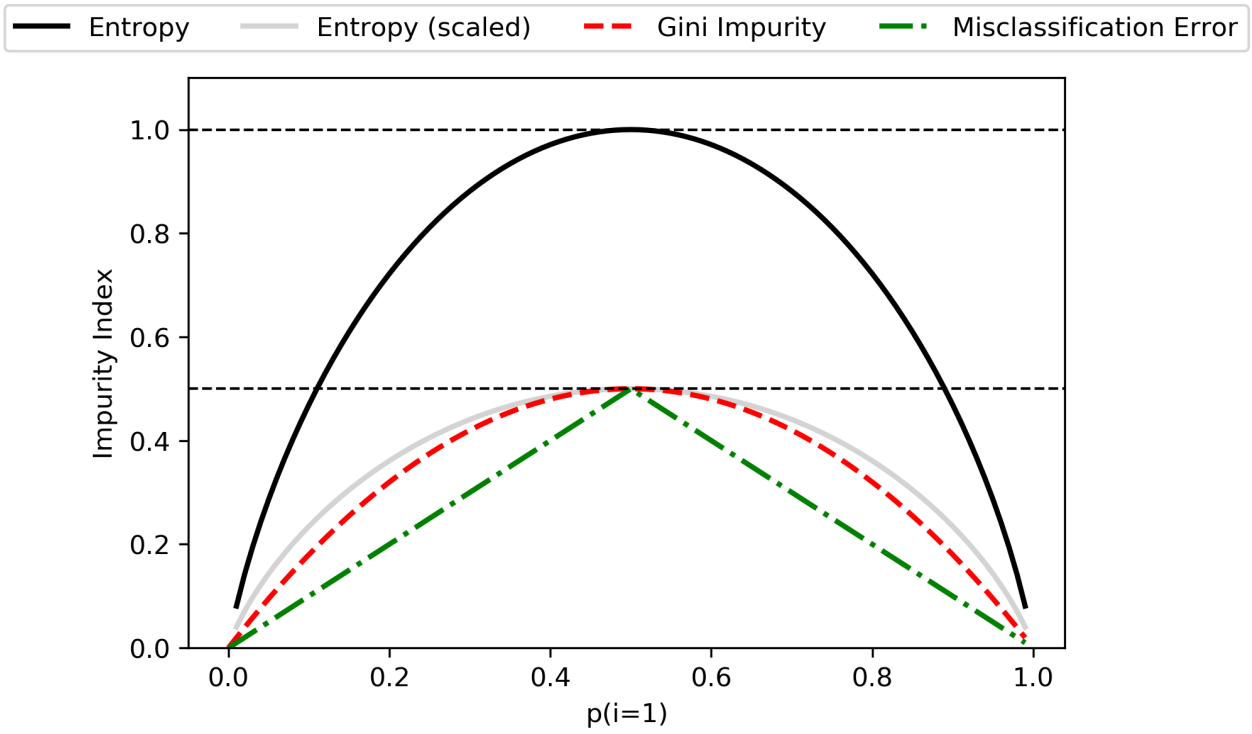
As it can be seen in the plot above, the weighted entropies of the child nodes is always larger than the average of the entropy values due to the convex shape of the entropy function. This means that we can always find a splitting criterion to continue growing the tree where the tree growth might get stuck because the error doesn't improve.

Gini Impurity

- Used in CART

$$Gini(t) = 1 - \sum_{i=0} \left(p(i|t)^2\right)$$

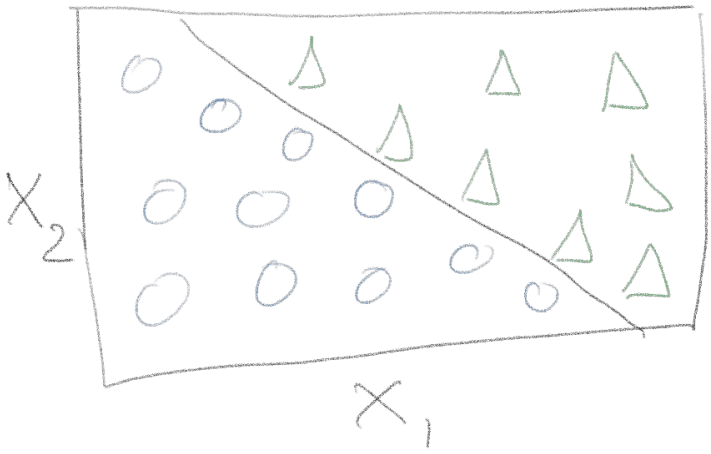
- Doesn't really matter, except, has the same bell shape which is important
- is easier to compute

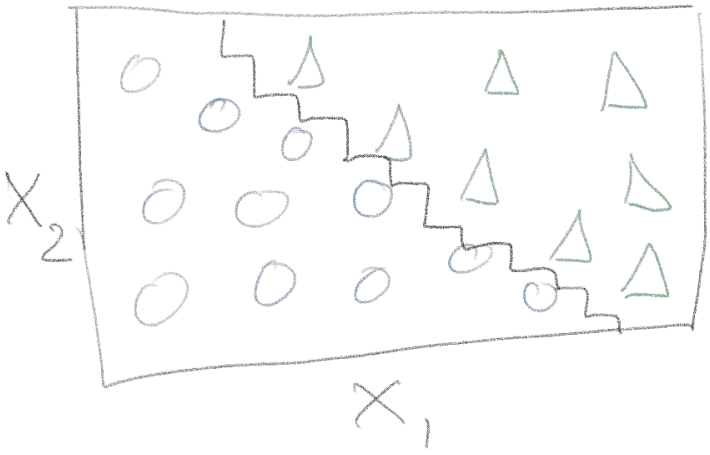


Improvements

Continuous Features

- splits occur perpendicular to axes





Grain Ratio

penalizes splitting categorical attributes with many values (e.g., think date column, or really bad: row ID) via the split information

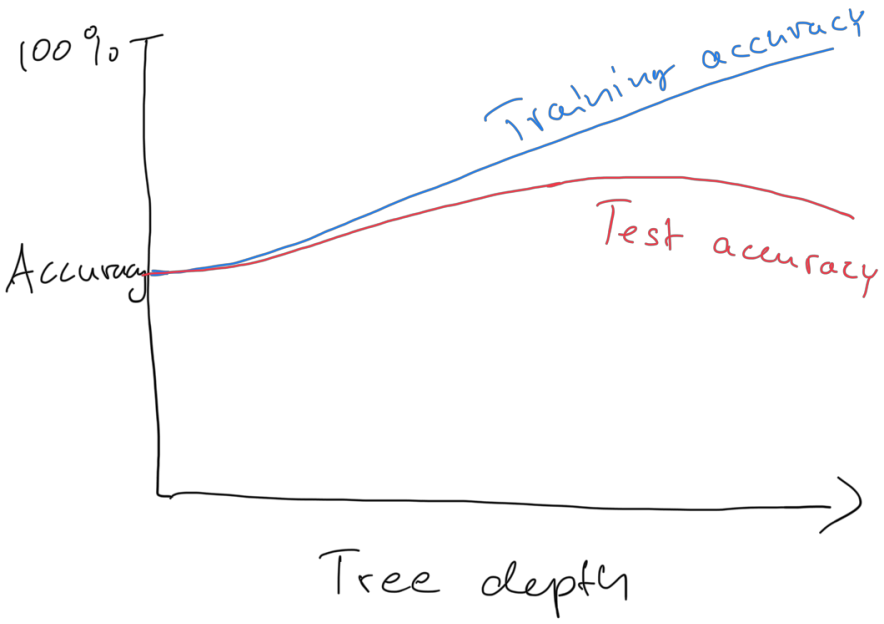
Quinlan 1986

$$GainRatio(\mathcal{D}, x_j) = \frac{Gain(\mathcal{D}, x_j)}{SplitInfo(\mathcal{D}, x_j)}$$

where SplitInfo measures the entropy of the attribute itself:

$$SplitInfo(\mathcal{D}, x_j) = -\sum_{v \in x_j} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} \log_2 \frac{|\mathcal{D}_v|}{|\mathcal{D}|}$$

Overfitting



- Plot starts at 50% ACC because of majority vote, test sample same distribution as train example.
- Overfitting occurs if models pick up noise or errors in the training dataset; hence, overfitting can be seen as a performance gap between training and test data

Tom Mitchell:

Given a hypothesis space H, a hypothesis $h \in H$is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

- think of zipcode example or forgetting the row index in the training set
- inductive bias
- the more features the more likely the three overfits
- Occam's razor: Favor simpler hypothesis, because a simpler hypothesis that fits the data equally well is more likely than a complext one

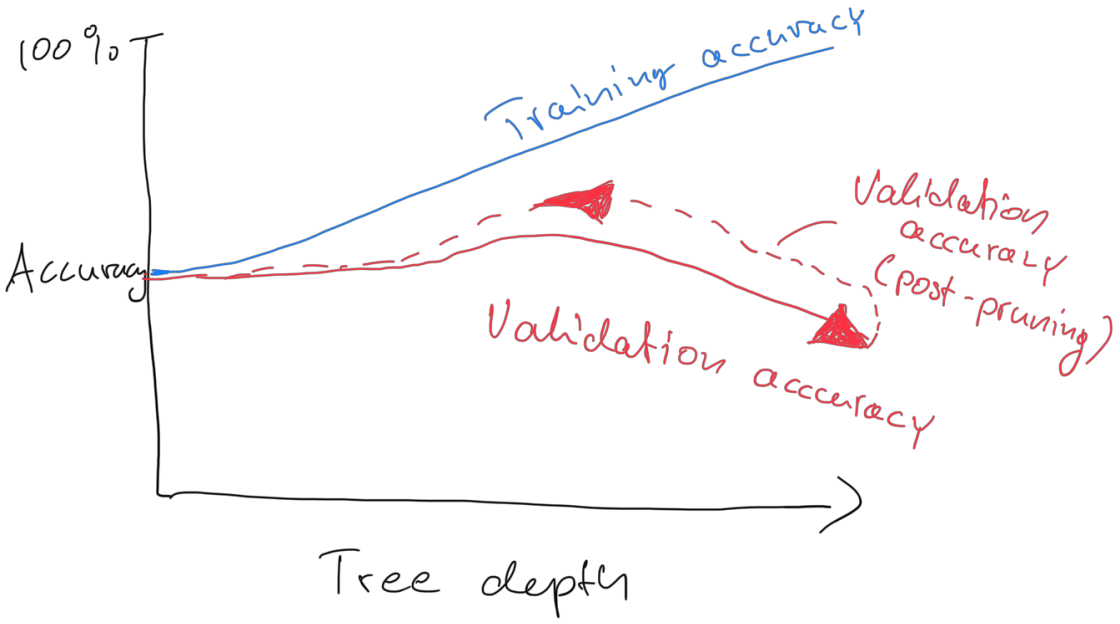
Pre-Pruning

- Set a depth cut-off (maximum tree depth) *a priori*
- Cost-complexity pruning: $I + \alpha|N|$, where I is an impurity measure, α is a tuning parameter, and $|N|$ is the total number of nodes.
- stop growing if split is not statistically significant (e.g., Chi2 test), i.e., like in ID3
- set a minimum number of examples per tree

Post-Pruning

- grow full tree first, then remove nodes, in C4.5
- Quinlan 1993
- chi2 test for splitting
- reduced-error pruning, via validation set, problematic for limited data

- greedily remove nodes based on validation set performance; generally improves performance
- can also convert trees to rules first and then prune the rules
    - there is one rule per leaf node
    - if rules aren't sorted, rule sets are very expensive to evaluate, but more expressive
    - rules from decision trees are mutually exclusive though
    - here we can prune rules independently of others (means we do not remove both child nodes if we remove the root node)
    - $(\text{Work to do?} = \text{False}) \cap (\text{Outlook} = \text{Rainy?}) \cap (\text{Friends busy?} = \text{Yes})$
        - If $(\text{Work to do?} = \text{False}) \cap (\text{Outlook} = \text{Rainy?})$ Then "Stay in = True"
            - if: antecedent, Then: consequent



Decision Tree for Regression

minimizes MSE at each node

Summary

Pros and Cons

- (+) Easy to interpret and communicate
- (-) Easy to overfit
- (-) Elaborate pruning required
- (-) Expensive to just fit a "diagonal line"
- (-) Output range is bounded (dep. on training examples) in regression trees