Decision Trees pt. 2

12 February 2026

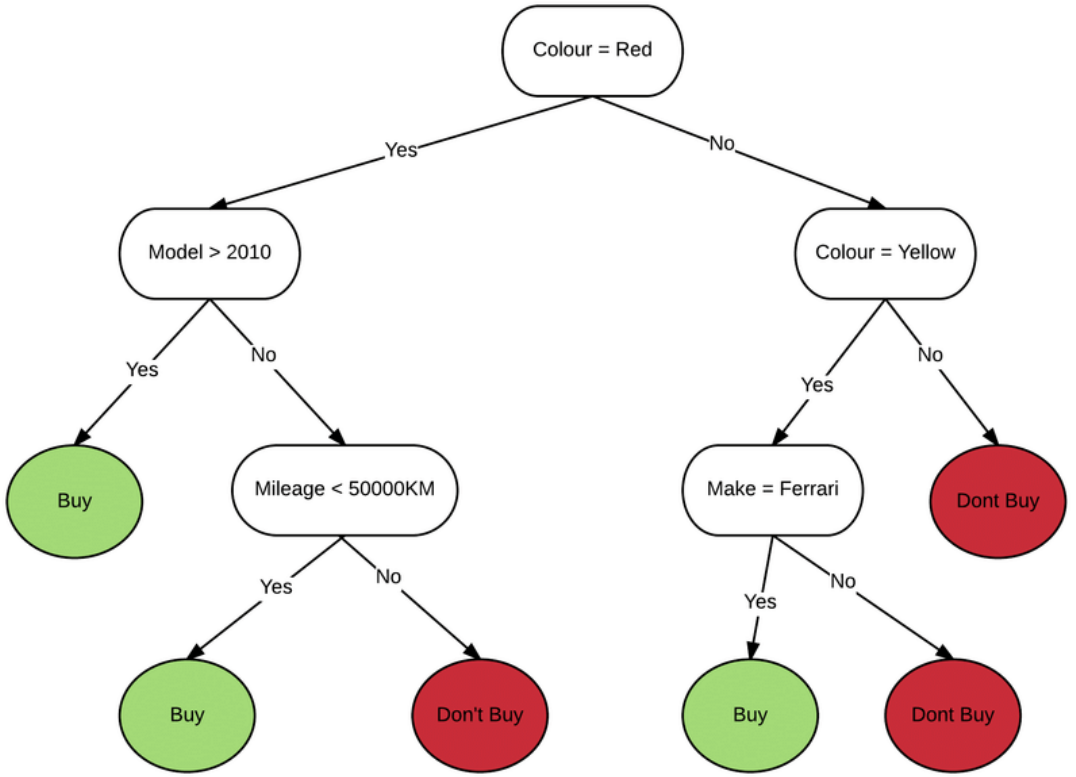
Alex Lyman

Quick Decision Tree Review

Decision Tree Review

- Decision tree is a classifier in the form of a tree.
  - Decision node: specifies a test on a **single attribute**
  - Leaf node: indicates the value of the target attribute
  - Edge: split of one attribute
  - Path: collection of choices from root to leaf

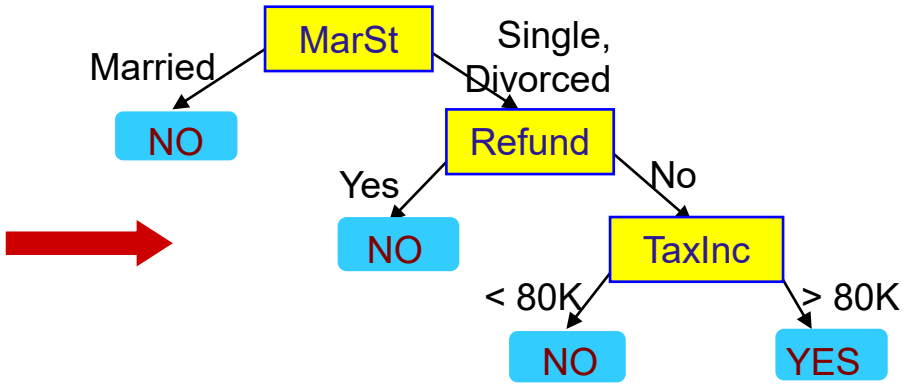
- Decision trees classify instances or examples by starting at the root of the tree and moving through it until hitting a leaf node.



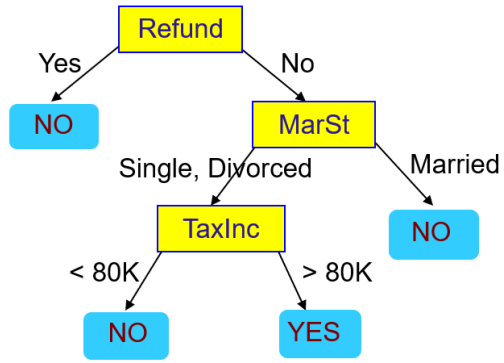
Non-Uniqueness

- Decision trees are not unique:
  - **You can make multiple different decision trees with the same dataset.**

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

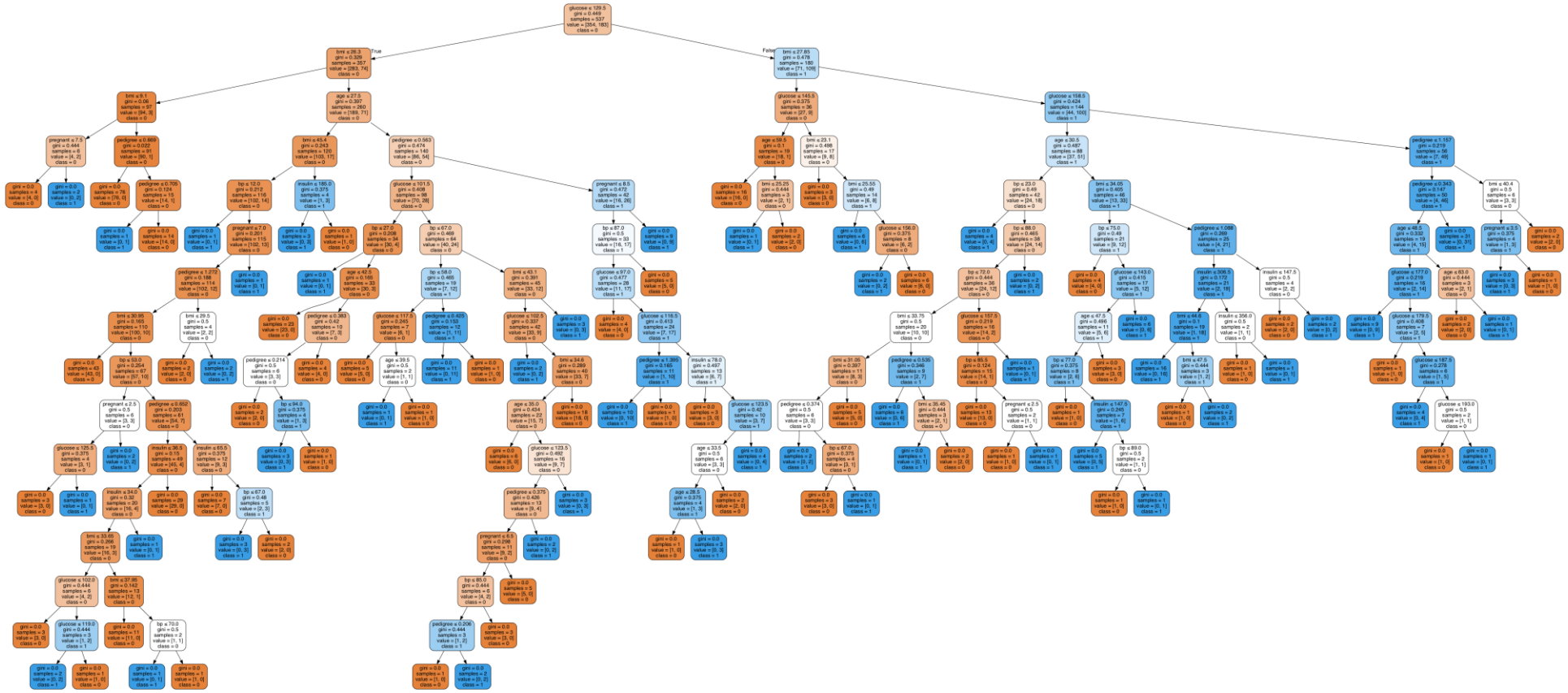


OR



Decision Trees

- Can grow very large and difficult to understand
- Typically they can overfit the training data



Creating Decision Trees

Decision Tree Algorithms

- **Greedy algorithms**
    - Grow a decision tree by making a series of ***locally optimum decisions on which attributes to use*** for partitioning the data.

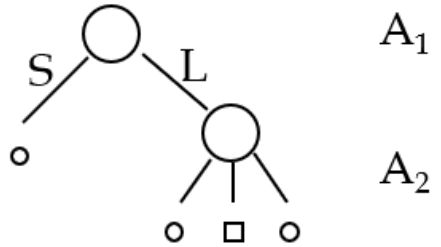
- Multiple ways to construct trees:
    - **ID3** (Iterative Dichotimiser 3)
    - **CART** (Classification and Regression Trees)

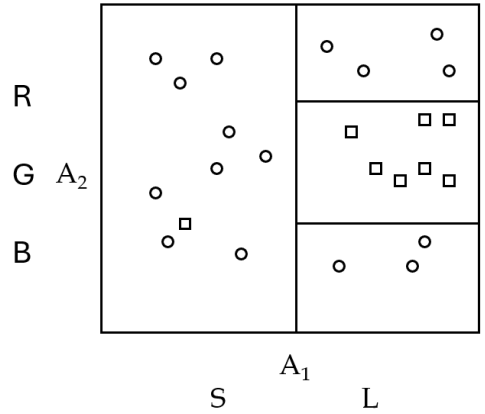
- ID3 and CART work similarly, by greedily splitting on the next best feature.

If we are trying to split iteratively, how do we decide which feature to split on?

Decision Tree Algorithms

- Think of creating Decision Trees like playing a game of 20 questions.
- You want to ask the ‘best question’ at each step. (get the most information)
- ID3 is GREEDY
  - (cares about taking the next best choice)
- At each step, “Which feature (attribute) would create the cleanest, purest groups”
- We can do this by measuring potential child node purity.





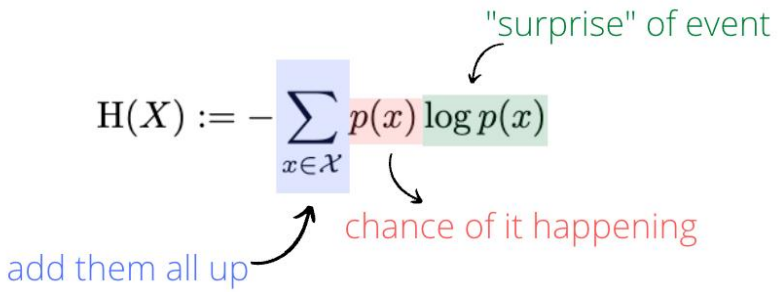
Impurity measures

- We mostly care about the first two measures:

$$\text{Entropy}(t) = -\sum_{i=1}^{c} p(i \mid t) \log p(i \mid t)$$

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} \left[p(i \mid t) \right]^2$$

$$\text{Classification error}(t) = 1 - \max_i \left[p(i \mid t) \right]$$



Pick-two approximation of Entropy

Impurity Measure Comparison

$$\text{Entropy}(t) = -\sum_{i=1}^{c} p(i \mid t) \log p(i \mid t)$$

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} [p(i \mid t)]^2$$

$$\text{Classification error}(t) = 1 - \max_i [p(i \mid t)]$$

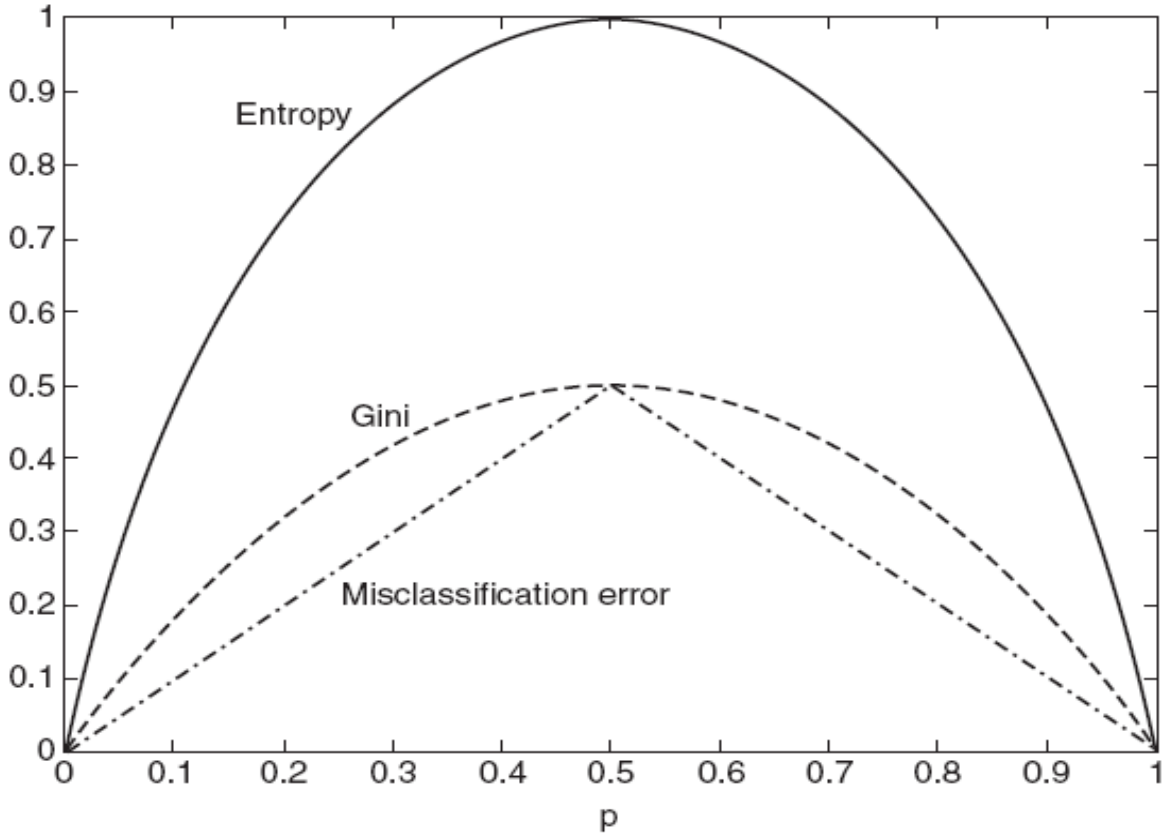


Figure 4.13. Comparison among the impurity measures for binary classification problems.

End of Review

How do we Know Which Feature to Split on?

Information Gain

Information Gain

- ***Gain of a test condition:*** compare the impurity of the parent node with the impurity of the child nodes.
- Entropy is a measure of information.

Information Gain

Information Gain:

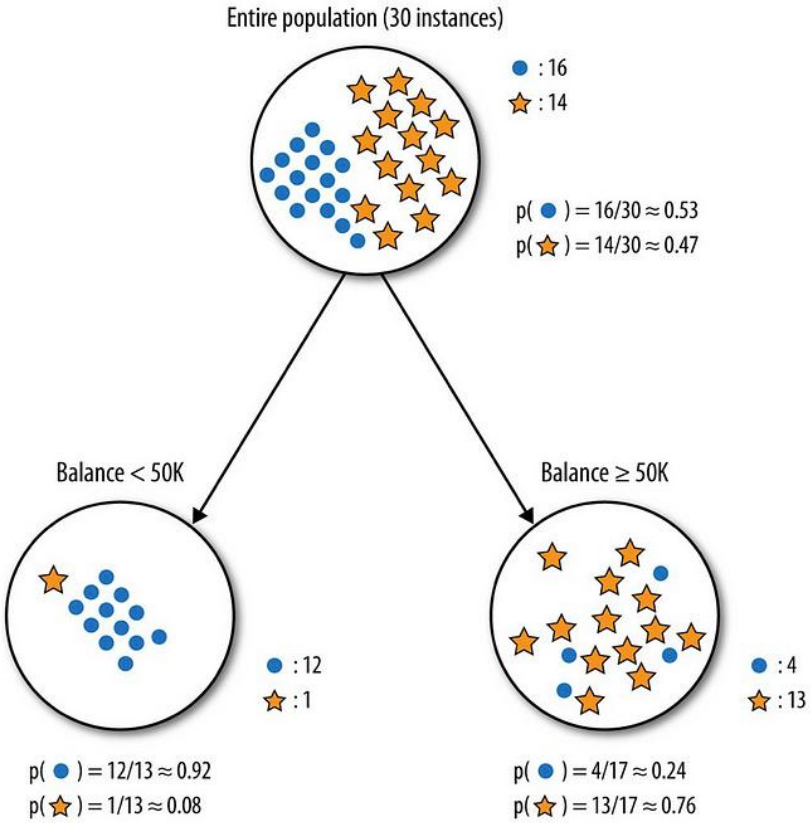
$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

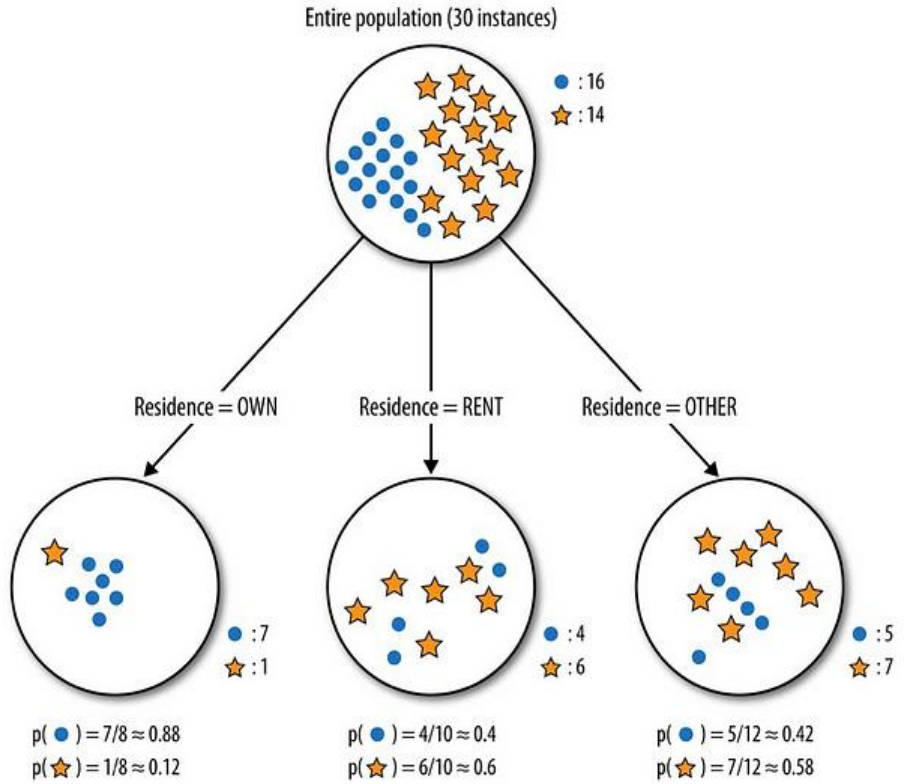
(where parent Node, p is split into k partitions, and
n_i is number of records in partition i)

- Entropy of parent – weighted sum of entropy of children
- Measures reduction in entropy achieved because of the split
- ID3 chooses to split on the attribute that results in the largest reduction, i.e, (maximizes GAIN)
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

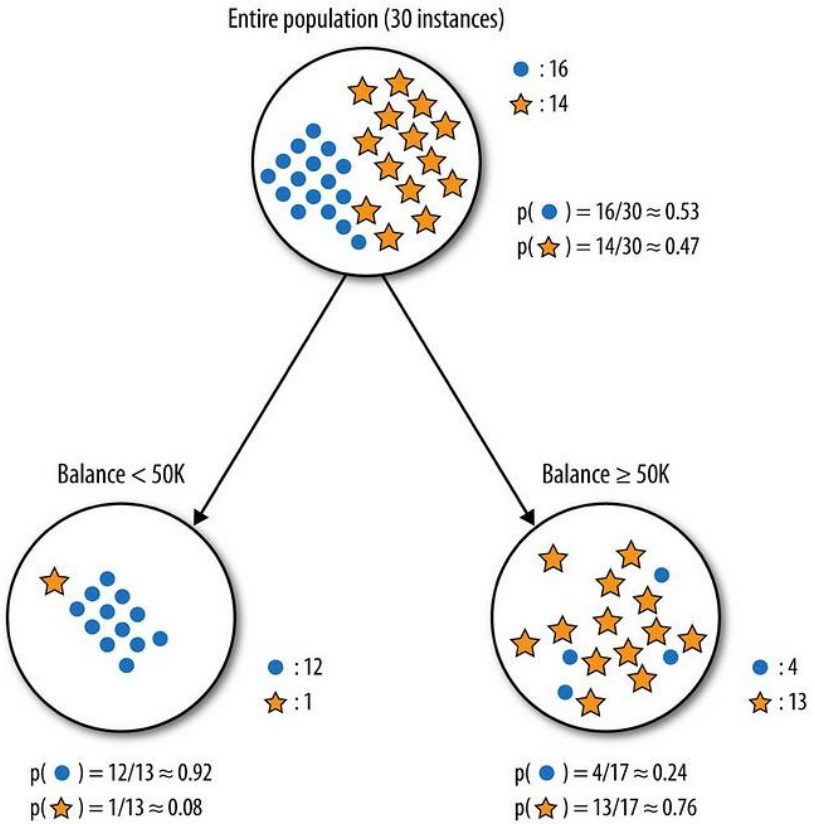
Information Gain

Information Gain: $GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$





Which feature should we split on? (visual intuition)



$$E(Parent) = -\frac{16}{30}\log_2\left(\frac{16}{30}\right) - \frac{14}{30}\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Balance < 50K) = -\frac{12}{13}\log_2\left(\frac{12}{13}\right) - \frac{1}{13}\log_2\left(\frac{1}{13}\right) \approx 0.39$$

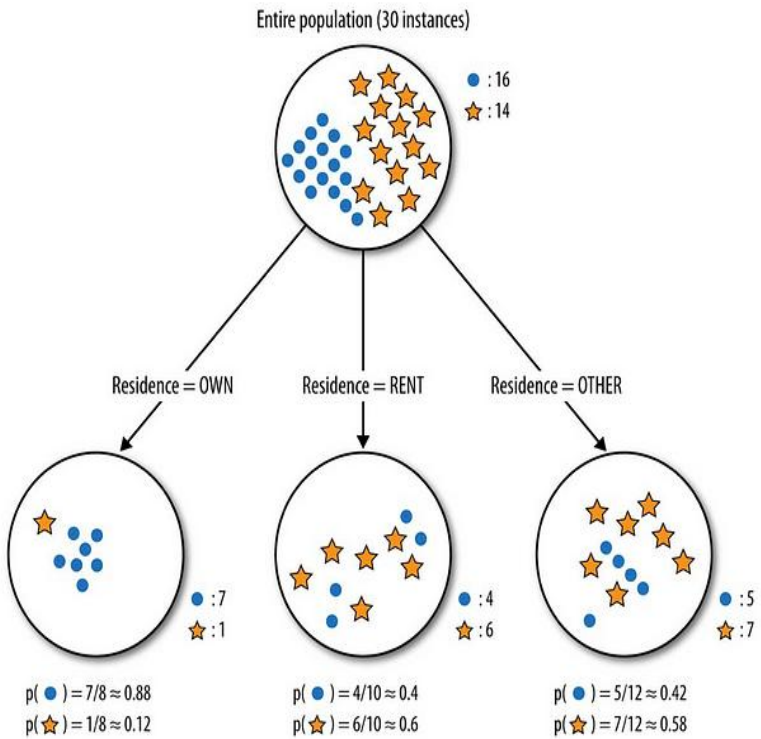
$$E(Balance > 50K) = -\frac{4}{17}\log_2\left(\frac{4}{17}\right) - \frac{13}{17}\log_2\left(\frac{13}{17}\right) \approx 0.79$$

Weighted Average of entropy for each node:

$$E(Balance) = \frac{13}{30} \times 0.39 + \frac{17}{30} \times 0.79$$
$$= 0.62$$

Information Gain:

$$IG(Parent, Balance) = E(Parent) - E(Balance)$$
$$= 0.99 - 0.62$$
$$= 0.37$$



$$E(\textit{Residence} = \textit{OWN}) = -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

$$E(\textit{Residence} = \textit{RENT}) = -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \approx 0.97$$

$$E(\textit{Residence} = \textit{OTHER}) = -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \approx 0.98$$

Weighted Average of entropies for each node:

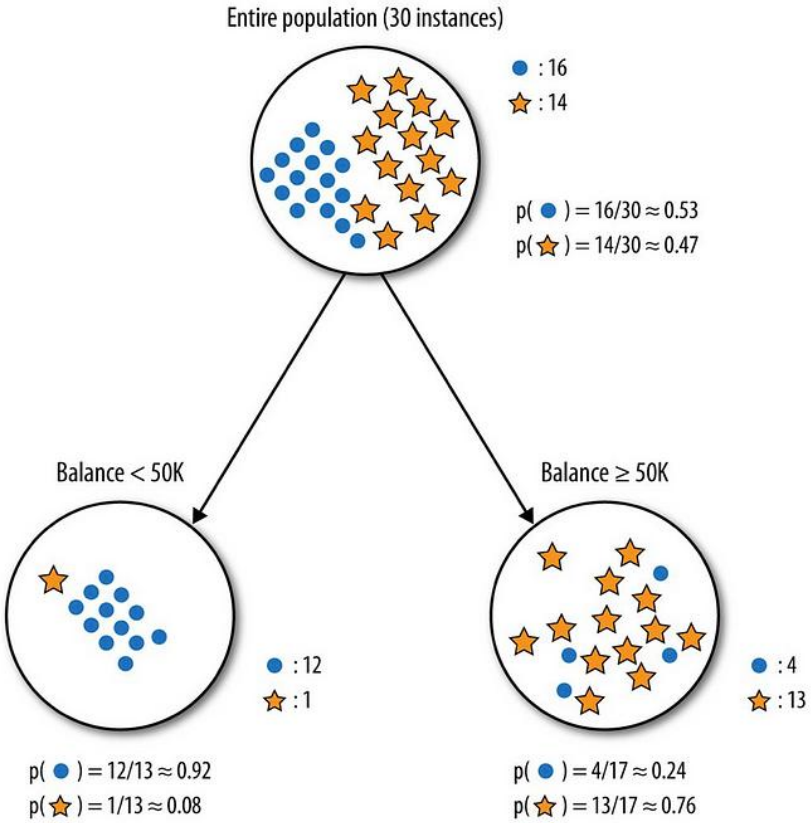
$$E(\textit{Residence}) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

Information Gain:

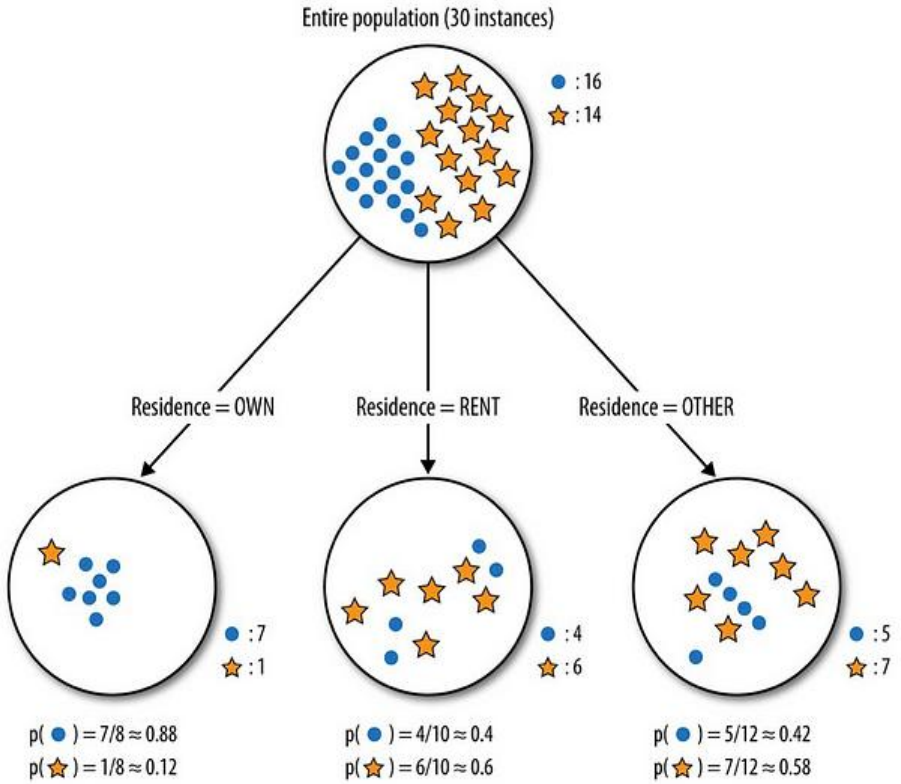
$$\begin{aligned} IG(\textit{Parent}, \textit{Residence}) &= E(\textit{Parent}) - E(\textit{Residence}) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

Information Gain

Gain = .37



Gain = .13



Which feature should we split on?

Information Gain Example 2

Example

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

- Compute the entropy for data-set
- For every attribute/feature:
  - Calculate entropy for all categorical values
  - Take average information entropy for the current attribute
  - Calculate gain for the current attribute
- Pick the highest gain attribute.
- Repeat until we get the tree we desired.

Entropy of the dataset S

$$\boldsymbol{E(S) = \sum_{c \in C} -p(c) log_2(p(c))}$$

$$\boldsymbol{C = \{yes, no\}}$$

Out of 14 instances, 9 are yes and 5 are no

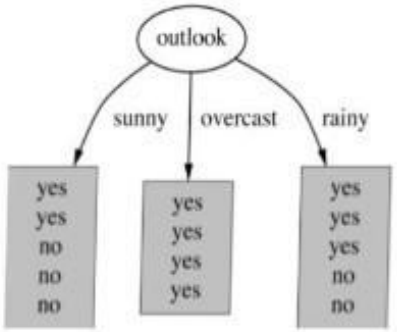
E(yes) = -(9/14)*$\log_2$(9/14) = 0.41

E(no) = -(5/14)*$\log_2$(5/14) = 0.53

E(S) = E(yes) + E(no) = 0.94

Example

For every feature, calculate the entropy and the information gain



E (Outlook=sunny) = $-\frac{2}{5}\log\left(\frac{2}{5}\right)-\frac{3}{5}\log\left(\frac{3}{5}\right)=0.971$

E (Outlook=overcast) = $-1\log(1)-0\log(0)=0$

E (Outlook=rainy) = $-\frac{3}{5}\log\left(\frac{3}{5}\right)-\frac{2}{5}\log\left(\frac{2}{5}\right)=0.971$

Average Entropy information for Outlook

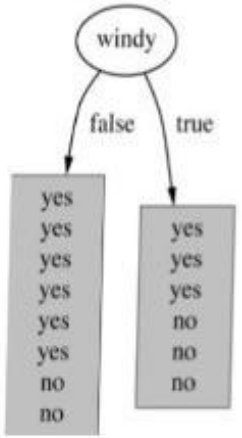
I (Outlook) = $\frac{5}{14}*0.971+\frac{4}{14}*0+\frac{5}{14}*0.971=0.693$

Gain (Outlook) = E(S) – I (outlook) = 0.94-.693 = 0.247

$\mathrm{E}(S, Outlook)$

$\sum_{t\in T} p(t)E(t)$

Gain $= E(S) - E(S, Outlook)$



E (Windy=false) = $-\frac{6}{8}\log\left(\frac{6}{8}\right)-\frac{2}{8}\log\left(\frac{2}{8}\right)=0.811$

E (Windy=true) = $-\frac{3}{6}\log\left(\frac{3}{6}\right)-\frac{3}{6}\log\left(\frac{3}{6}\right)=1$

Average entropy information for Windy

I (Windy) = $\frac{8}{14}*0.811+\frac{6}{14}*1=0.892$

Gain (Windy) = E(S) – *I* (Windy) = 0.94-0.892=0.048

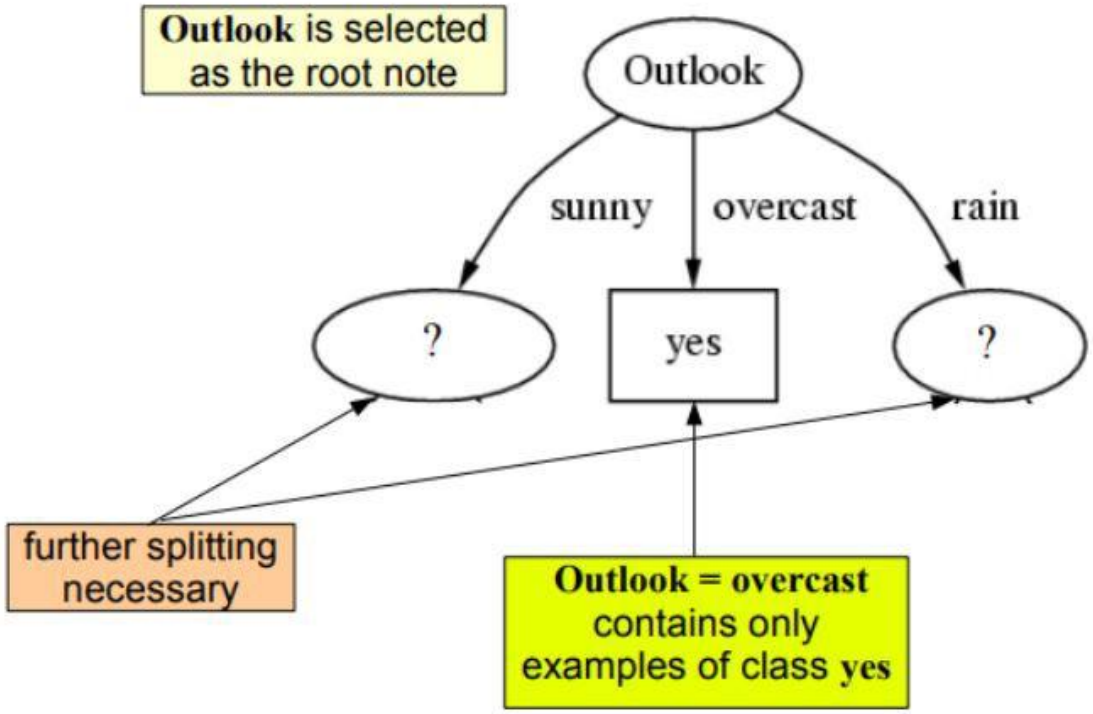
Similarly, calculate the entropy and the information gain for Humidity and Temperature

Example

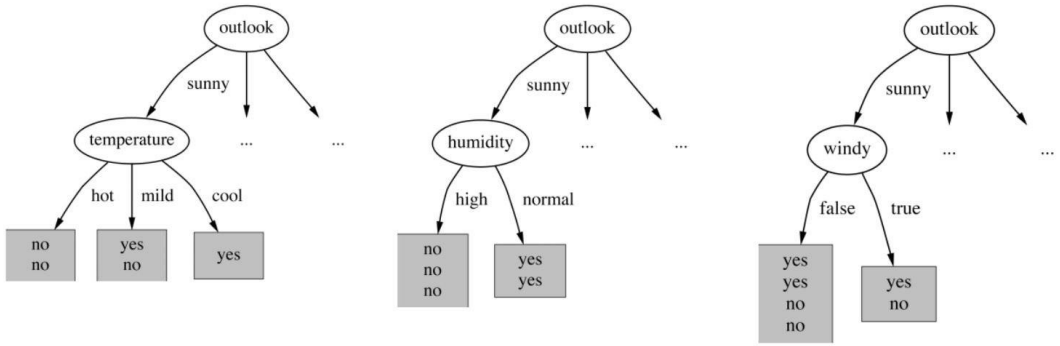
Pick the highest Gain attribute

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048

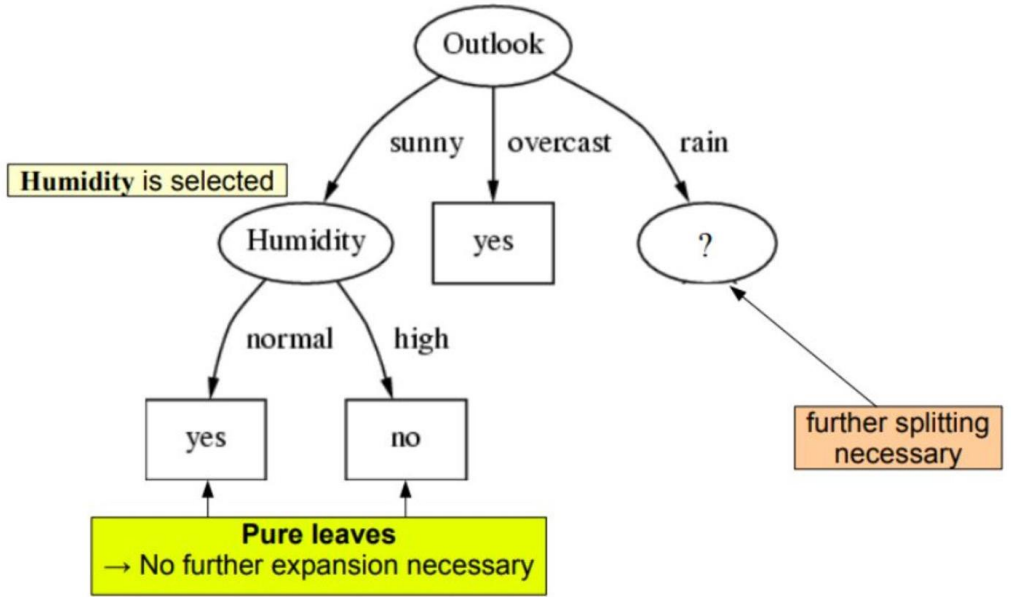
Example


Outlook is selected as the root note
Outlook
sunny
overcast
rain
?
yes
?
further splitting necessary
Outlook = overcast contains only examples of class yes

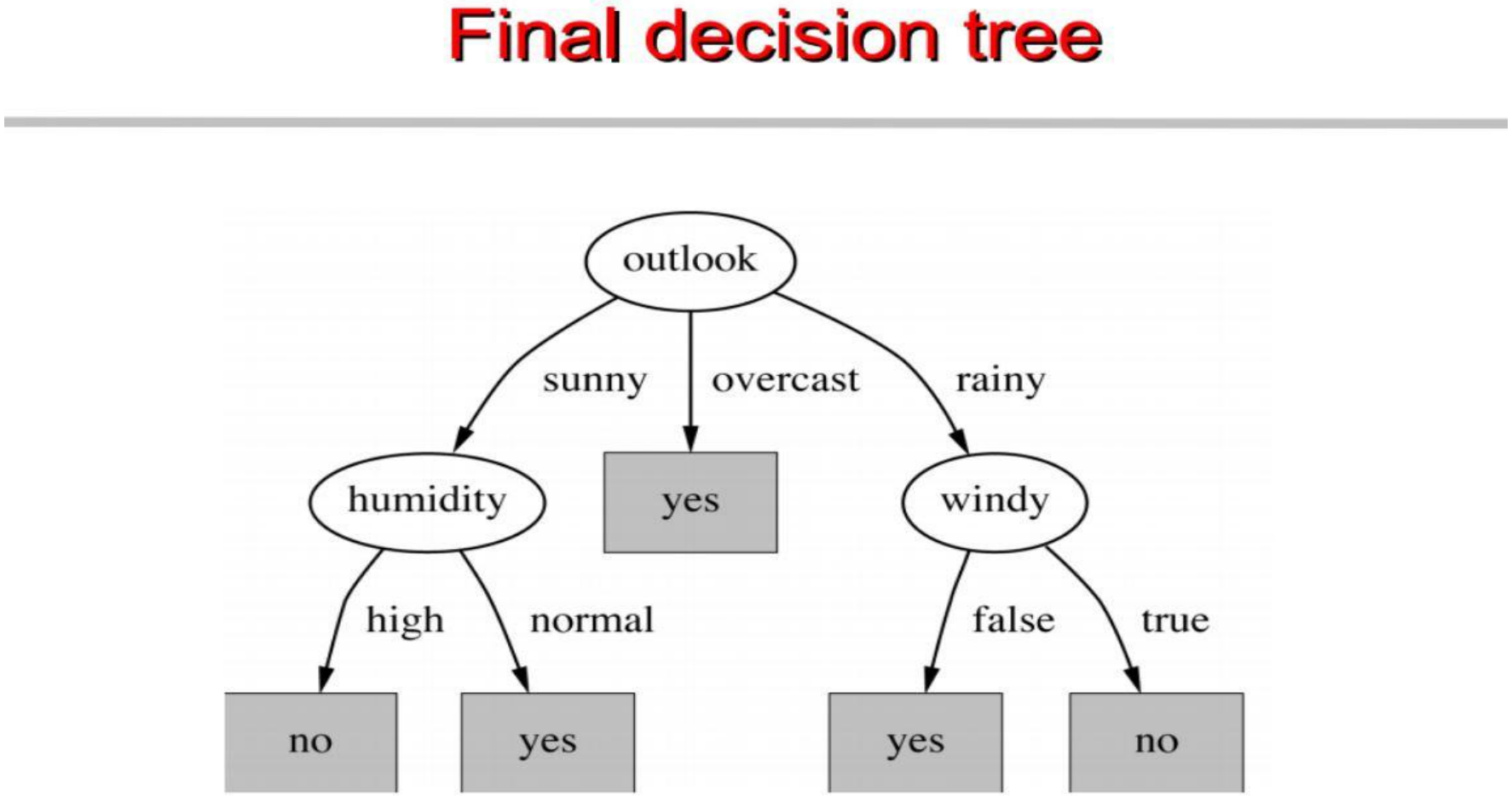
Example



$\text{Gain}(\textit{Temperature}) =$
$\text{Gain}(\textit{Humidity}) =$
$\text{Gain}(\textit{Windy}) =$



Example


Final decision tree
outlook
sunny
overcast
rainy
humidity
yes
windy
high
normal
false
true
no
yes
yes
no

Information Gain

- We use information gain (recursively) to decide what feature to split on.
- However…

Information gain favors attributes with many attribute values

- If *A* has random values (SS#), but ends up with only 1 example in each partition, it would have maximum information gain, though a terrible choice.
- Occam's razor would suggest seeking trees with fewer overall nodes. Thus, attributes with fewer possible values might be given some kind of preference.
- Binary attributes (CART) are one solution, but lead to deeper trees, and somewhat higher complexity in possible ways of splitting attributes
- Can use a penalty for attributes with many values such as Laplacian: $(n_c+1)/(n+|C|)$, though real issue is splits with little data
- Gain Ratio is the approach used in original ID3/C4.5

Gain Ratio

Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \quad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

(where parent Node, p is split into k partitions, and
n_i is number of records in partition i)

- Designed to overcome the disadvantage of GAIN
- Adjusts GAIN by the entropy of the partitioning (SplitINFO)
- **Higher entropy partitioning (large number of small partitions) is penalized**
- Used by C4.5 (an extension of ID3)

Determining Splits

How to Specify Test Condition?

Depends on attribute types

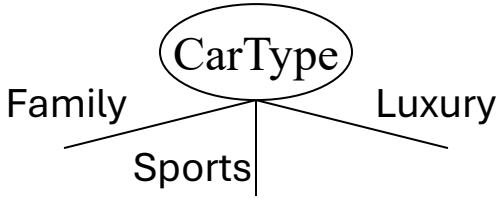
- Nominal
- Ordinal
- Continuous

Depends on number of ways to split

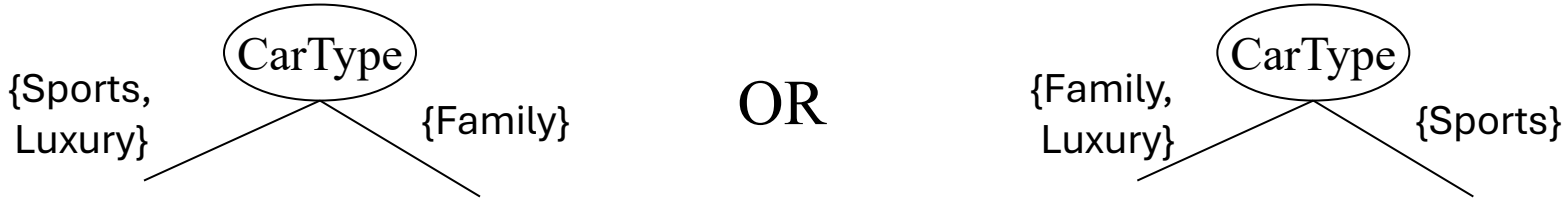
- Binary split
- Multi-way split

Splitting Based on Nominal Attributes

Multi-way split: Use as many partitions as values



Binary split: Divide values into two subsets



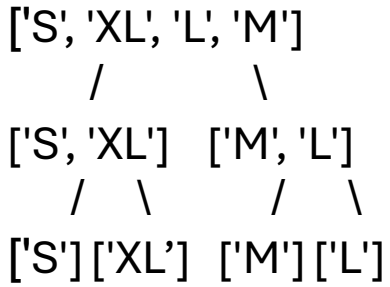
Need to find optimal partitioning!

For Ordinal attributes, make sure the splitting doesn't violate the order.

Splitting Based on Ordinal Values

- Ordinal values imply ordering
- Can just split like nominal but force ordering
- Does ordering matter?

Consider predicting weight based on T-shirt size



If we don't consider ordering, the tree must become deeper.

We could be fairly accurate with just one level if we considered ordering.

Splitting Based on Continuous Attributes

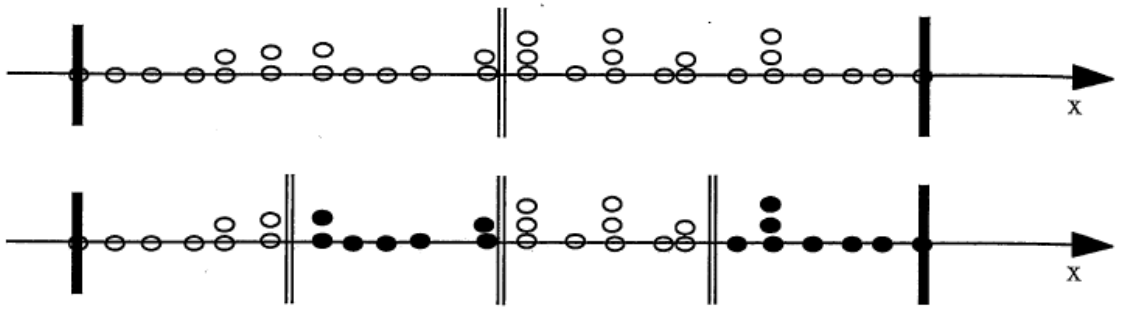
- Different ways of handling
  - Multi-way split: form ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – repeat on each new partition
  - Binary split: $(A < v)$ or $(A \geq v)$
    - How to choose v?

Need to find optimal partitioning!

Can use GAIN or GINI !

Real Valued Features

- C4.5: Continuous data is handled by testing all *n*-1 possible binary thresholds for each continuous feature to see which gives best information gain. The split point with highest gain gives the score for that feature which then competes with all other features.
  - More efficient to just test thresholds where there is a change of classification.
  - Is binary split sufficient? Attribute may need to be split again lower in the tree, no longer have a strict depth bound



ID3 vs CART

ID3 vs CART

	ID3	CART
Full Name	Iterative Dichotomiser 3	Classification & Regression Trees
Metric	**Entropy** (Information Gain)	**Gini Impurity**
The Split	**Multi-way Splits** (One branch per category).	**Binary Splits** (Always True/False)
Data Types	Handles **Categories** well. Struggles with Numbers (better in 4.5).	Handles **Numbers** & Categories.
Capabilities	Classification ONLY.	Classification **AND Regression** (we'll talk about this later)
Pros/Cons	Easy to understand, but can overfit on "ID-like" features.	Faster, standard in Scikit-Learn / XGBoost.

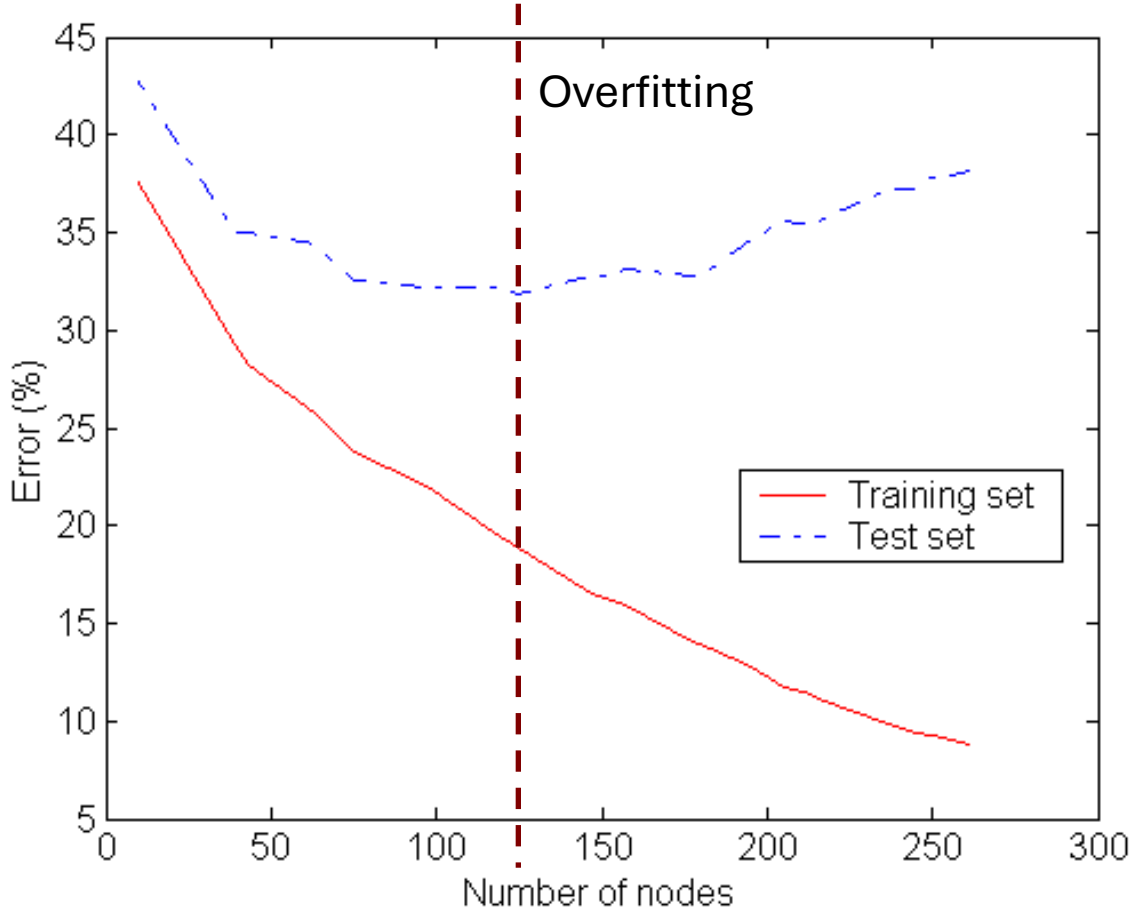
Break Time!

Overfitting and Underfitting

Overfitting and Underfitting

- Overfitting:
  - Given a model space H, a specific model $h \in H$ is said to overfit the training data if there exists some alternative model $h' \in H$, such that h has smaller error than h' over the training examples, but h' has smaller error than h over the entire distribution of instances
- Underfitting:
  - The model is too simple, so that both training and test errors are large

Detecting Overfitting



Overfitting in Decision Tree Learning

Overfitting results in decision trees that are more complex than necessary.

- Tree growth went too far
- Number of instances gets smaller as we build the tree (e.g., several leaves match a single example)

Training error no longer provides a good estimate of how well the tree will perform on previously unseen records.

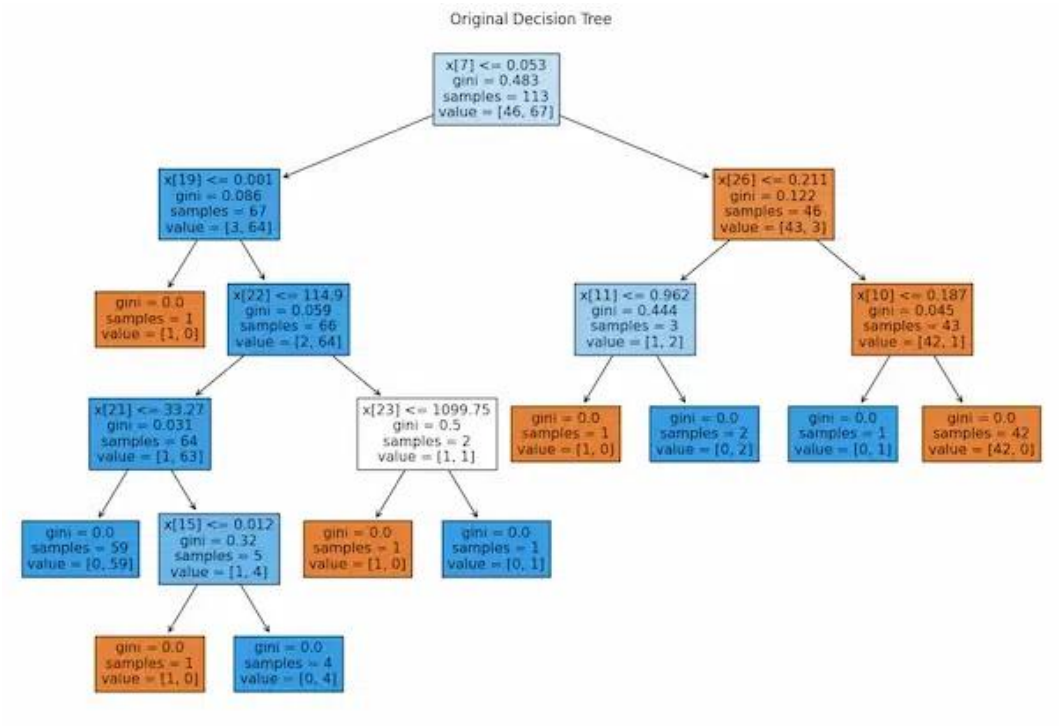
Avoiding Tree Overfitting – Solution 1

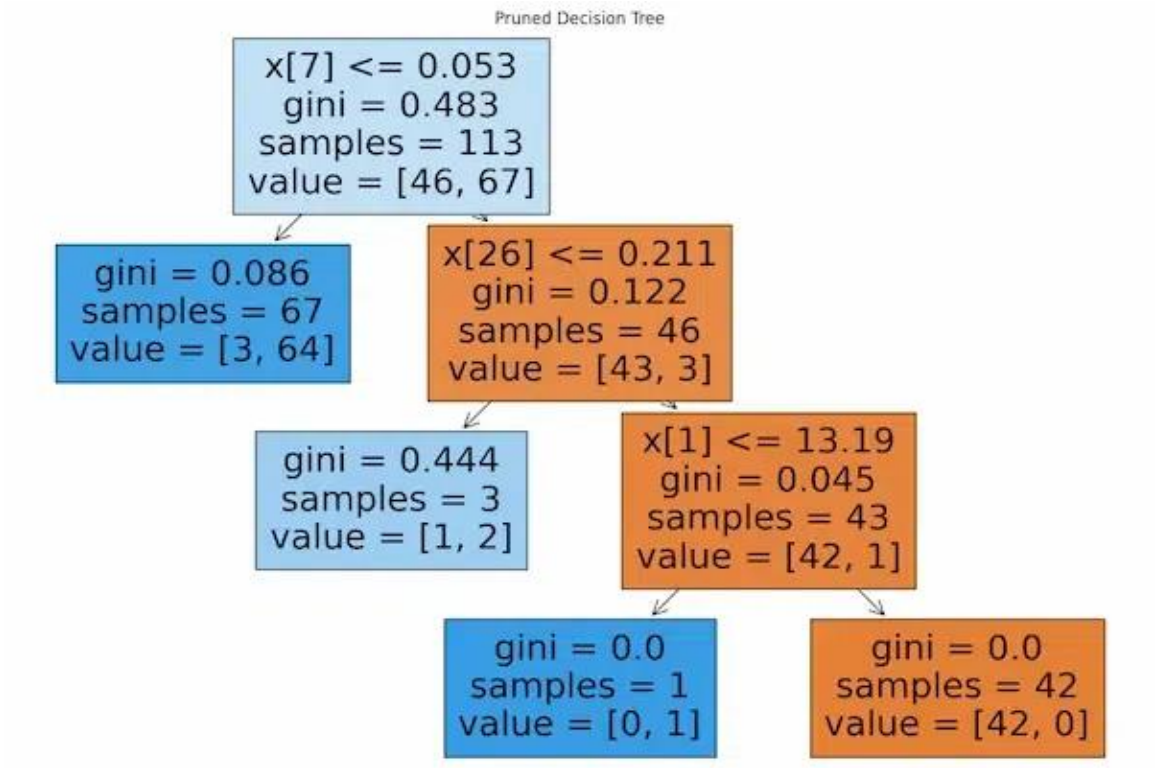
- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., GINI or GAIN)
    - Stop if reach max_depth

Avoiding Tree Overfitting – Solution 2

- Post-pruning
- Pruning a full tree (one where all possible nodes have been added)
  - Prune any nodes which would not hurt accuracy
  - Could allow some higher order combinations that would have been missed with early stopping
  - Can simultaneously consider all nodes for pruning rather than just the current frontier

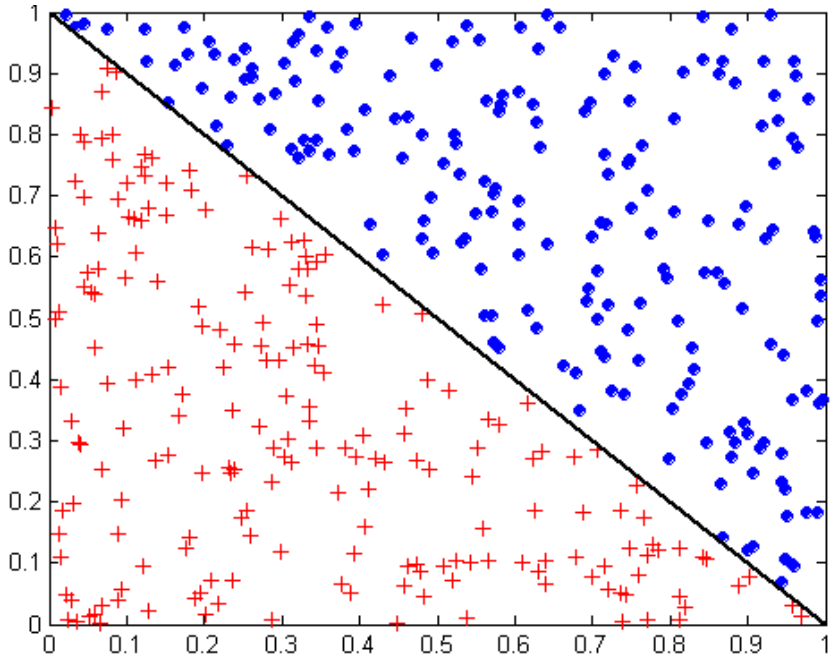
1. Train tree out fully (empty or consistent partitions or no more attributes)
2. For EACH non-leaf node, test accuracy on a validation set for a modified tree where the sub-trees of the node are removed and the node is assigned the majority class based on the instances it represents from the training set
3. Keep pruned tree which does best on the validation set and does at least as well as the current tree on the validation set
4. Repeat until no pruned tree does as well as the current tree

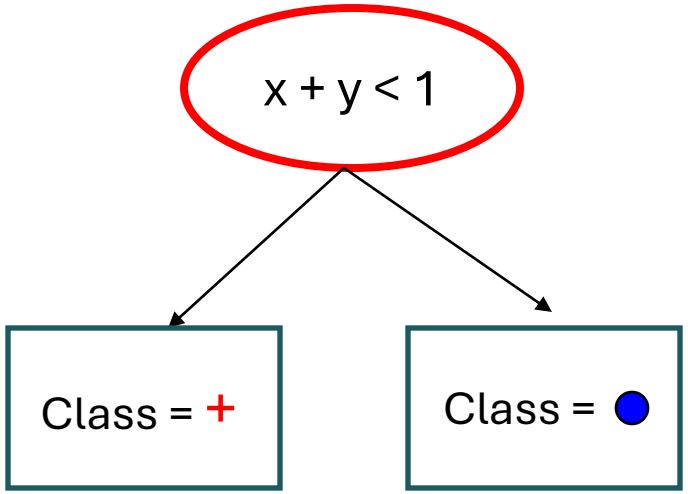

Original Decision Tree
x[7] <= 0.053
gini = 0.483
samples = 113
value = [46, 67]
x[19] <= 0.001
gini = 0.086
samples = 67
value = [3, 64]
x[26] <= 0.211
gini = 0.122
samples = 46
value = [43, 3]
gini = 0.0
samples = 1
value = [1, 0]
x[22] <= 114.9
gini = 0.059
samples = 66
value = [2, 64]
x[11] <= 0.962
gini = 0.444
samples = 3
value = [1, 2]
x[10] <= 0.187
gini = 0.045
samples = 43
value = [42, 1]
x[21] <= 33.27
gini = 0.031
samples = 64
value = [1, 63]
x[23] <= 1099.75
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 42
value = [42, 0]
gini = 0.0
samples = 59
value = [0, 59]
x[15] <= 0.012
gini = 0.32
samples = 5
value = [1, 4]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 4
value = [0, 4]


Pruned Decision Tree
x[7] <= 0.053
gini = 0.483
samples = 113
value = [46, 67]
gini = 0.086
samples = 67
value = [3, 64]
x[26] <= 0.211
gini = 0.122
samples = 46
value = [43, 3]
gini = 0.444
samples = 3
value = [1, 2]
x[1] <= 13.19
gini = 0.045
samples = 43
value = [42, 1]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 42
value = [42, 0]

Decision Tree Miscellaneous

Oblique Decision Trees





- Test condition may involve multiple attributes. We said we can **only split on one attribute.**
- There's a way around that.
- More expressive representation
- Finding optimal test condition is computationally expensive

Computational Complexity

- Attributes which best discriminate between classes are chosen
- Complexity:
  - At each tree node with a set of instances the work is
    - O(|*Instances*| * |*remaining attributes*|), which is Polynomial
  - Total complexity is empirically polynomial
    - O(|*TrainingSet*| * |*attributes*| * |*nodes in the tree*|)
    - where the number of nodes is bound by the number of attributes and can be kept smaller through stopping criteria, etc.
- Keep in mind the number of instances
  - It may be O(|*Instances*| * |*attributes*|) but … Big Data

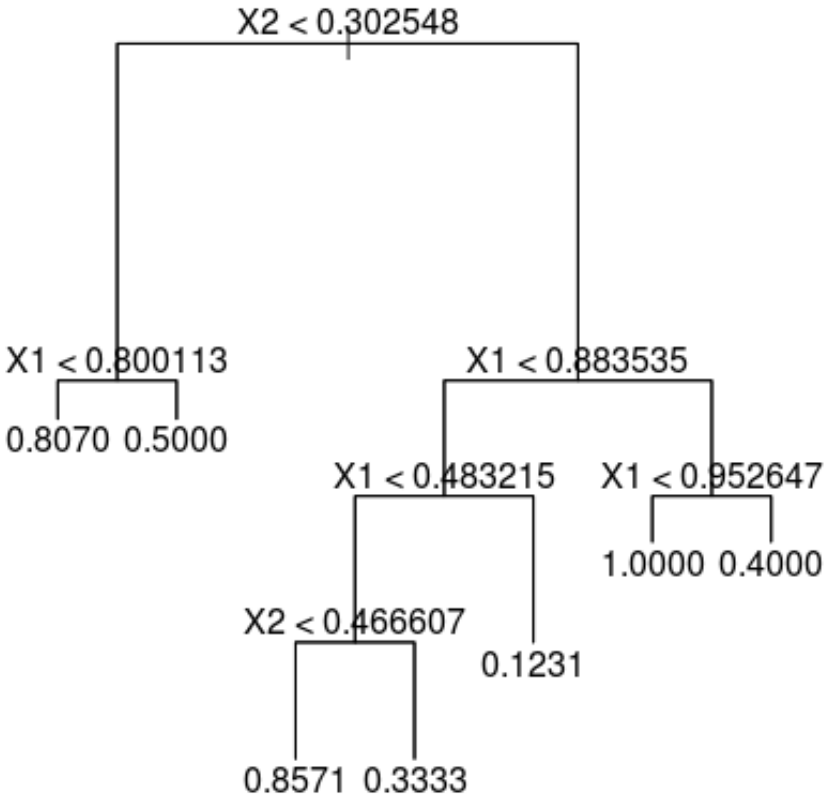
DT Interpretability

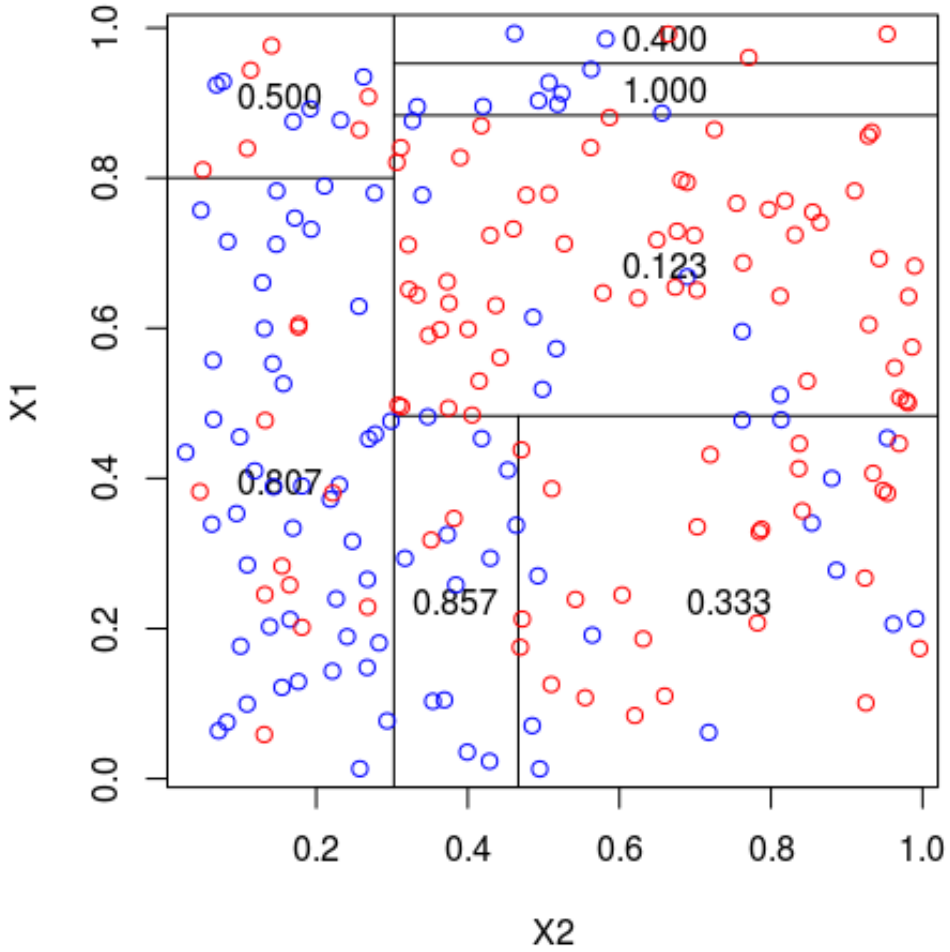
- Intelligibility of DT – When trees get large, intelligibility drops off
- How critical is intelligibility in general?
  - Will truly hard problems have a simple explanation?

Decision Tree Regression

Regression

- DecisionTreeRegressor
- Regression Tree – The output value for a leaf is just the average of the outputs of the instances represented by that leaf
  - Could adjust for outliers, etc.
- For regression training the score for a node is not GINI/Info impurity, but is the SSE of instances represented by the node
- The feature score for a potential split is the weighted sum of the two child nodes scores (SSE)
- Then, just like with classification, we choose the lowest score amongst all possible feature splits
- As long as there is significant variance in node examples, splitting will continue


X2 < 0.302548
X1 < 0.300113
X1 < 0.883535
0.8070
0.5000
X1 < 0.483215
X1 < 0.952647
1.0000
0.4000
X2 < 0.466607
0.1231
0.8571
0.3333


X1
X2
0.0
0.2
0.4
0.6
0.8
1.0
0.500
0.400
1.000
0.123
0.807
0.857
0.333

Decision Trees - Conclusion

- Good Empirical Results
- Comparable application robustness and accuracy with MLPs
- Fast learning since no iterations
- One of the most used and well known of current symbolic systems
- Can be used as a feature filter for other algorithms – Attributes higher in the tree are best, those rarely used can be dropped

Code Time