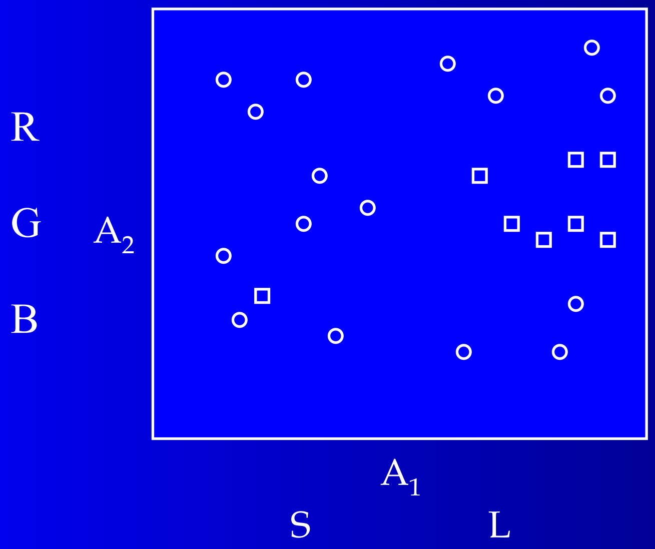
DECISION TREES

Decision Trees

- Highly used and successful
- Iteratively split the Data Set into subsets one feature at a time, using greedy most informative features first
  - Thus, constructively chooses which features to use and ignore
- Features – discrete/nominal (can extend to continuous features)
- Smaller/shallower trees generalize the best (i.e. using just the most informative features)
  - Searching for smallest tree takes exponential time

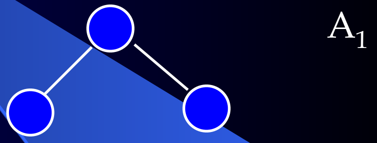
Decision Tree Learning

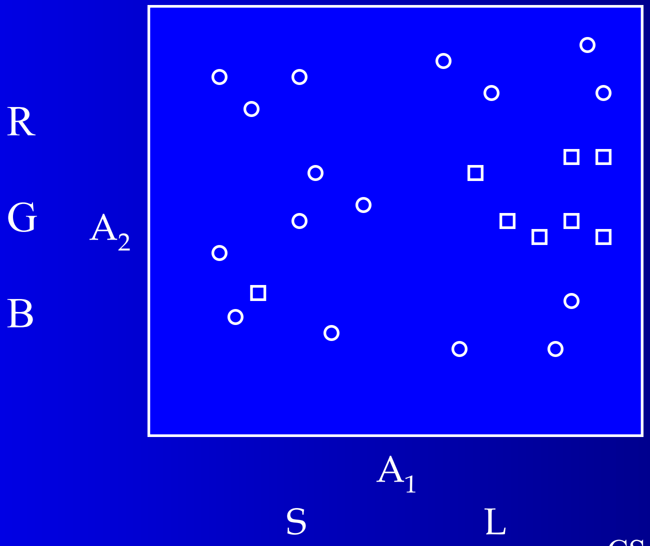
- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- A goal is to get "pure" leaf nodes. What feature would you split on?

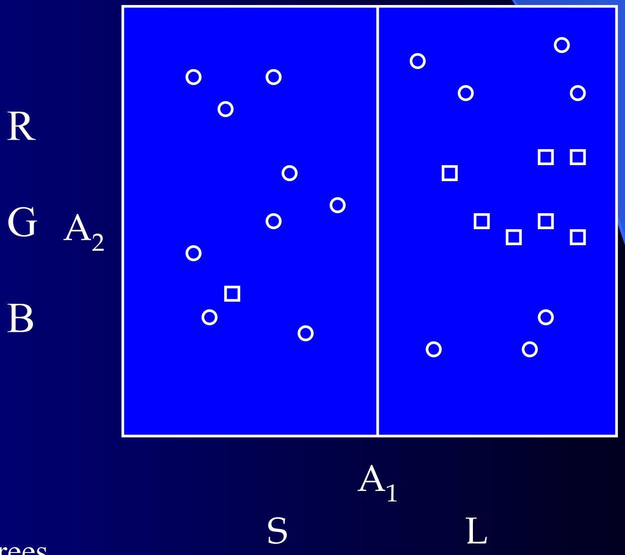


Decision Tree Learning

- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Next step for left and right children?

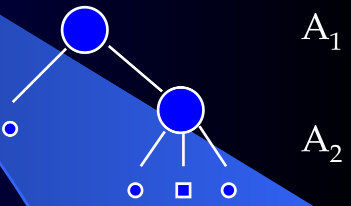


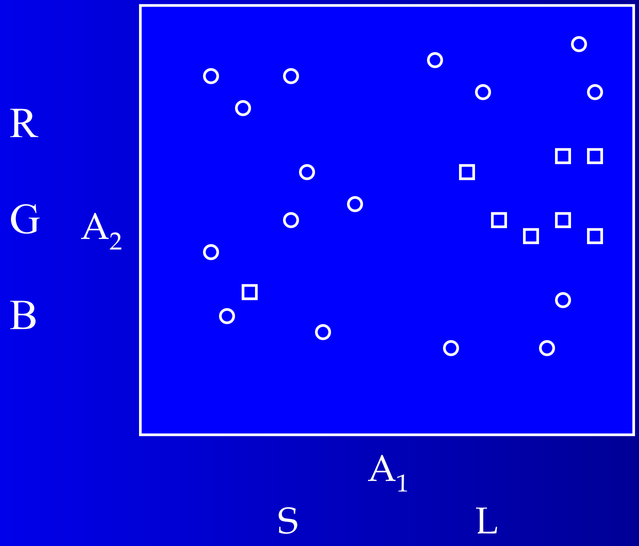


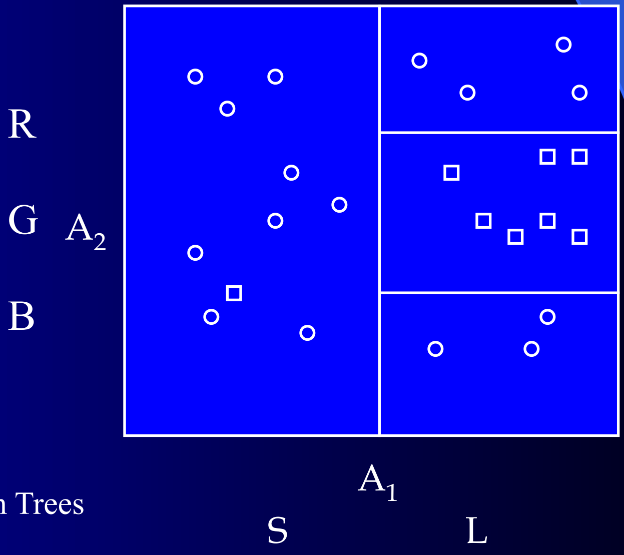


Decision Tree Learning

- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Decision surfaces are axis aligned Hyper-Rectangles
- Label leaf nodes with their majority class







Information

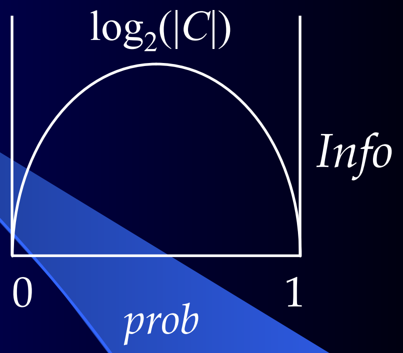
- Information of a message in bits: $I(m) = -\log_2(p_m)$
- If there is a dataset S of C classes, then information for one class c is: $I(c) = -\log_2(p_c)$
- Total info of the data set is just the sum (of the info per class times the proportion of that class)

$$\text{Info}(S) = \text{Entropy}(S) = -\sum_i p_i \cdot \log_2 p_i$$

- Info(S) is the average amount of information needed to identify the class of a data instance in set S

Information Gain Metric

- $\text{Info}(S) = \text{Entropy}(S) = -\sum_i p_i \cdot \log_2 p_i$
- $0 \le \text{Info}(S) \le \log_2(|C|)$, $|C|$ is # of output classes
- Mostly pure sets have $\text{Info}(S) = \text{Entropy}(S) \approx 0$
- $\text{Gain}(S, F) = \text{Info}(S) - \text{Info}_F(S)$ (i.e. minimize $\text{Info}_F(S)$)



Decision Tree Algorithms

- J Ross Quinlan – Australia, ML researcher
  - ID3 (Iterative Dichotimiser 3) – 1986
  - C4.5 – (Version 4.5 written in C) 1993, Handles real valued inputs
  - C5.0 – More efficient implementation
- Leo Breiman - UC Berkeley
  - CART (Classification and Regression Trees) – 1984
    - This is the decision tree approach currently supported in Sklearn
  - Random Forests - 2001
- Independently discovered

ID3/C4.5 Learning Approach

- S is a set of data instances
- Test on feature F partitions S into $\{S_1, S_2, \ldots, S_{|F|}\}$ where $|F|$ is the number of values F can take on
- Start with the training set as S and first find a *good* feature F for the root
- Continue recursively until either all subsets well classified, you run out of features, or some stopping criteria is reached

ID3/C4.5 Learning Algorithm

1. S = Training Set
2. Calculate gain for each remaining feature: Gain(S, F) = Info(S) - Info$_F$(S)
3. Select feature with highest gain and create a new node for each partition
4. For each new child node
   - if pure (one class), or if no features remain, or if stopping criteria met (e.g. pure enough, too small a number of examples remaining, not enough information gain, max depth reached), then label node with majority class and end
   - else repeat at step 2 with remaining features and training set

$$Info(S) = -\sum_{i=1}^{|C|} p_i log_2 p_i$$

$$\text{Info}_F(S) = \sum_{j=1}^{|F|} \frac{|S_j|}{|S|} \cdot \text{Info}(S_j)$$

Where S is the remaining training set at the node
$|F|$ is the number of values for the feature F
$|C|$ is the number of output classes for the task

Example

$$Info(S) = -\sum_{i=1}^{|C|} p_i log_2 p_i$$

$$\text{Info}_F(S) = \sum_{j=1}^{|F|} \frac{|S_j|}{|S|} \cdot \text{Info}(S_j)$$

$$\text{Gain}(S, F) = \text{Info}(S) - \text{Info}_F(S)$$

Meat N,Y	Crust D,S,T	Veg N,Y	Quality B,G,Gr
Y	Thin	N	Great
N	Deep	N	Bad
N	Stuffed	Y	Good
Y	Stuffed	Y	Great
Y	Deep	N	Good
Y	Deep	Y	Great
N	Thin	Y	Good
Y	Deep	N	Good
N	Thin	N	Bad

- Info(S) = $- 2/9 \cdot \log_2 2/9 - 4/9 \cdot \log_2 4/9 - 3/9 \cdot \log_2 3/9 = 1.53$
- Starting with all instances, calculate gain for each feature
- Start on feature Meat
- Info$_{\text{Meat}}$(S) = ?
  - Information Gain is ?

Example

$$Info(S) = -\sum_{i=1}^{|C|} p_i log_2 p_i$$

$$\text{Info}_F(S) = \sum_{j=1}^{|F|} \frac{|S_j|}{|S|} \cdot \text{Info}(S_j)$$

$$\text{Gain}(S, F) = \text{Info}(S) - \text{Info}_F(S)$$

Meat N,Y	Crust D,S,T	Veg N,Y	Quality B,G,Gr
Y	Thin	N	Great
N	Deep	N	Bad
N	Stuffed	Y	Good
Y	Stuffed	Y	Great
Y	Deep	N	Good
Y	Deep	Y	Great
N	Thin	Y	Good
Y	Deep	N	Good
N	Thin	N	Bad

- Info(S) = - 2/9·log$_2$2/9 - 4/9·log$_2$4/9 -3/9·log$_2$3/9 = 1.53
- Starting with all instances, calculate gain for each feature
- Info$_{\text{Meat}}$(S) = 4/9·(-2/4log$_2$2/4 - 2/4·log$_2$2/4 - 0·log$_2$0/4) + 5/9·(-0/5·log$_2$0/5 - 2/5·log$_2$2/5 - 3/5·log$_2$3/5) = .98
  – Information Gain is 1.53 - .98 = .55

Challenge Question

$$Info(S) = -\sum_{i=1}^{|C|} p_i log_2 p_i$$

$$\text{Info}_F(S) = \sum_{j=1}^{|F|} \frac{|S_j|}{|S|} \cdot \text{Info}(S_j)$$

$$\text{Gain}(S, F) = \text{Info}(S) - \text{Info}_F(S)$$

- What is the information for crust $\text{Info}_{\text{Crust}}(S)$:
  A. .98
  B. 1.35
  C. .12
  D. 1.41
  E. None of the Above

- Is it a better feature to split on than Meat?

Meat N,Y	Crust D,S,T	Veg N,Y	Quality B,G,Gr
Y	Thin	N	Great
N	Deep	N	Bad
N	Stuffed	Y	Good
Y	Stuffed	Y	Great
Y	Deep	N	Good
Y	Deep	Y	Great
N	Thin	Y	Good
Y	Deep	N	Good
N	Thin	N	Bad

Decision Tree Example

$$Info(S) = -\sum_{i=1}^{|C|} p_i log_2 p_i$$

$$\text{Info}_F(S) = \sum_{j=1}^{|F|} \frac{|S_j|}{|S|} \cdot \text{Info}(S_j)$$

$$\text{Gain}(S, F) = \text{Info}(S) - \text{Info}_F(S)$$

Meat N,Y	**Crust D,S,T**	**VegN ,Y**	**Quality B,G,Gr**
Y	Thin	N	Great
N	Deep	N	Bad
N	Stuffed	Y	Good
Y	Stuffed	Y	Great
Y	Deep	N	Good
Y	Deep	Y	Great
N	Thin	Y	Good
Y	Deep	N	Good
N	Thin	N	Bad

- $\text{Info}_{\text{Meat}}(S) = 4/9\cdot(-2/4\log_2 2/4 - 2/4\cdot\log_2 2/4 - 0\cdot\log_2 0/4) +$
  $5/9\cdot(-0/5\cdot\log_2 0/5 - 2/5\cdot\log_2 2/5 - 3/5\cdot\log_2 3/5) = .98$

- $\text{Info}_{\text{Crust}}(S) = 4/9\cdot(-1/4\log_2 1/4 - 2/4\cdot\log_2 2/4 - 1/4\cdot\log_2 1/4) +$
  $2/9\cdot(-0/2\cdot\log_2 0/2 - 1/2\cdot\log_2 1/2 - 1/2\cdot\log_2 1/2) +$
  $3/9\cdot(-1/3\cdot\log_2 1/3 - 1/3\cdot\log_2 1/3 - 1/3\cdot\log_2 1/3) = 1.41$

- Meat leaves less info (higher gain) and thus is the better of these two

Decision Tree Homework

$$Info(S) = -\sum_{i=1}^{|C|} p_i log_2 p_i$$

$$\text{Info}_F(S) = \sum_{j=1}^{|F|} \frac{|S_j|}{|S|} \cdot \text{Info}(S_j)$$

$$\text{Gain}(S, F) = \text{Info}(S) - \text{Info}_F(S)$$

Meat N,Y	**Crust** D,S,T	**Veg** N,Y	**Quality** B,G,Gr
Y	Thin	N	Great
N	Deep	N	Bad
N	Stuffed	Y	Good
Y	Stuffed	Y	Great
Y	Deep	N	Good
Y	Deep	Y	Great
N	Thin	Y	Good
Y	Deep	N	Good
N	Thin	N	Bad

- Finish the first level, find the best feature and split
- Then find the best feature for the left most node at the second level and split the node accordingly
  - Assume sub-nodes are sorted alphabetically left to right by feature
  - Label any leaf nodes with their majority class
  - You could/should continue with the other nodes to get more practice

DT Interpretability

- Intelligibility of DT – When trees get large, intelligibility drops off
- C4.5 - transforms tree into prioritized rule list. It does simplification of irrelevant features by greedy elimination strategy. Prunes less productive rules within rule classes.
- How critical is intelligibility in general?
  - Will truly hard problems have a simple explanation?

Decision Tree Overfit Avoidance

- Noise can cause inability to converge 100% or can lead to overly complex decision trees (overfitting). Thus, we usually allow leafs with multiple classes.
  - Can select the majority class as the output, or output a confidence vector
- Also, may not have sufficient features to perfectly divide data
- Common approach is to not split when the number of examples at the node are less than a threshold and just label this leaf node with its majority class – early stopping
- Could use a validation set and only add a new node if improvement (or no decrease) in accuracy on the validation set – checked independently at each branch of the tree using data set from parent
  - But shrinking data problem with decision trees

Reduced Error 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

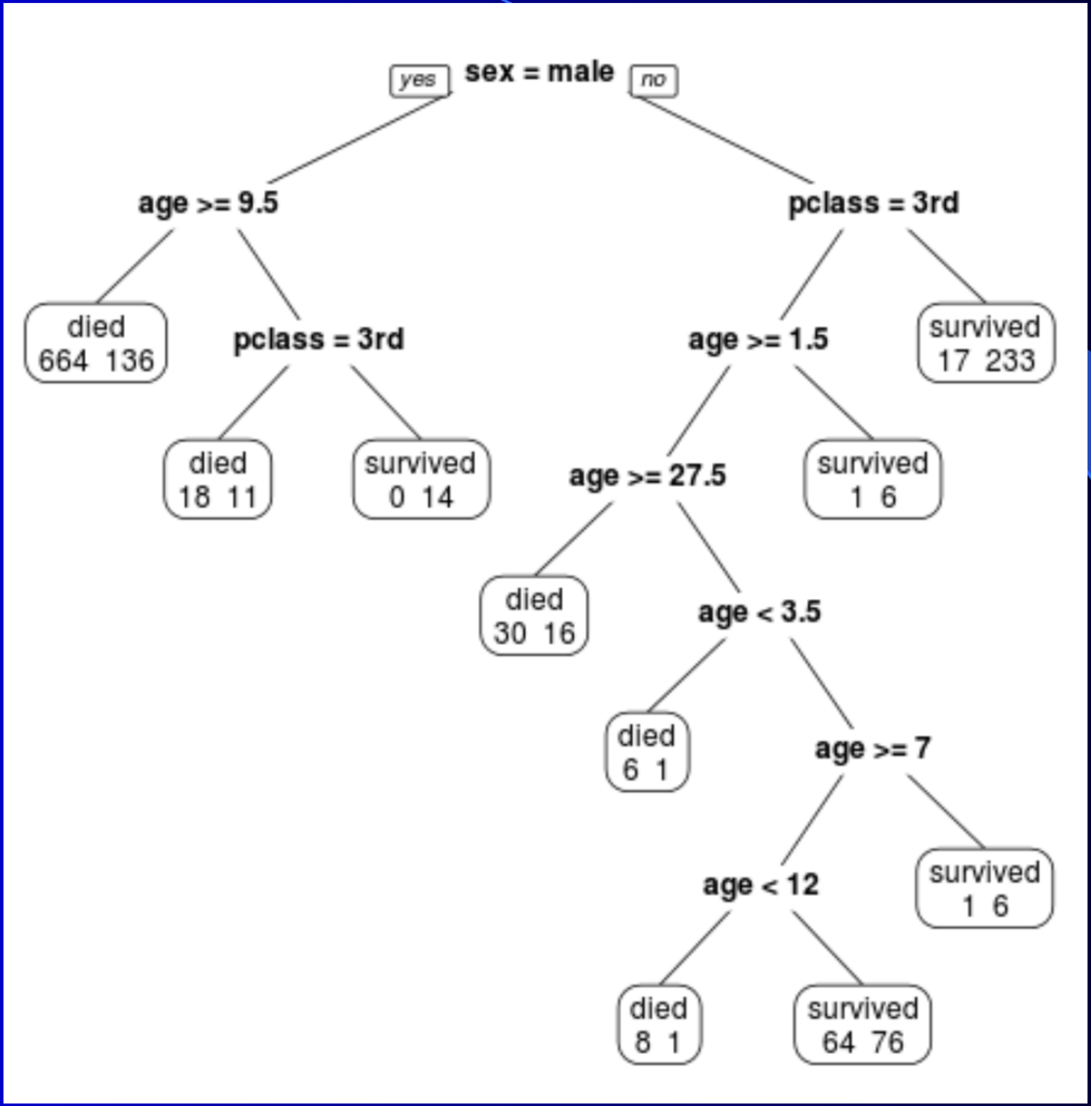
1. Train tree out fully (empty or consistent partitions or no more features)
2. For EACH non-leaf node, test accuracy on a validation set for a modified tree where the sub-trees of the node are removed and the node is assigned the majority class based on the instances it represents from the training 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

CLASSIFICATION AND REGRESSION TREES

CART – Classification and Regression Trees

- Binary Tree – Considers *all* possible splits, also with nominals
  - Color = blue (vs not blue), Height >= 60 inches
  - Recursive binary splitting – Same feature could be split multiple times
  - No preset limit on depth like with C4.5 with nominal features
    - Does avoid bushy split problem of C4.5 (SS#)

Titanic Survival Dataset



Information gain favors features with many feature values

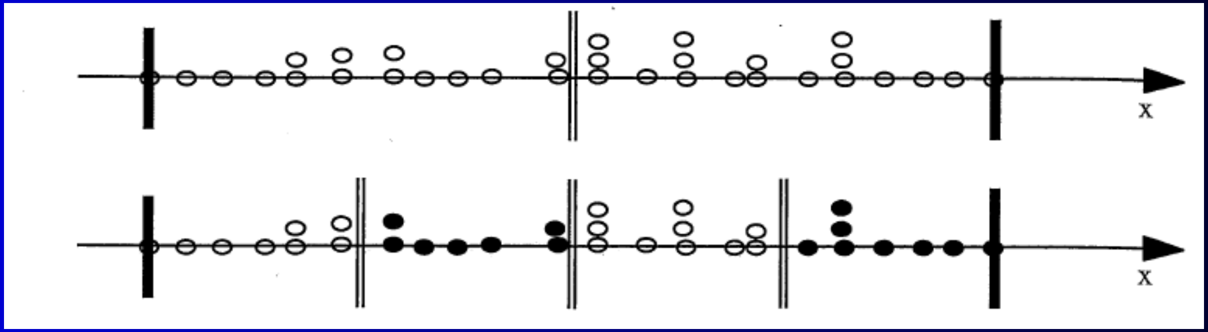
- If *F* has random values (SS#), but ends up with only 1 example in each partition, it would have maximum information gain, though a terrible choice.
- Occam's razor would suggest seeking trees with less overall nodes. Thus, features with less possible values might be given some kind of preference.
- Binary features (CART) are one solution, but lead to deeper trees, and somewhat higher complexity in possible ways of splitting features

Gini Impurity

- For classification CART uses Gini impurity for node score
- Gini score for a node is: $G = 1 - \sum_{i=1}^{|C|} p_i^2$
  - p_i is percentage of leaf's instances with output class i
  - Best case is 0 (all one class), worse is 1-1/|C| (equal percentage of each)
- Total score for a given split is the weighted sum of the two sub-node G's

Real Valued Features

- 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? Feature may need to be split again lower in the tree, no longer have a strict depth bound



CART Regression

- 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.
  - Could do the same with C4.5
- For regression the score for a node is not GINI 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

CART Overfit Avoidance

- Most common approach is to stop when there are only a small number of examples at a node which is a type of early stopping
  - Hyperparameter - don't split further when less than (e.g. 5, 10)
- Can also constrain the tree to be smaller (max nodes, max depth, etc.) but could cause underfit! (like using less hidden nodes in a MLP)
- Can use pruning after full learning for regularization
  - Sklearn has a different pruning algorithm for CART than Reduced Error Pruning

Decision Trees - Conclusion

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

Decision Tree Lab

- Nominals – SKlearn CART only accepts numeric features - Fits CART fine since that is how CART thinks of nominal features anyways, breaking them into separate one-hot features for each possible feature value

- So a nominal feature Color with 3 feature values (Red, Green, Blue), would be represented as 3 one-hot features
  - Is-Red, Is-Green, Is-Blue
  - Binary features can just be represented as 0/1

- Note that Color could appear multiple times in a branch unlike in C4.5. A not Is-Red branch could later consider Is-Blue or Is-Green.