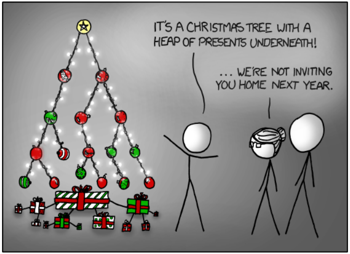


Image. xkcd comics - # 835

Data Mining: Classification: Part 4

Decision Trees

CS 4821 - CS 5831 - s24

Some slides adapted from P. Smyth; A. Moore, D. Klein Han, Kamber, Pei; Tan, Steinbach, Kumar; L. Kaebling; R. Tibshirani; T. Taylor; and L. Hannah

Decision Trees

Decision Trees

Decision Trees are one of the most popular and useful data mining method

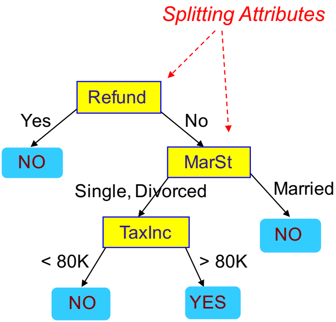
- Pros:
    - can handle nominal, ordinal, and numeric inputs
    - speed and scalability
    - robustness to outliers and missing values
    - **intrepretability / visualization**
    - compactness of classification rules
    - trees can be used for regression or classification
    - very popular in industry
- Cons
    - several tuning parameters to set
    - decision boundary is non-continuous
    - instability in learning

Example 1: Decision Trees

categorical categorical continuous class

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

Training Data

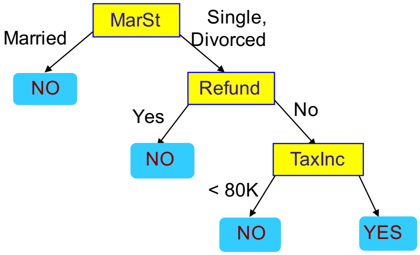


Model: Decision Tree

Example 1b: Same Data, Different Tree

categorical categorical continuous class

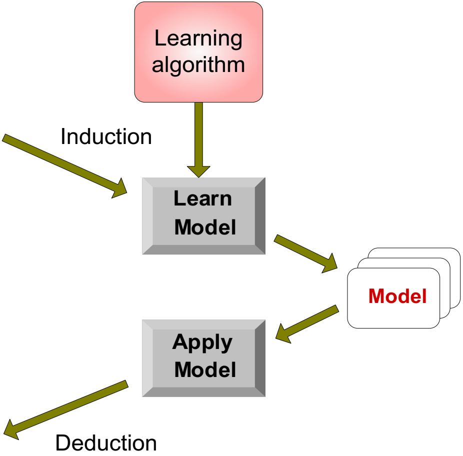
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



Decision Tree: Deduction

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

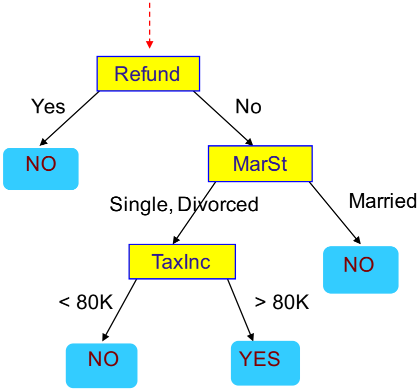


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Example 2: Apply Model to Test Data

Start from the root of tree.



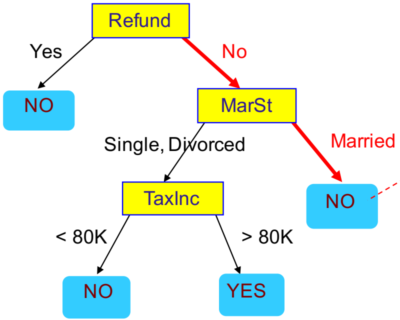
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Example 2: Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



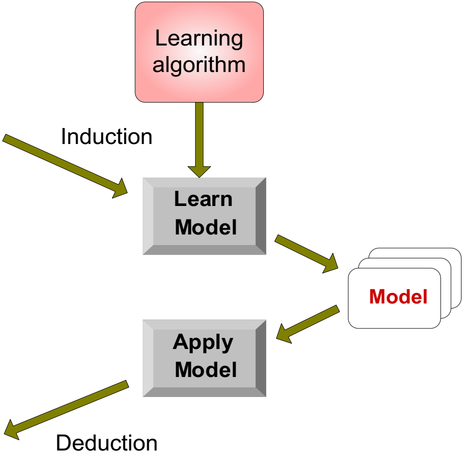
Assign Cheat to “No”

Decision Tree Induction

Decision Tree Induction

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Induction Methods

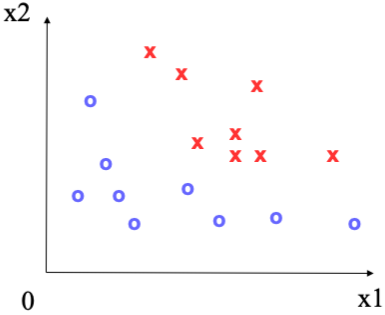
There are **many** Decision Tree (DT) induction algorithms:

- Hunt's Algorithm
- CART (Classification and Regression Trees)
- ID3, C4.5, C5.0, C5.5, ... (Quinlin, information gain)
- CHAID (CHi-squared Automatic Interaction Detection)
- MARS
- SLIQ, SPRINT
- ... and many more

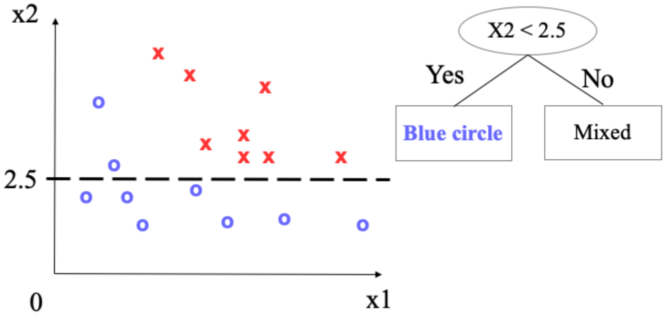
Intuition:

Use attributes to split the data recursively, until each split contains only a single class.

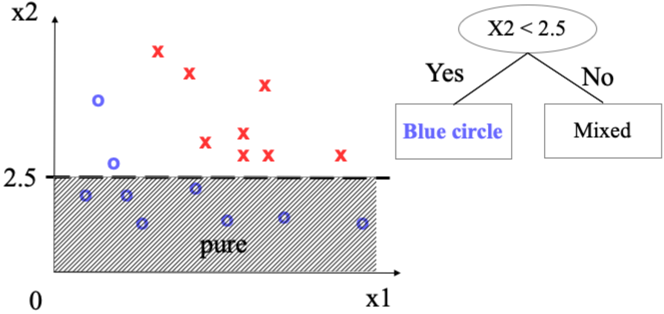
Example 3: Creating a Decision Tree



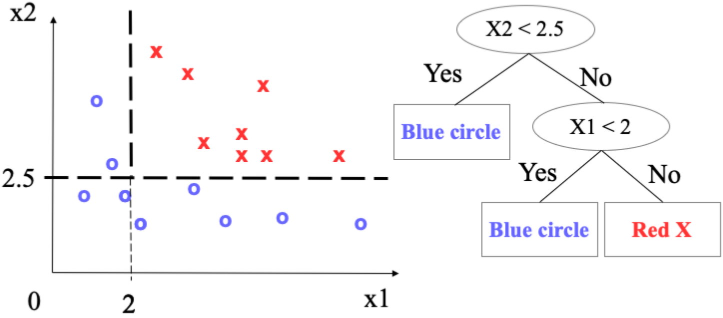
Example 3: Creating a Decision Tree



Example 3: Creating a Decision Tree



Example 3: Creating a Decision Tree



Decision Tree Hyper-parameters

Recursive Partitioning Method Choices

How do we learn a tree from training data?

Answer: iterative greedy splitting

Basic strategy is a top-down, recursive, divide-and-conquer method

Questions:

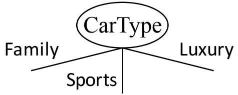
- How to split the data among different attribute types?
- How to determine the best split?
  - Entropy / information gain: ID3 - Iterative Dichotomiser 3, C4.5
  - Gini index: CART - Classification and Regression Trees
  - Classification error
- When to stop splitting?

How to Split? Different Attribute Types

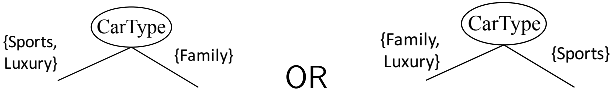
- Trees induction methods in theory can work on categorical and numeric data
- In practice R methods can work on categorical data, Python's `sklearn` does not currently support this.
  - Must use encoding in `sklearn` to support categorical attributes
- For categorical, can explore exponential number of ways to split the values

How to Split? Nominal Attributes

- Multi-way split: use as many partitions as values/categories

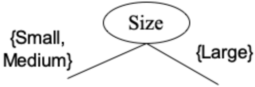


- Binary Split: divide values into two subsets; need to find optimal partitioning

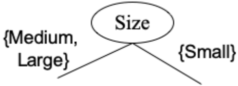


How to Split? Ordinal Attributes

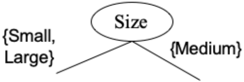
- Divide values into two subsets;
  need to find optimal partitioning



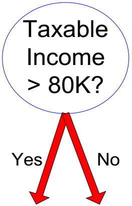
OR



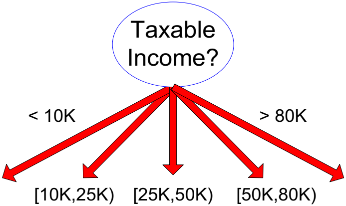
- What about this split?



Splitting Continuous Variables



(i) Binary split



(ii) Multi-way split

- Discretization to form an ordinal variable
  - Static - discretize the data once at the beginning
    Ex. equal interval, equal frequency, etc.
  - Dynamic - discretize during the tree construction
    Ex. for a binary split $(X_j \leq x')$ or $(X_j > x')$, consider all possible splits and select the best cut

How to Determine the Best Split?

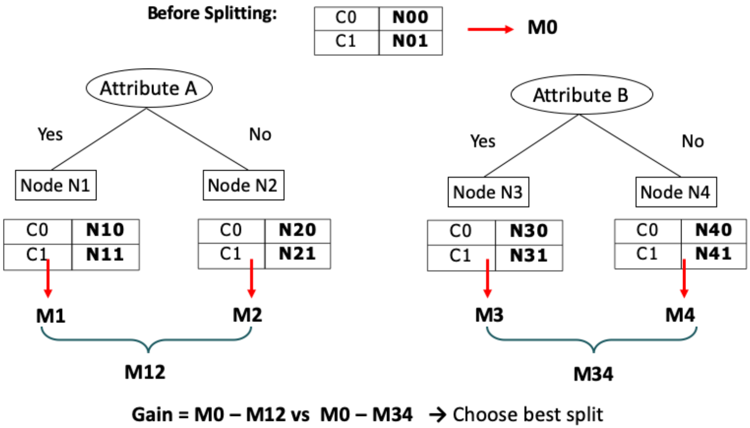
- Greedy approach
  - Nodes with homogeneous class distributions are preferred
- Need a measure of node impurity

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

- Measures of Node Impurity
  - Entropy
  - GINI Index
  - Classification Error

How to Find the Best Split?

For some measure **M** of node purity:



Entropy

Suppose node m has k classes with probabilities of:

$p_m = (p_{m,1}, p_{m,2}, \ldots, p_{m,k})$,

where $p_{m,i}$ is the relative frequency of class i at node m

Ex. data with 3 classes, a node m could have

$p_{m,1} = 0.25$, $p_{m,2} = 0.42$ and $p_{m,3} = 0.33$

Entropy: $$H(X_m) = -\sum_{i=1}^{k} p_{m,i} \log p_{m,i}$$

- Maximized when $p_{m,i} = \frac{1}{k}$ with value $\log k$
  when records are equally distribution among all classes implying least information
- Minimized, 0, when one class has all records in it
- Minimizing entropy will favor *pure* nodes

Examples with computing Entropy

$$H(X_m) = -\sum_{i=1}^{k} p_{m,i} \log p_{m,i}$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

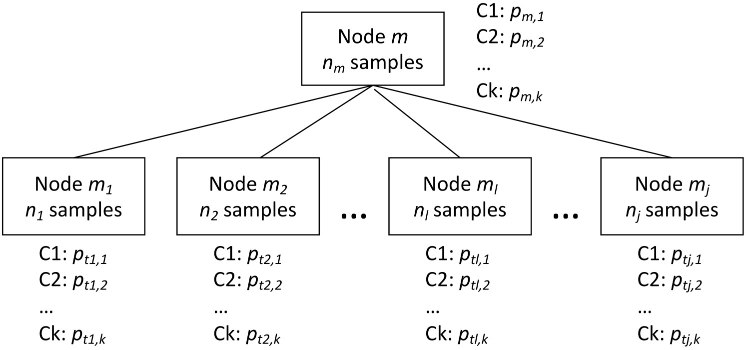
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

General Splitting Framework

- Parent node m, with n_m records and $p_m = (p_{m,1}, p_{m,2}, \ldots, p_{m,k})$ relative frequencies of classes
- Split into j new child nodes $(m_l)_{1 \leq l \leq j}$
- Each child node has n_l records and relative frequencies $(p_{m_l,1}, p_{m_l,2}, \ldots, p_{m_l,k})$.



Splitting based on Gain in Entropy

- Parent node m, with n_m records and Entropy $H(X_m)$ is to be split into j new child nodes $(m_l)_{1 \leq l \leq j}$
- Each child node, m_l, has n_l records and probs. $(p_{m_l,1}, p_{m_l,2}, \ldots, p_{m_l,k})$, and Entropy, $H(X_{m_l})$
- The gain in Entropy for this split is

$$\begin{aligned} Gain(X_m)_H &= H(X_m) - \frac{\sum_{l=1}^{j} n_l H(X_{m_l})}{\sum_{l=1}^{j} n_l} \\ &= H(X_m) - \underbrace{\left(\sum_{l=1}^{j} \frac{n_l}{n_m} H(X_{m_l}) \right)}_{\text{Ave. Entropy among } m\text{'s children}} \end{aligned}$$

- Select node with largest **gain** (reduction in Entropy, H)
- Used in ID3, C4.5, C5.0
- Disadvantage: tends to prefer splits that result in large number of partitions, each being small but pure

Splitting based on Gain Ratio

Gain Ratio

$$GainRatio = \frac{Gain(X_m)_H}{SplitInfo}; \quad SplitInfo = -\sum_{l=1}^{j} \frac{n_l}{n_m} \log \frac{n_l}{n_m}$$

- Node m split into j partitions, n_i is the number of records in partition i
- Adjusts Information Gain by the entropy of the partitioning (SplitInfo), higher entropy partitioning (large number of small partitions) is penalized
- Used in C4.5

$GINI$ Index

Suppose there are k classes and node m has probabilities:

$p_t = (p_{m,1}, p_{m,1}, \ldots, p_{m,k})$

$$GINI(X_m) = \sum_{(j,j') \in \{1,\ldots,k\}: j \neq j'} p_{m,j} p_{m,j'} = 1 - \sum_{j=1}^{k} p_{m,j}^2$$

- Maximized when $p_{m,j} = \frac{1}{k}$ with value $1 - \frac{1}{k}$
  when records are equally distributed among all classes
- Minimized, 0, when all records belong to a single class
- Minimizing GINI will favor *pure* nodes

Examples with computing $GINI$ Index

$$GINI(X_m) = 1 - \sum_{j=1}^{k} p_{m,j}^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Maximal impurity here is $\frac{1}{2} = 0.5$

Splitting based on $GINI$ Index

- Parent node m, with n_m records is to be split into j new child nodes $(m_l)_{1 \leq l \leq j}$
- Each child node, m_l, has n_l records and probs. $(p_{m_l,1}, p_{m_l,2}, \ldots, p_{m_l,k})$, and GINIs, $GINI(X_{m_l})$

- The $GINI_S plit$ (Average GINI over m's children):

$$GINI_{Split}(X_m) = \frac{\sum_{l=1}^{j} n_l GINI(X_{m_l})}{\sum_{l=1}^{j} n_l} = \sum_{l=1}^{j} \frac{n_l}{n_m} GINI(X_{m_l})$$

Average GINI index for each of the children nodes of m

- Select node with smallest in $GINI_{Split}$

Splitting based on Gain of $GINI$ Index

- Parent node m, with n_m records is to be split into j new child nodes $(m_l)_{1 \le l \le j}$
- Each child node, m_l, has n_l records and probs. $(p_{m_l,1}, p_{m_l,2}, \ldots, p_{m_l,k})$, and GINIs, $GINI(X_{m_l})$
- The gain in $GINI$ is:

$$
\begin{aligned}
Gain(X_m)_{GINI} &= GINI(X_m) - \sum_{l=1}^{j} \frac{n_l}{n_m} GINI(X_{m_l}) \\
&= GINI(X_m) - GINI_{SPLIT}(X_m)
\end{aligned}
$$

- Select node with largest gain in $GINI$ index
- Used in CART, SLIQ, SPRINT

Classification Error

Suppose there are k classes and node m has probabilities $p_m = (p_{m,1}, p_{m,2}, \ldots, p_{m,k})$

$$MC(X_m) = 1 - \max_i p_{m,i}$$

- Maximized with value $1 - \frac{1}{k}$
  when records are equally distributed among all classes
- Minimized, 0, when all records belong to a single class
- Not as smooth as $GINI$ and H

Examples with computing MC

$$MC(X_m) = 1 - \max_i p_{m,i}$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

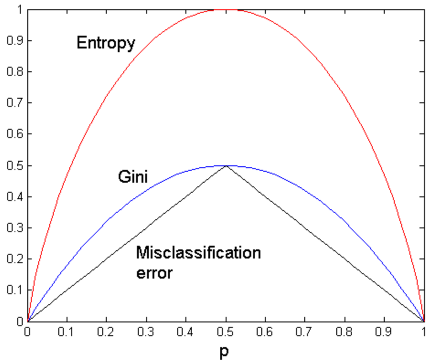
C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Compare Splitting Criterion

For a 2-class problem



Comparing Splitting Criterion

The measures, in general, return good results where:

- Information gain (Entropy):
  - biased towards multi-valued attributes
- Gain ratio:
  - tends to prefer unbalanced splits in which one partition is much smaller than the others
- Gini index:
  - biased to multi-valued attributes
  - has difficulty when # of classes is large
  - tends to favor tests that results in equal-sized partitions and purity in both partitions

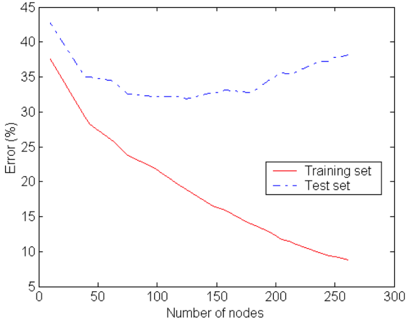
Other Attribute Selection Methods

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Stopping Criterion for Tree Induction

- Stopping Criterion
    - Stop expanding a node when all the records belong to the same class
    - Stop expanding a node when all records have same attribute values
    - Early termination - *to be discussed*
- Tree depth
    - As trees get deeper, or if splits are multi-way the number of data points per leaf node drops very quickly
    - Trees that are too deep tend to overfit the data

Overfitting in Tree Induction

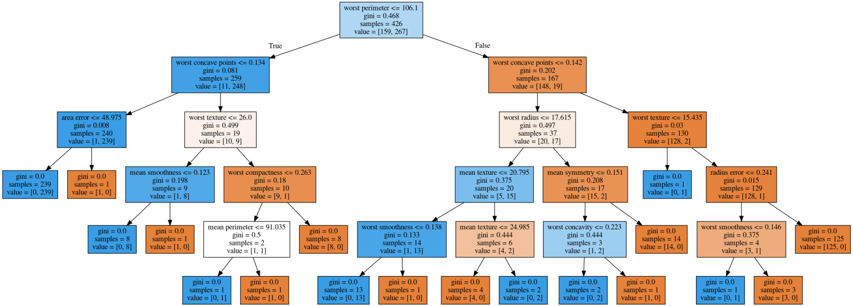


- How to avoid overfitting?
  - stopping rules (pre-pruning)
  - post-pruning

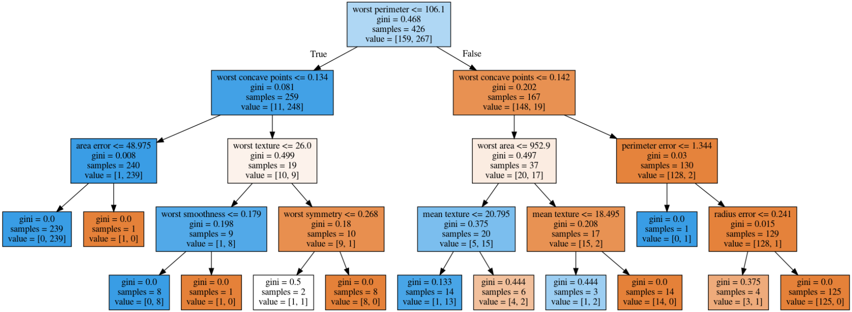
Hyper-parameters for Decision Tree Pruning

- Pre-pruning vs. post-pruning
- Pre-pruning (restrict tree growth):
  - max_depth
  - max_leaf_nodes
  - min_samples_split
  - min_impurity_decrease

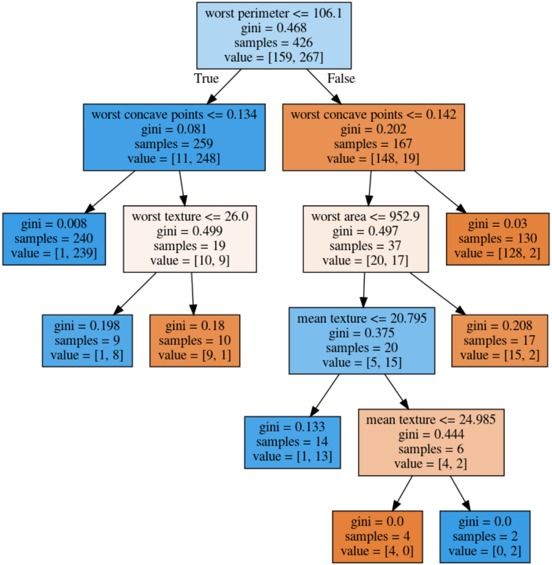
Example: No Pruning



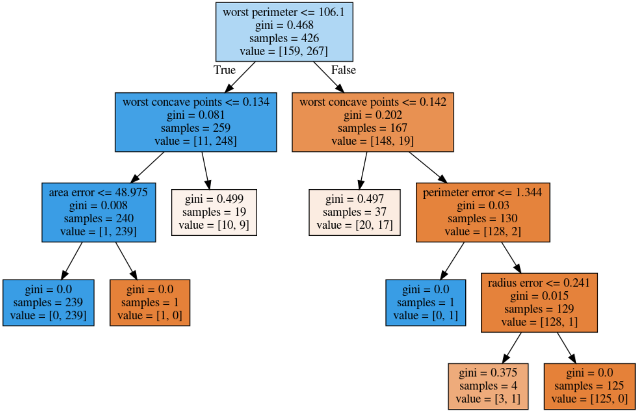
Example: max_depth = 4



Example: max_leaf_nodes = 8



Example: min_samples_split = 50



Cost Complexity Pruning

Cost complexity pruning: add penalty for tree size

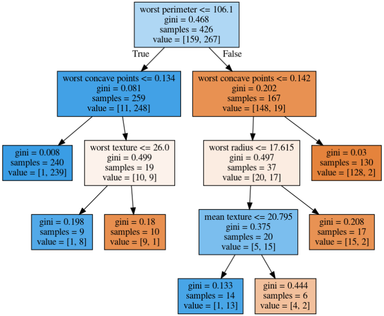
- fully expand tree
- $|T|$ is the number of terminal/leaf nodes
- want to find a subtree that minimizes $Cost(\alpha)$ for a fixed α

$$Cost(\alpha) = \sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

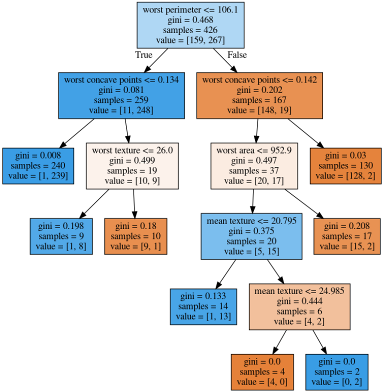
where R_m is the rectangle corresponding to the mth terminal node and $\hat{y}_{R_m}$ is mean of training observations in R_m

Example: Pre- and Post-Pruning

Cost-complexity Pruning



Max Leaf Nodes (Grid Search)



Decision Trees Summary

Decision Trees are one of the most popular and useful data mining method

- Pros:
    - can handle nominal, ordinal, and numeric inputs
    - speed and scalability
    - robustness to outliers and missing values
    - **intrepretability / visualization**
    - compactness of classification rules
    - trees can be used for regression or classification
- Cons
    - several tuning parameters to set
    - decision boundary is non-continuous
    - instability of learning trees