UNIVERSITY OF BERGEN

INFO 284 – Machine Learning

Decision Trees

Bjørnar Tessem

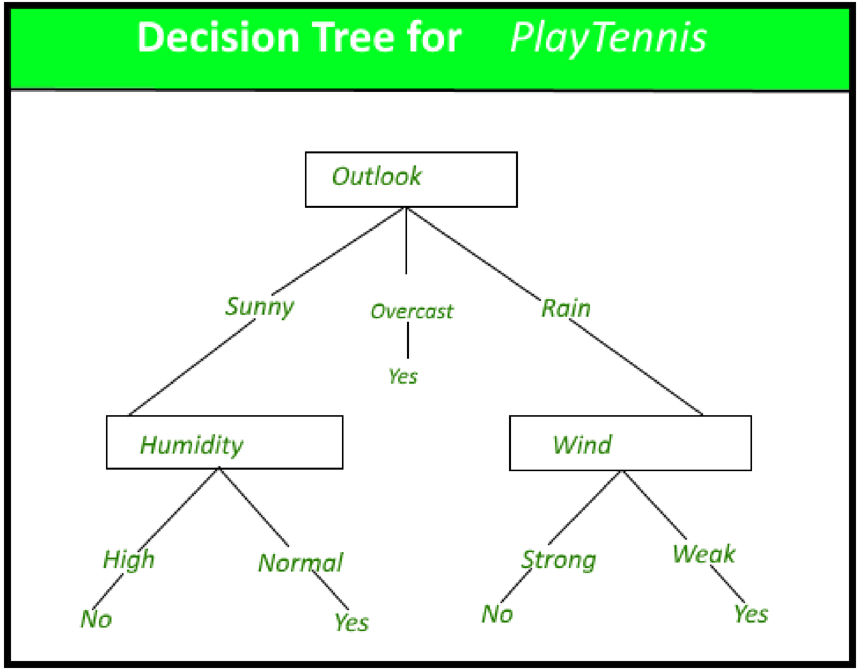
UNIVERSITY OF BERGEN

A Dataset

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Figure 18.3 Examples for the restaurant domain.

A decision tree



A Dataset

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Figure 18.3 Examples for the restaurant domain.

A Decision Tree

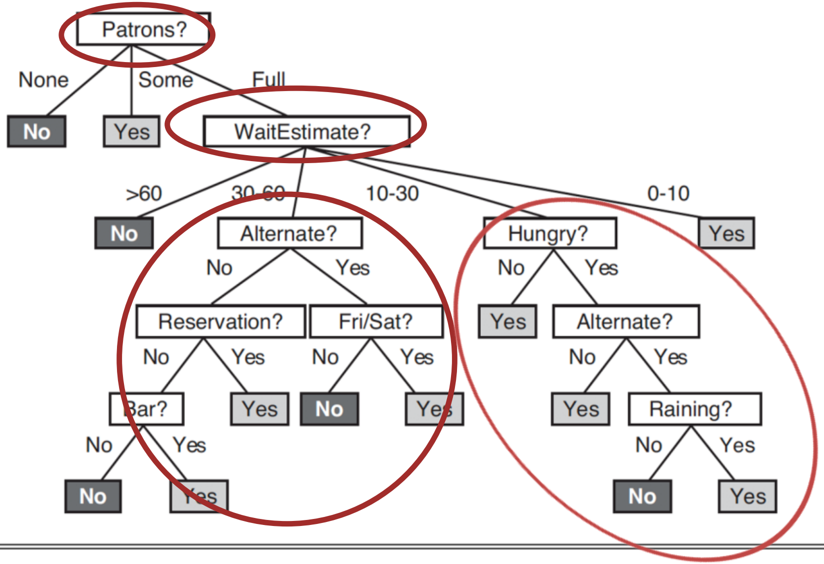


Figure 18.2 A decision tree for deciding whether to wait for a table.

- What is a good tree?
- Each order of features give a different tree
- Build a tree by using feature that provides good information

Decision Tree – An Approach

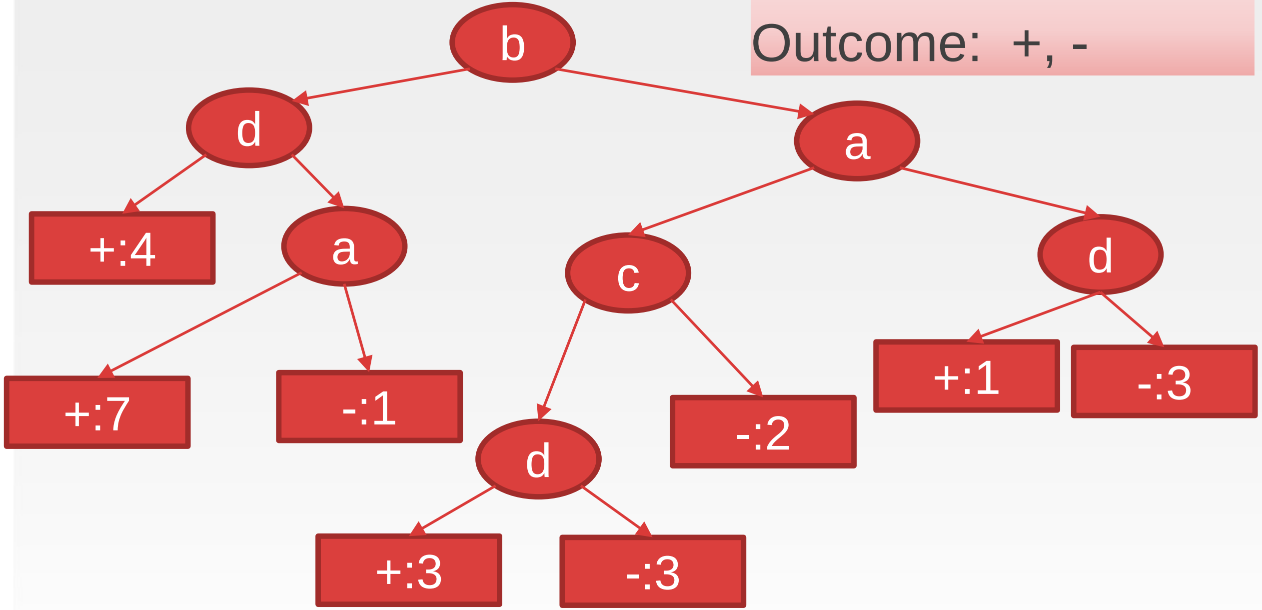
- Assumption: we can measure information gain from deciding on selecting a feature for test
- Steps:
  1. Make an empty decision tree
  2. Select one path in the tree (from root to leaf)
  3. If remaining examples in the current path are in same class we are done (or if empty)
  4. Select the feature V that gives best information gain at this path
  5. Make new leafs in the tree for each of the possible outcomes of V
  6. Repeat from 2 until finished

Example

Features: a, b, c, d

values: L,R

Outcome: +, -



Paths: {}
Paths: {b}
Paths: {b,d},{b,a}
Paths: {b,d,a},{b,a}
Paths: {b,a}
Paths: {b,a,c},{b,a,d}
Paths: {b,a,c,d},{b,a,d}
Paths: {b,a,d}
Paths:

Information Gain

- Gini impurity
  - How often a randomly chosen element of a set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset

$$I_G(p) = \boxed{\sum_{i=i}^{J} \boxed{p_i} \boxed{\sum_{k\neq i} p_k}} = \boxed{\sum_{i=1}^{J} p_i(1-p_i)}$$

- Entropy
  - Uncertainty in a variable: acquisition of information lead to less entropy

$$H(V) = \sum_i p_i \log_2 \frac{1}{p_i} = -\sum_i p_i \log_2 p_i$$

Classification process

- New datapoint → follow tree to root
- No examples left in leaf → use the one with maximal count for node above
- If disagreeing examples in leaf → error/noise in data → choose most common value

Continuous values

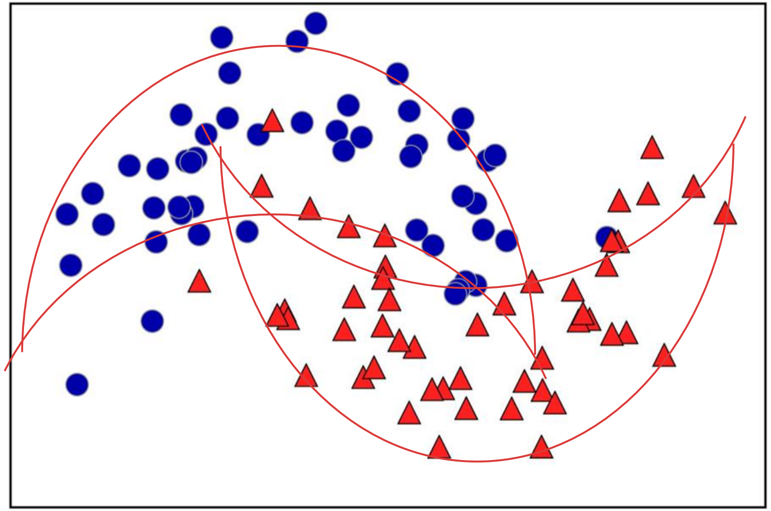
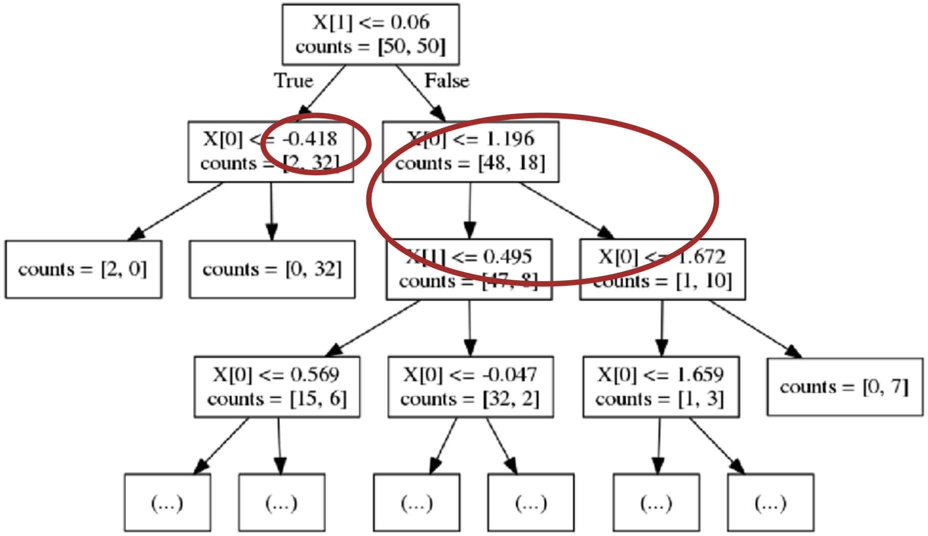
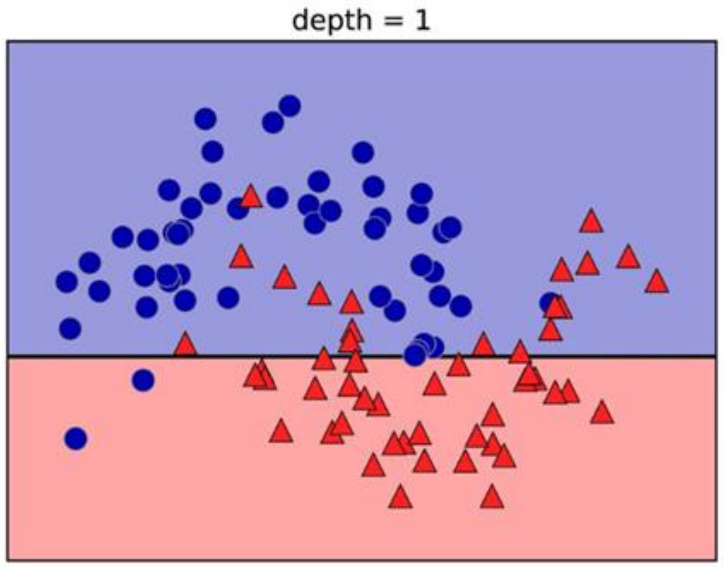


Figure 2-23. Two-moons dataset on which the decision tree will be built



Continuous features - approach

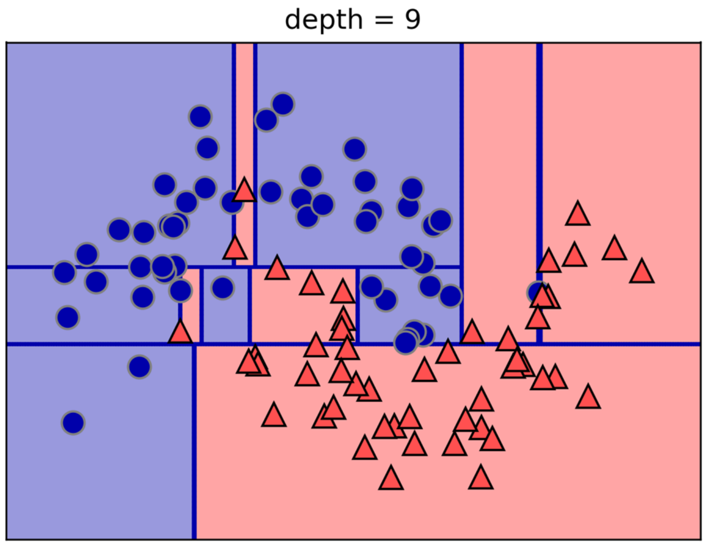


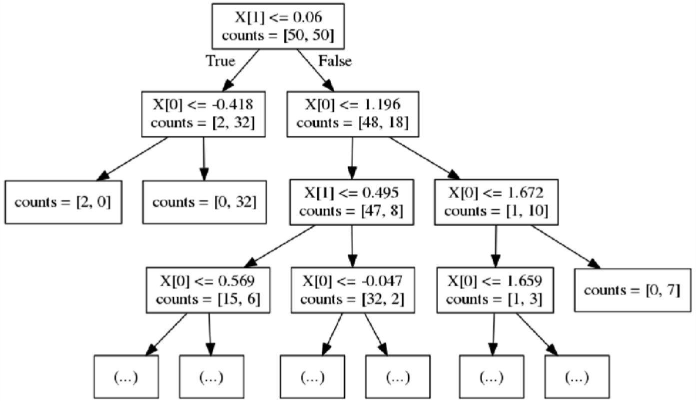
Decision boundaries parallel to x-axis or y-axis

Split – find best decision boundary - by:

Find optimal boundary for each variable (according to information gain)
Select variable (and boundary) with the most information gain

Complexity of decision trees





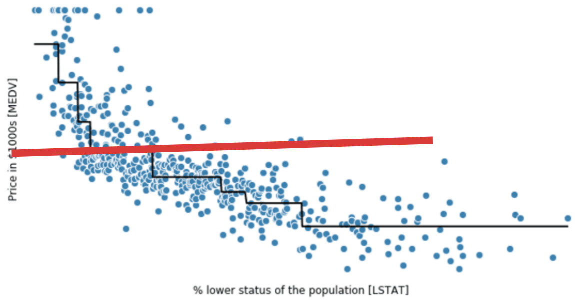
Too large trees may lead to overfitting – and is computationally hard to compute

Controlling complexity - strategies

- Pre-pruning: Stop the creation of the tree when
  - Depth grow above limit
  - Number of leafs grow above limit
  - Number of datapoints in node falls below limit
- Post-pruning:
  - Removing or collapsing nodes that contain little information
- Voting in classification

Regression in decision trees

- When checking a path during tree constuction:
  - For each feature find decision boundary
    - split current dataset so that **variance** in each group is lowest
  - Select variable with lowest variance sum
  - Create new test node with decision boundary



Stop splitting if target variance in remaining data is below chosen threshold

Advantages and disadvantages

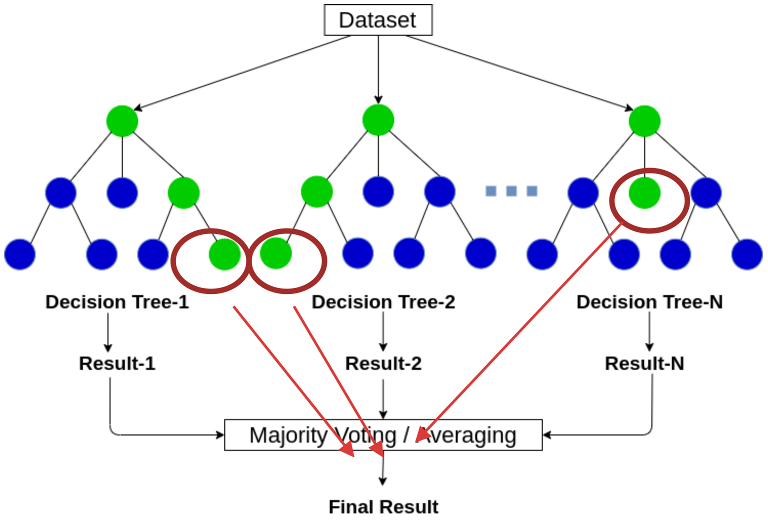
- The resulting model can be easily visualised and understood
  - Tree structure is intuitive and compact
  - Feature importance measure how significant a feature is in decision making
- The algorithms are invariant to scaling of the data
  - Normalisation of features is not needed
- Danger of overfitting
- Decision trees are not good with sparse data
  - Many features and many 0-s (text data)

Ensembles of decision trees

- Ensembles of methods:
  - Combine multiple machine learning models
  - Use average or voting to decide on value
- Effects:
  - Better generalization even if each model overfit
  - Using subsets of data reduces computational complexity for each model

Random forests

- Generate many decision trees
- Use different extracts of data for building each tree
  - Subset of datapoints
  - Subset of features
  - Both
- Each tree overfits
- Averaging or voting reduces overfitting
  - Soft voting if leaf nodes are not «pure»



Gradient boosting

- Ensemble method
- Approach
  - Build decision tree (generation 0)
  - For each data point compute error in classification
    - Use this error as an alternative target
  - Build decision tree with this new target (generation 1)
    - Use «error in error» as new «error» target
  - Build decision tree for new target (generation 2)
  - …

Gradient boosting

- Pre-pruning to make small trees
- Some parameters
  - Learning rate indicate how much of value or observed error is left to next generation
  - Specify number of trees
- Prediction:
  - Run data point through all decision trees in sequence
    - Add up predictions
      - Value + «error» + «error in error» + …
  - New generation trees slowly correct for errors in earlier level prediction

Properties of decision tree ensembles

- Random trees
  - Popular method
  - Controls overfitting
- Gradient boosting
  - Like random forests, but need fewer trees
- Both
  - Careful tuning of parameters is needed
  - Tree building is computationally complex

uib.no