Decision Trees

Dr. Ratna Babu Chinnam

Industrial & Systems Engineering

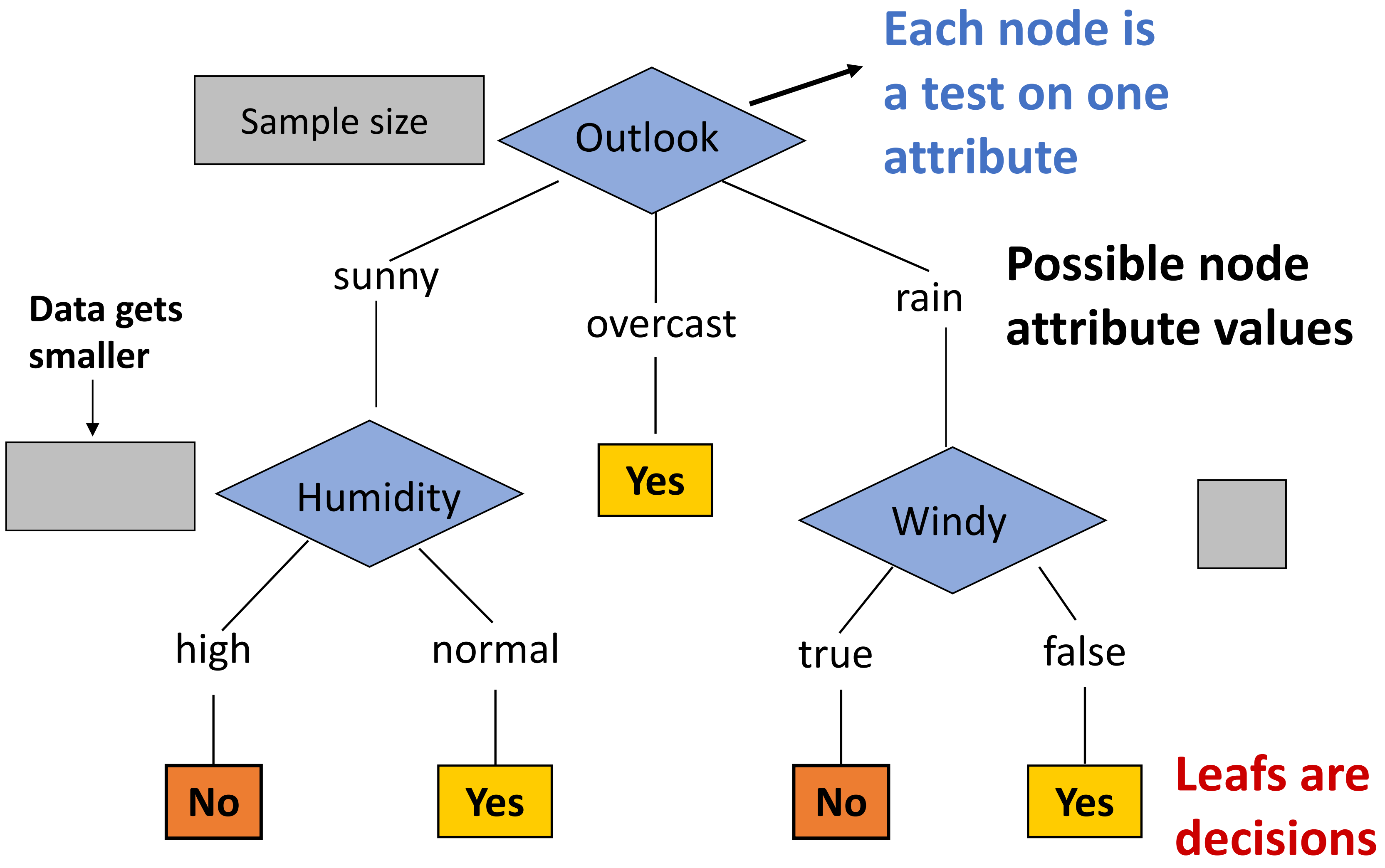
Wayne State University

Decision Trees: Overview

- Non-linear classifier
- Easy to use (simple math)
- Easy to interpret and explain
  - Can be reduced to decision “rules”
- Susceptible to overfitting but can be avoided
- Can be extended to handle continuous attributes
- Can be extended to deal with missing data
- Quite robust and popular in practice!
- Can be extended to handle regression: Regression Trees

Anatomy of a Decision Tree

Example: To 'play tennis' or not?



Decision Tree Rules:

(Outlook ==overcast) -> yes
(Outlook==rain) & (Windy==false) ->yes
(Outlook==sunny) & (Humidity=normal) ->yes

A New Test Example:

(Outlook==rain) & (Windy==false)

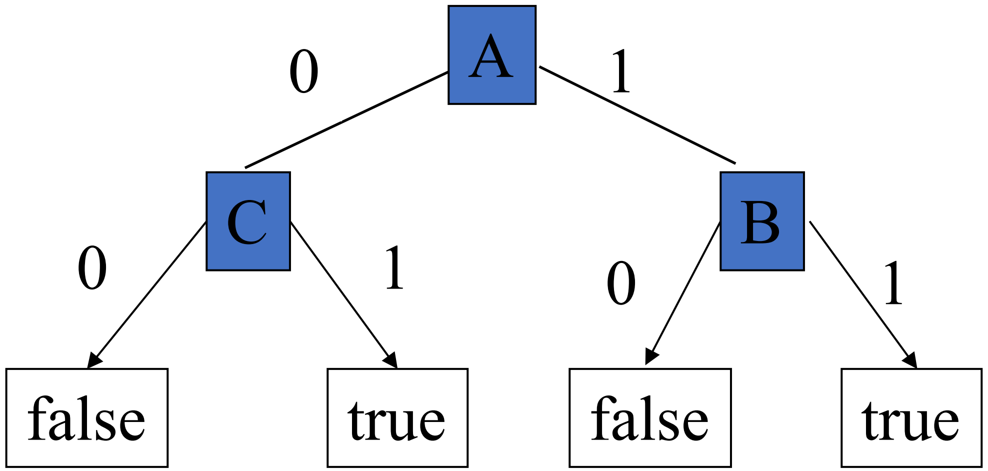
Pass it on tree:
-> Decision is yes.

Decision Trees

- **Definition**: Decision trees represent a “disjunction” of “conjunctions” of constraints on attribute values of instances

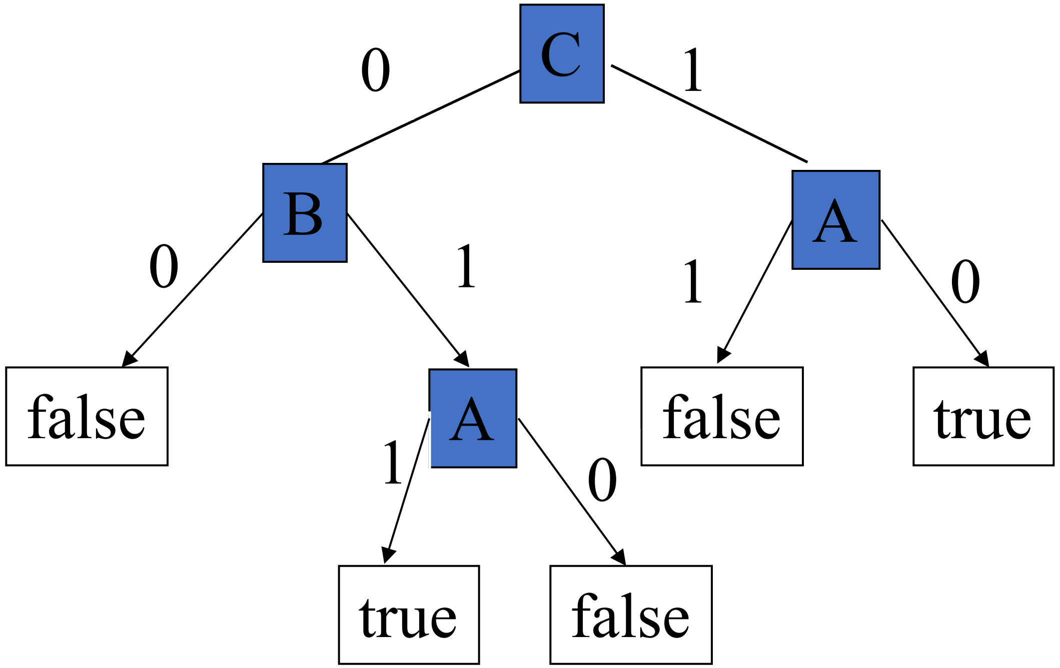
```
(Outlook==overcast)
OR
((Outlook==rain) AND (Windy==false))
OR
((Outlook==sunny) AND (Humidity==normal))
=> yes play tennis
```

Decision Tree Representation



Y=((A and B) or ((not A) and C))

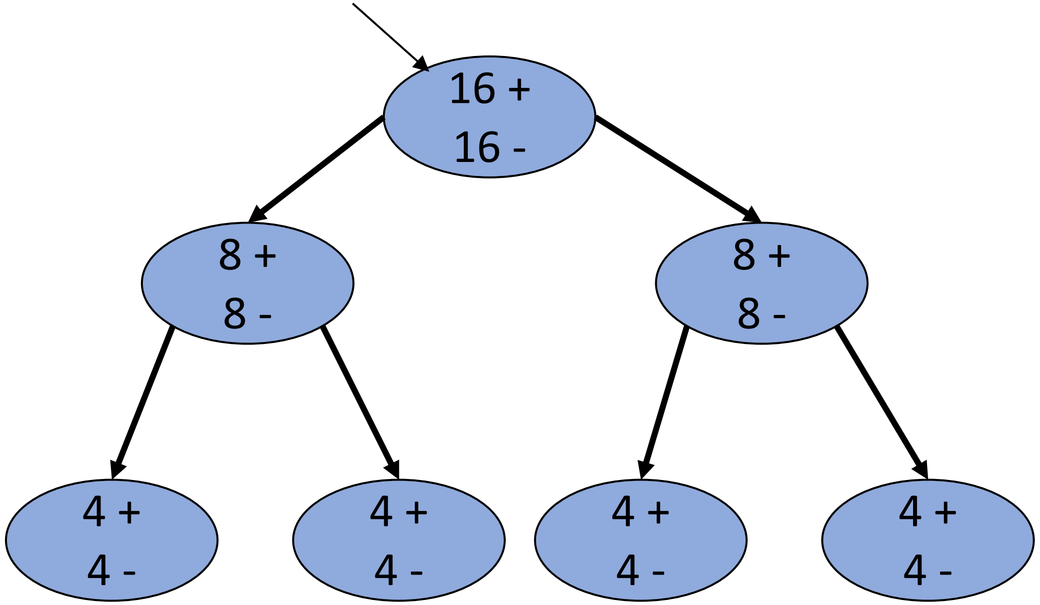
Same Concept Different Representation



What do you prefer? Why?

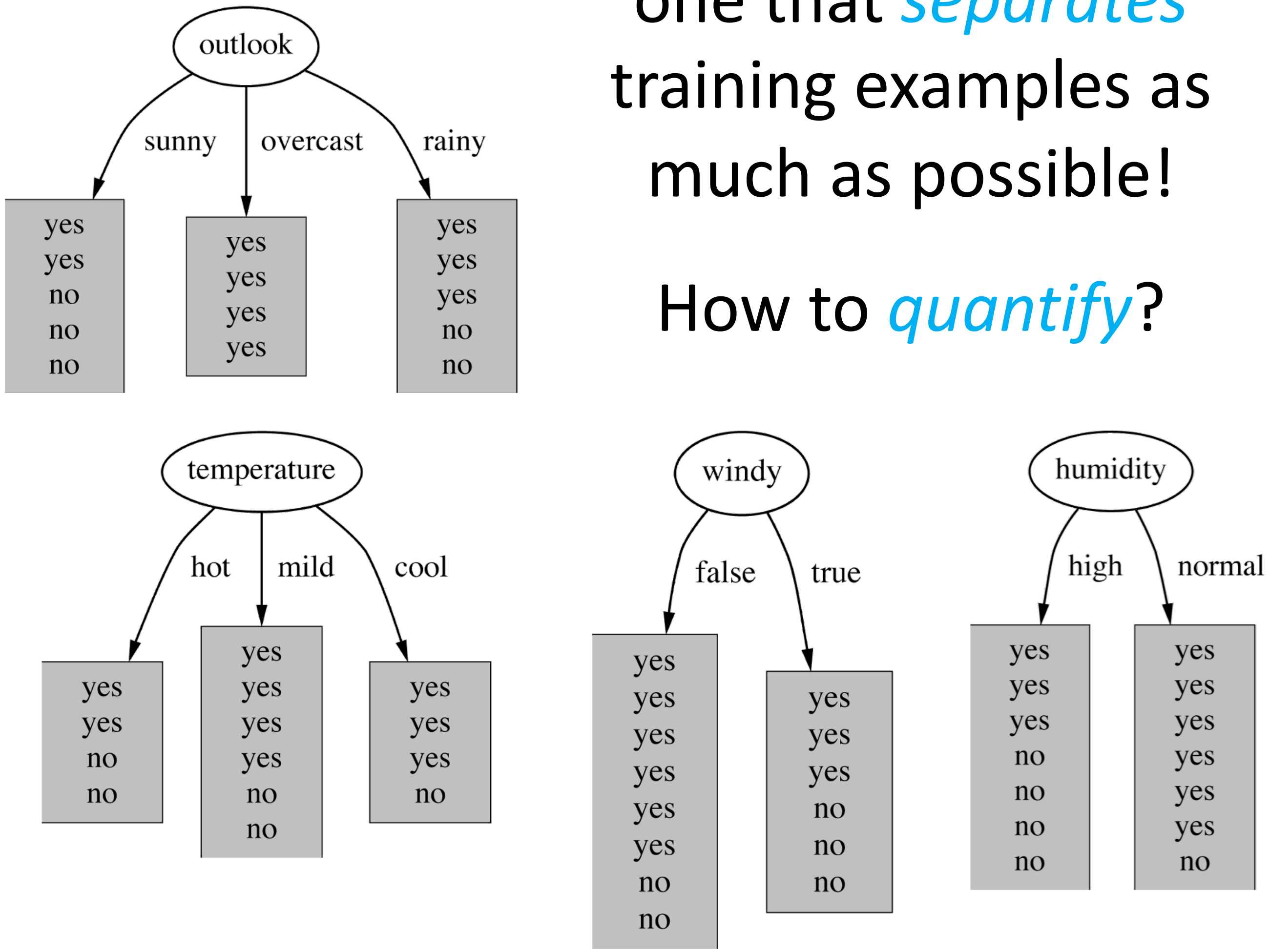
Which Attribute To Select For Splitting?

Distribution of each class (not attribute)



This is bad splitting…

Example: To 'play tennis' or not?



Intuitively, we prefer one that *separates* training examples as much as possible!

How to *quantify*?

“Information”

- Imagine:
  - Someone is about to tell you your own name
  - You are about to observe the outcome of a coin flip
  - You are about to observe the outcome of a biased coin flip
- Each situation has a different ***amount of uncertainty***
- **Definition**: Information is reduction in uncertainty (amount of surprise in outcome)

$$I(E) = -\log_2 p(x)$$

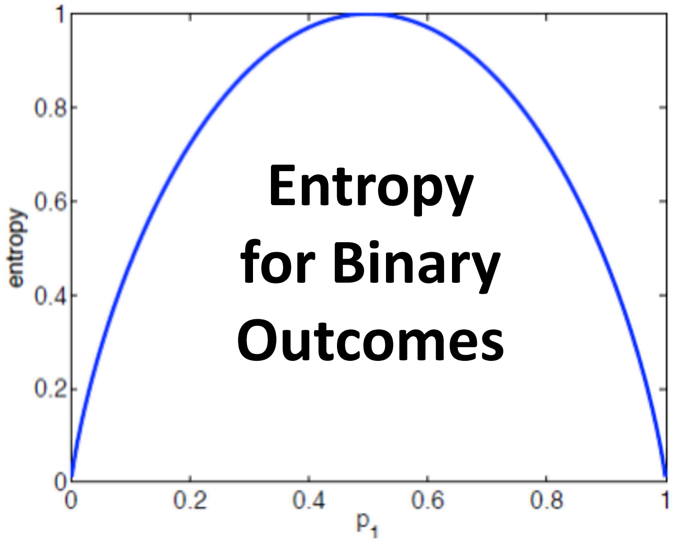
- If probability of event happening is small and it happens, information (revealed) is large:
  - Observing outcome of a coin flip is head: $I = -\log_2 1/2 = 1$
  - Observe outcome of a dice is 6: $I = -\log_2 1/6 = 2.585$

Entropy (measures "Purity")

- **Definition**: Expected amount of information when observing output of a random variable X

$$H(X) = E(I(X)) = \sum_i p(x_i)I(x_i) = -\sum_i p(x_i)\log_2 p(x_i)$$

- Balanced coin: $H(X) = -\sum_i \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$
- Balanced dice: $H(X) = -\sum_i \frac{1}{6}\log_2\left(\frac{1}{6}\right) = 2.585$
- If there are k possible outcomes: $H(X) \leq \log_2 k$
  - Equality holds when all outcomes are equally likely
  - The more the probability distribution deviates from "Uniform", the lower the entropy (Uniform distribution has highest entropy)



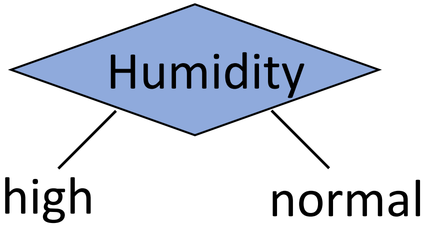
Information Gain

- **Definition**: Information gain measures change in entropy from before and after split (conditioning)
  - Can be used as a criteria to decide on the attribute to split
- **Conditional Entropy:**

$$H(X|Y) = -\sum_j p(y_j) H(X|Y = y_j) = \sum_j p(y_j) \sum_i p(x_i|y_j) \log_2 p(x_i|y_j)$$

Example: To 'play tennis' or not?

Full Set, S: [9+, 5-]
Entropy: 0.940

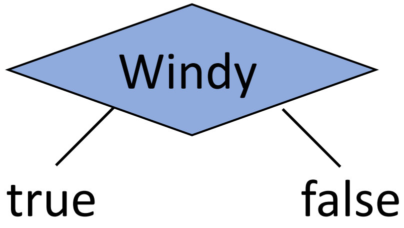


Set: [3+, 4-]
Entropy: 0.985

Set: [6+, 1-]
Entropy: 0.592

$$Gain(S|Humidity) = .940 - \left(\frac{7}{14}\right).985 - \left(\frac{7}{14}\right).592 = 0.151$$

Full Set, S: [9+, 5-]
Entropy: 0.940



Set: [6+, 2-]
Entropy: 0. 811

Set: [3+, 3-]
Entropy: 1.0

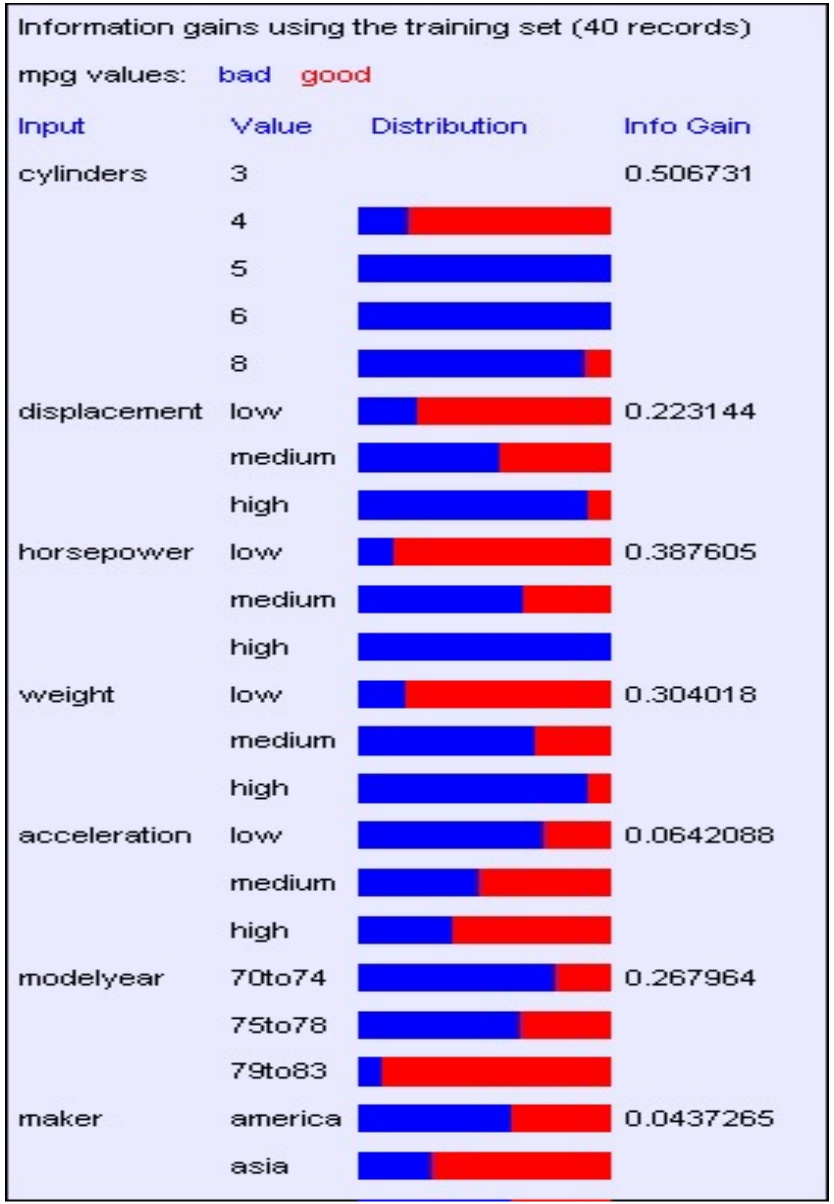
$$Gain(S|Wind) = .940 - \left(\frac{8}{14}\right).811 - \left(\frac{6}{14}\right)1.0 = 0.048$$

Humidity* provides greater information gain than *Wind, relative to target classification!

Decision Tree Example: Classifying Fuel Efficiency

- From UCI Repository (R. Quinlan)
  - 40 Records

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe



Decision Tree: Variants

- Most **decision tree learning algorithms are variations on a core algorithm** that employs a ***top-down greedy search***
  - ***ID3 algorithm*** (Quinlan, 1986) & successor ***C4.5*** (Quinlan, 1993)
- **ID3 Algorithm:** Begins with the question "**which attribute should be tested at root of the tree**?"
  - Evaluates each attribute using a statistical test to determine how well it alone classifies training examples
  - Best attribute is selected as test at root node of the tree
  - Descendant of root node is created for each possible value of attribute
  - Training examples are sorted to the appropriate descendant node
  - Entire process is repeated using training examples associated with each descendant node
  - Forms a *greedy search* for an acceptable decision tree and algorithm never backtracks to reconsider earlier choices

Decision Tree: Variants …

- **C4.5, Quinlan's next iteration**: New features (versus ID3) are:
  - Accepts both continuous and discrete features;
  - Handles incomplete data points;
  - Solves over-fitting problem by (very clever) bottom-up technique usually known as "pruning"; and
  - Different weights can be applied the features that comprise the training data.
- **C5.0 Algorithm**: Most significant feature unique to C5.0 is a scheme for deriving rule sets.
  - After a tree is grown, the splitting rules that define the terminal nodes can sometimes be simplified: that is, one or more condition can be dropped without changing the subset of observations that fall in the node.
- **CART or Classification And Regression Trees:**
  - Often used as a generic acronym for the term Decision Tree, though it apparently has a more specific meaning. In sum, the CART implementation is very similar to C4.5.

Decision Trees: Caveats/Concerns

INFORMATION GAIN

- Number of possible values influences information gain
- The more possible values, the higher the gain
  - The more likely it is to form small, but pure partitions

Other Purity Measures:

- Gini Index: $1 - \sum_j p(j)^2$
  - Smaller the better
- Chi-square Test

OVERFITTING

- You can perfectly fit to any training data (with enough attributes)
- Zero bias, high variance

Solution:

- Stop growing tree when further splitting is not yielding big improvement
- Grow a full tree, then prune the tree, eliminating weak nodes

Optimal Decision Trees

- Prevalent methods (e.g., `ctree` and `rpart` available in R) utilize a single variable in each split
  - Limits expressiveness and, in some cases, model accuracy
- New research is seeking to develop “optimal” decision trees
  - Exploit optimization-based framework to exploit multivariate splits and enable more expressive, comprehensible and accurate tree models.
- Sample “R” Package: bsnsing | Link
  - Author: Dr. Yanchao Liu, Wayne State University
  - A mixed integer program (MIP) is solved at each node to identify optimal combination of features used to split node
  - Supported MIP solvers include CPLEX (commercial) and lpSolve (free)
  - Created Boolean attributes can compete with other discrete-valued candidate attributes available for growing decision tree.

Decision Trees: Matlab | Link

WAYNE STATE
UNIVERSITY

Classification Trees

Binary decision trees for multiclass learning

To interactively grow a classification tree, use the **Classification Learner** app. For greater flexibility, grow a classification tree using `fitctree` at the command line. After growing a classification tree, predict labels by passing the tree and new predictor data to `predict`.

Apps

Classification Learner	Train models to classify data using supervised machine learning

Functions

expand all

› Create Classification Tree

› Improve Classification Tree

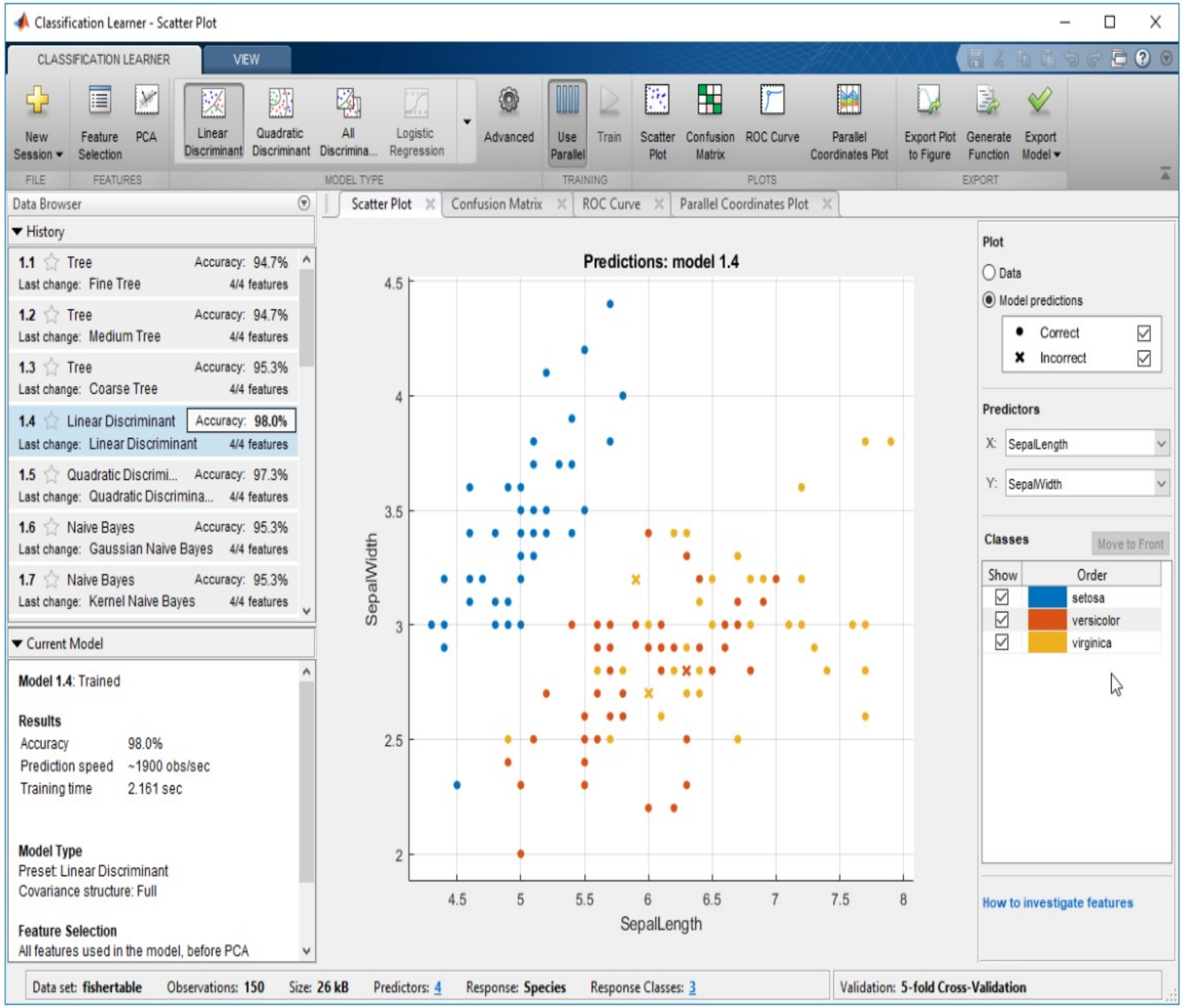
› Cross-Validate

› Measure Performance

› Classify Observations

Classes

`ClassificationTree`	Binary decision tree for classification
`CompactClassificationTree`	Compact classification tree
`ClassificationPartitionedModel`	Cross-validated classification model



Decision Trees: Python (sklearn) | [Link]

scikit learn

Prev Up Next

scikit-learn 1.0.2

Other versions

Please **cite us** if you use the software.