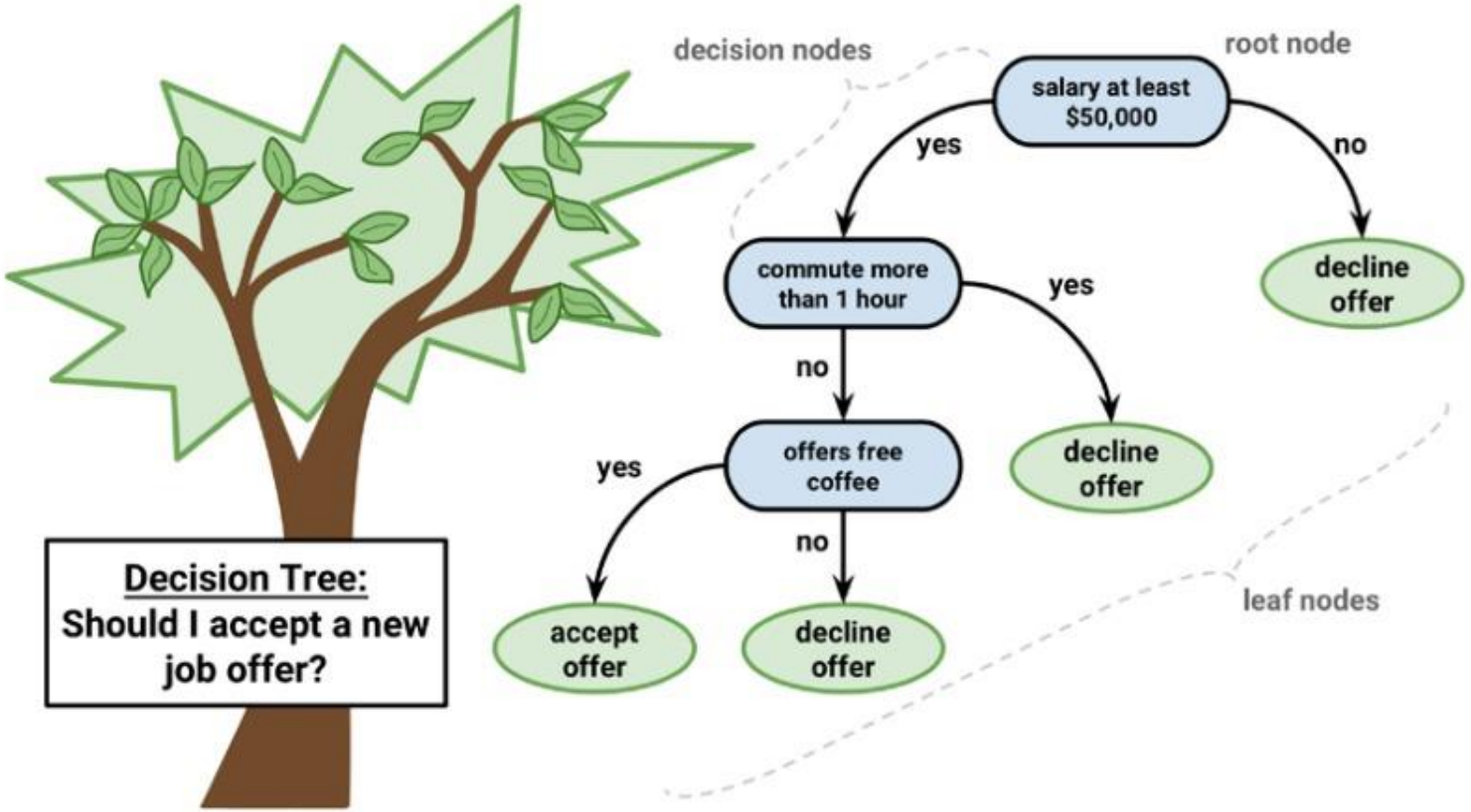
Classification Using Decision Trees and Rules

Chapter 5

Introduction

- Decision tree learners use a tree structure to model the relationships among the features and the potential outcomes.
  - a structure of branching decisions into a final predicted class value
- Decision begins at the root node, then passed through decision nodes that require choices.
- Choices split the data across branches that indicate potential outcomes of a decision
- Tree is terminated by leaf nodes that denote the action to be taken as the result of the series of decisions.

Decision Tree Example



Benefits

- Flowchart-like tree structure is not necessarily exclusively for the learner's internal use.
  - Resulting structure in a human-readable format.
  - Provides insight into how and why the model works or doesn't work well for a particular task.
    - Useful where classification mechanism needs to be transparent for legal reasons, or in case the results need to be shared with others in order to inform future business practices
      - Credit scoring models where criteria that causes an applicant to be rejected need to be clearly documented and free from bias
      - Marketing studies of customer behavior such as satisfaction or churn, which will be shared with management or advertising agencies
      - Diagnosis of medical conditions based on laboratory measurements, symptoms, or the rate of disease progression

Applicability

- Widely used machine learning technique
- Can be applied to model almost any type of data with excellent results
- Does not fit task where the data has a large number of nominal features with many levels or it has a large number of numeric features.
  - Result in large number of decisions and an overly complex tree.
  - Tendency of decision trees to overfit data, though this can be overcome by adjusting some simple parameters

Divide and Conquer

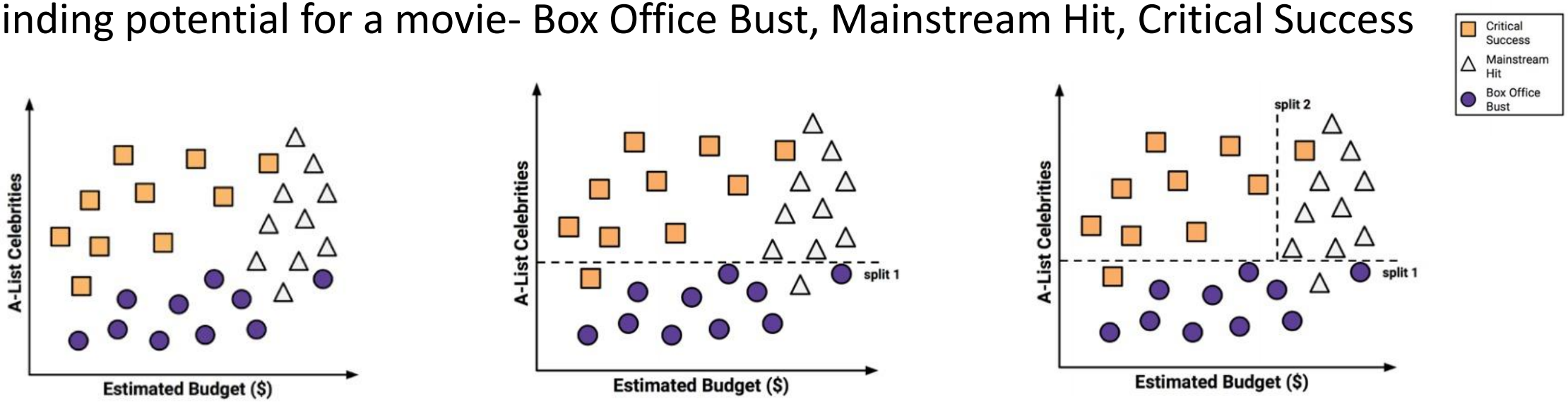
- Decision trees are built using a heuristic called recursive partitioning.
  - Divide and conquer because it splits the data into subsets, which are then split repeatedly into even smaller subsets,
  - Stops when the data within the subsets are sufficiently homogenous, or another stopping criterion has been met.
- Root node represents the entire dataset
  - Algorithm must choose a feature to split upon
    - Choose the feature most predictive of the target class.
  - Algorithm continues to divide and conquer the data, choosing the best candidate feature each time to create another decision node, until a stopping criterion is reached.

Divide and Conquer

- Stopping Conditions
  - All (or nearly all) of the examples at the node have the same class
  - There are no remaining features to distinguish among the examples
  - The tree has grown to a predefined size limit

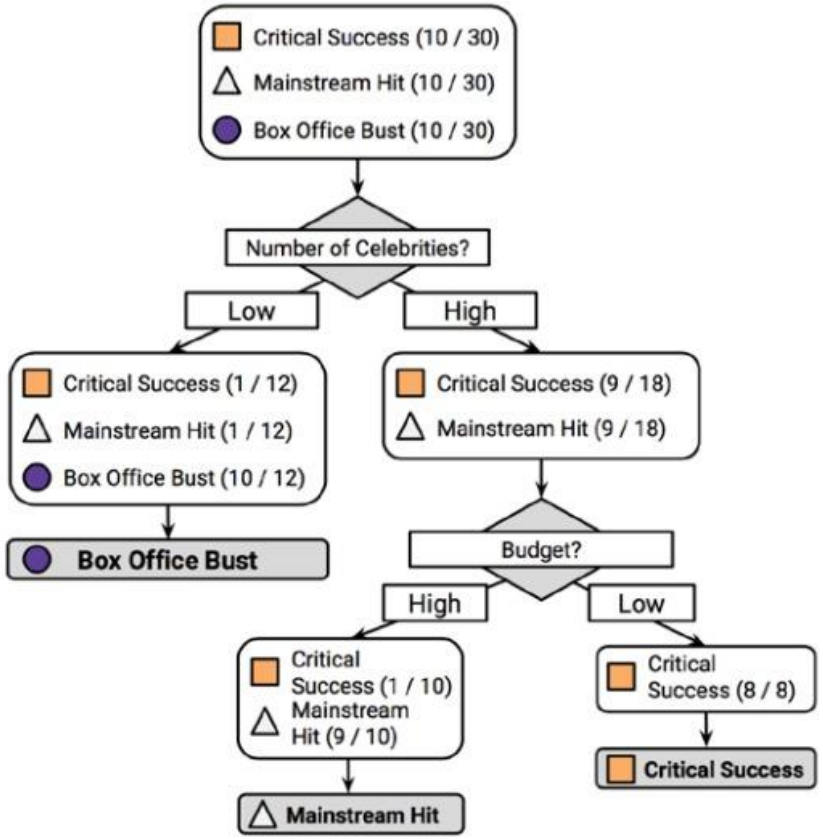
Example

- Finding potential for a movie- Box Office Bust, Mainstream Hit, Critical Success



- Diagonal lines might have split the data even more cleanly.
  - Limitation of the decision tree's knowledge representation, which uses axis-parallel splits.
  - Each split considers one feature at a time prevents the decision tree from forming more complex decision boundaries

Decision Tree



Predict: Will John Play Tennis?

Training examples: **9 yes / 5 no**

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Predict on	D15	Rain	High	Weak	?	Vs. **k-NN: Lookup**

- Difficult to Interpret
- Understanding of when John plays
- Divide and conquer
  - Split into subsets
  - Is it pure?
    - Yes: Stop
    - No: Repeat
  - Understand which subset the data falls into

Credit Example From V. Lavrenko

Predict: Will John Play Tennis?

9 yes / 5 no

Outlook

Pure Subset

Overcast

Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

Sunny

Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

Split Further
2 yes/3 No

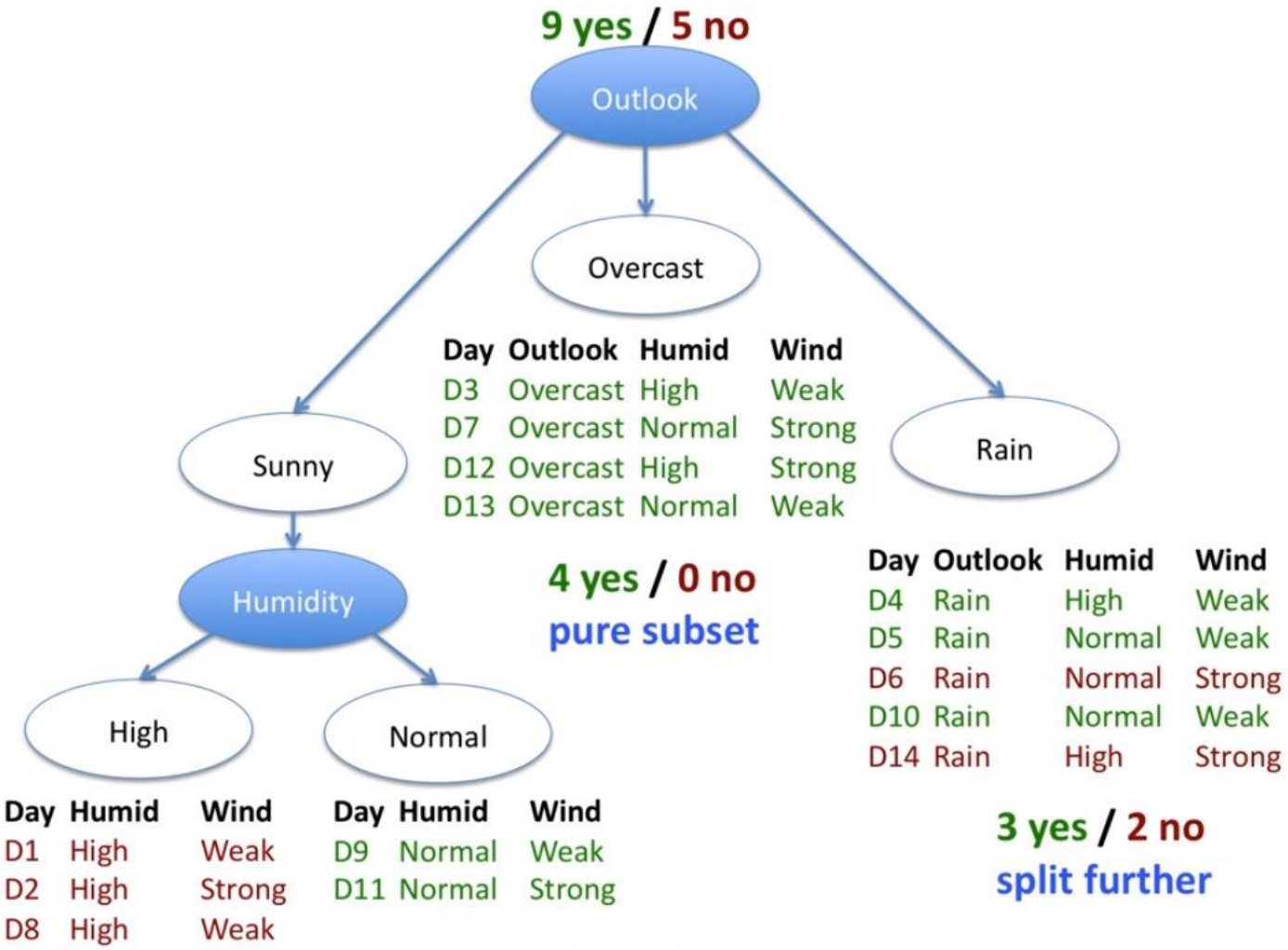
Pure Subset
4 yes/0 No

Rain

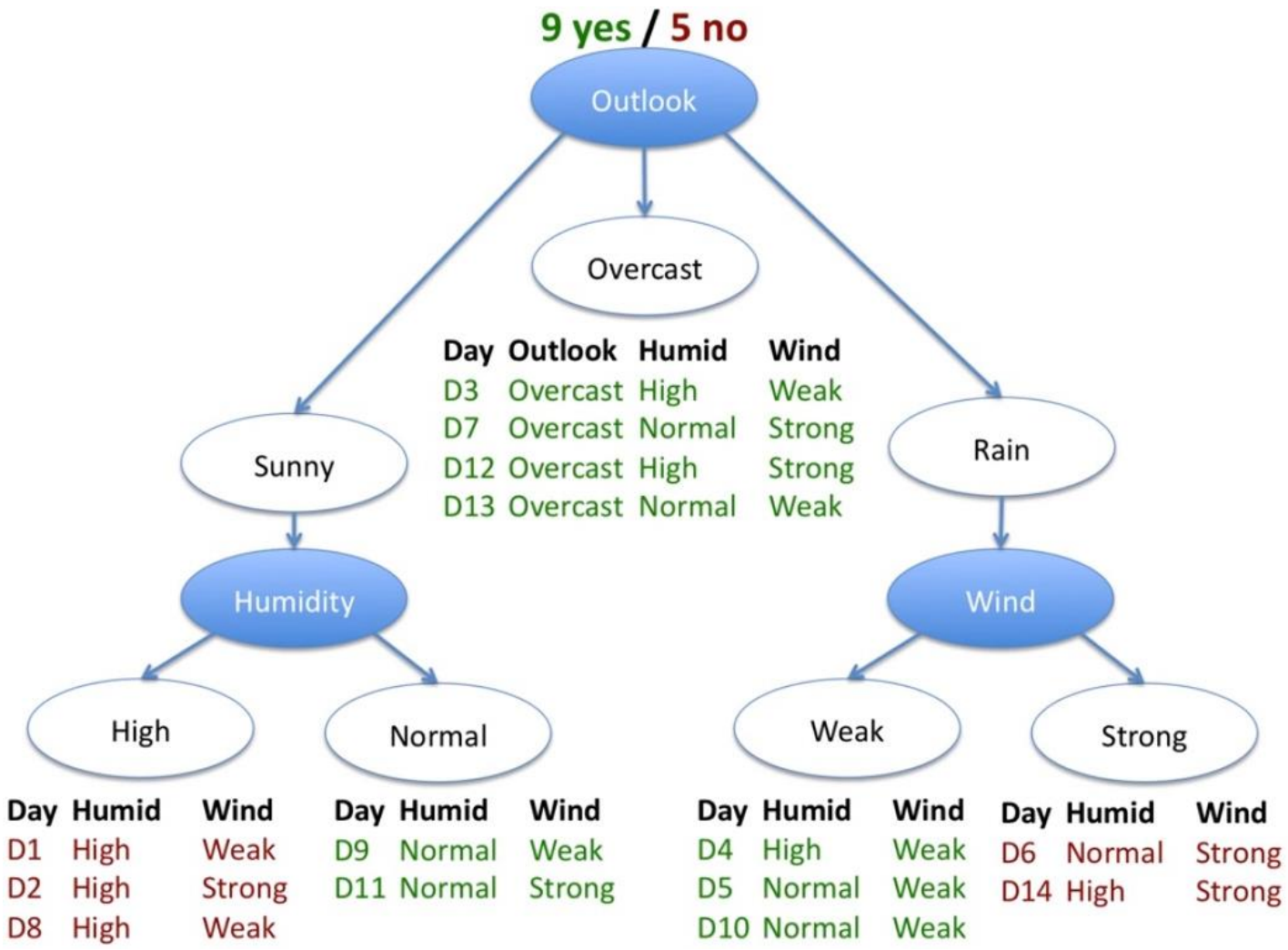
Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

Split Further
3 yes/2 No

Will John Play Tennis?

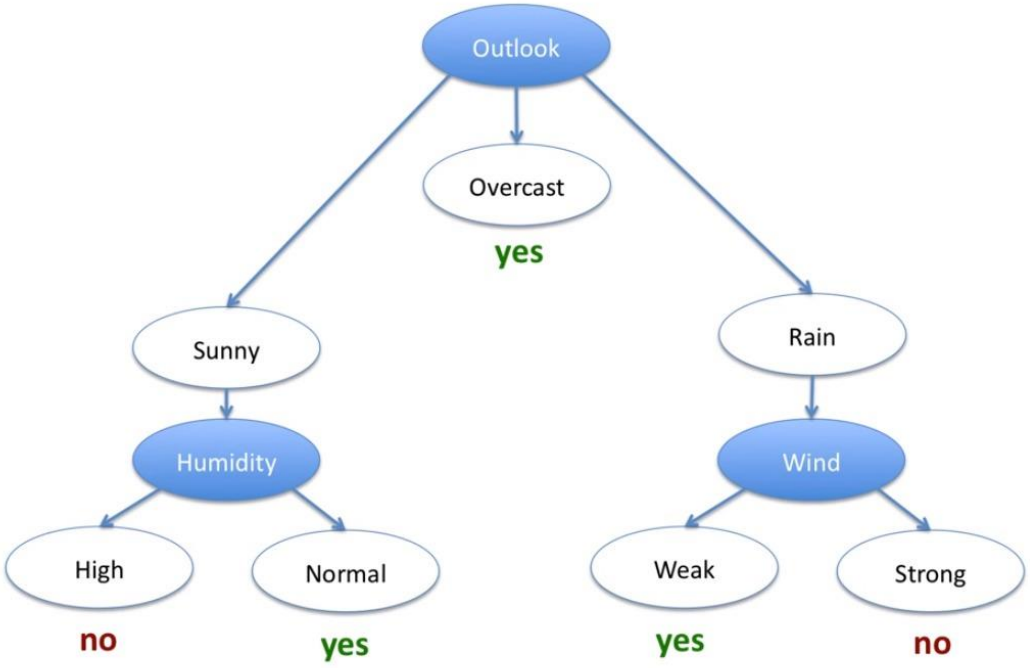


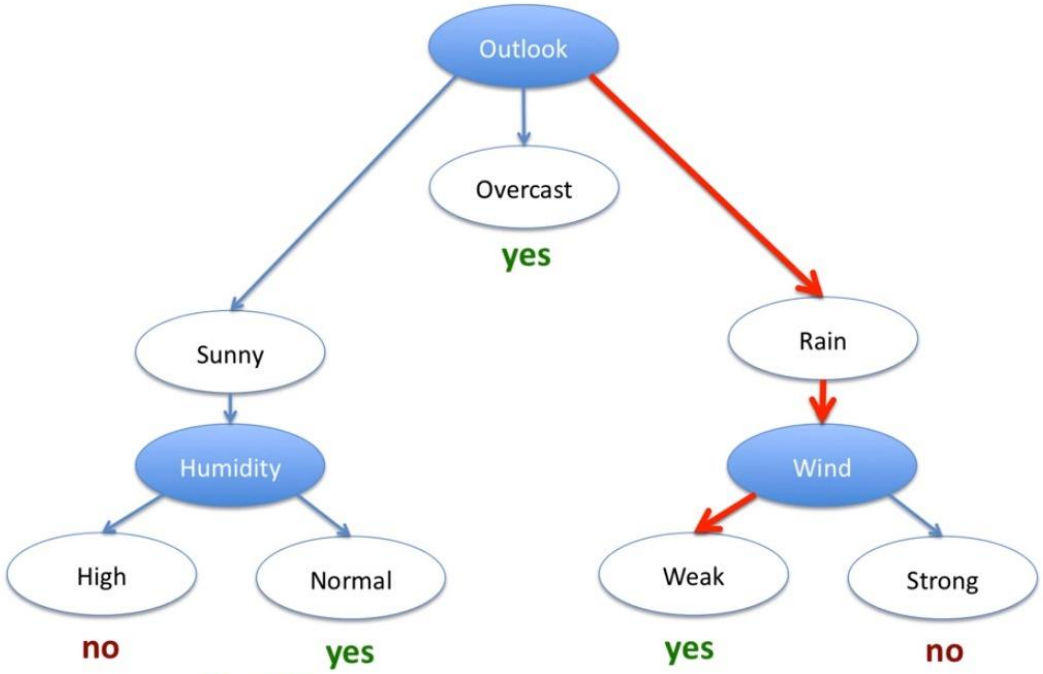
Will John Play Tennis?



Will John Play Tennis?

- Output of the Decision Tree
  - Decision Labels at the Leaf
  - Balanced Tree not required





New data:

Day	Outlook	Humid	Wind	
D15	Rain	High	Weak	→ Yes

ID3

- Split (node, {examples}):
  1. A ← the best attribute for splitting the {examples}
  2. Decision attribute for this node ← A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. For each child node / subset:

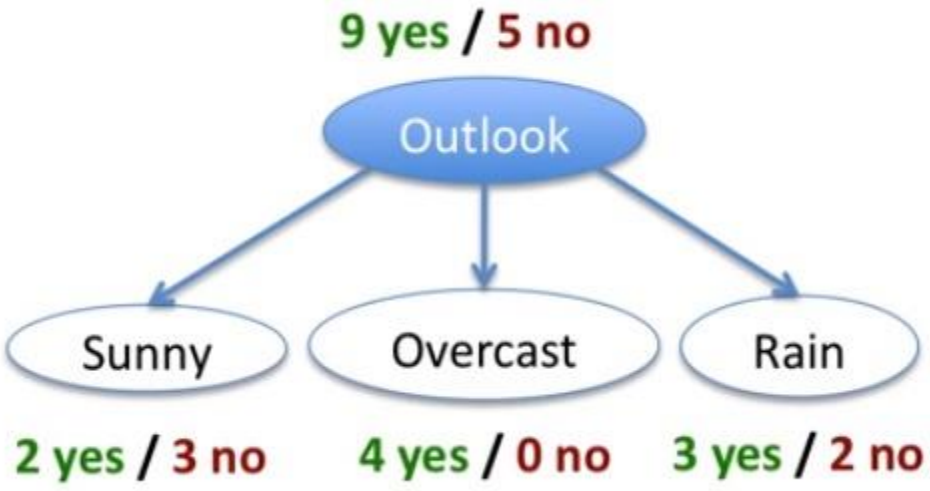
     if subset is pure: STOP

     else: Split (child_node, {subset})

- Ross Quinlan (ID3, C4.5)
- Breimanetal (CaRT)

Choosing the Best Split

- Choice of Attribute critically determines quality of the Decision Tree



- Outlook : Pure Subset
- Wind: Weak better than Strong
  - Strong completely uncertain
- Need subset strongly biased towards positive or negative.
  - Certainty over Uncertainty: 4/0 as pure as 0/4

Entropy

- Quantifies the randomness, or disorder, within a set of class values.
  - High entropy sets are diverse
  - Find splits that reduce entropy, increasing homogeneity within the groups.
- If there are two classes, entropy values can range from 0 to 1.
- For n classes, entropy ranges from 0 to log2 (n)
  - Minimum value indicates that the set is completely homogenous
  - Maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

Entropy

Entropy: $H(S) = - p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$ bits

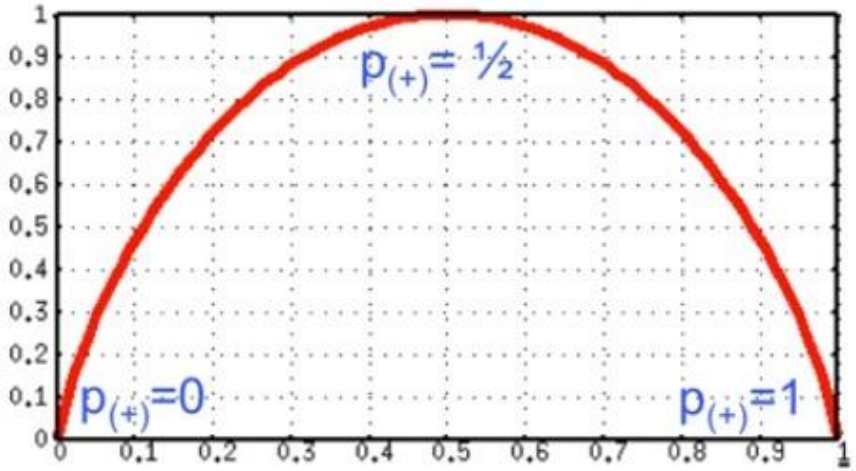
– S … subset of training examples

– $p_{(+)}$ / $p_{(-)}$ … % of positive / negative examples in S

- Interpretation: If X is in S, how many bits are used to represent whether X is positive or negative?
  - Impure (3 pos/ 3 neg)

    H(S) = -3/6 log(3/6) – 3/6 log(3/6) = 1/2+1/2 = 1
  - Pure (4 pos/0 neg)

    H(S) = -4/4 log (4/4) – 0/4 log(0/4) = 0



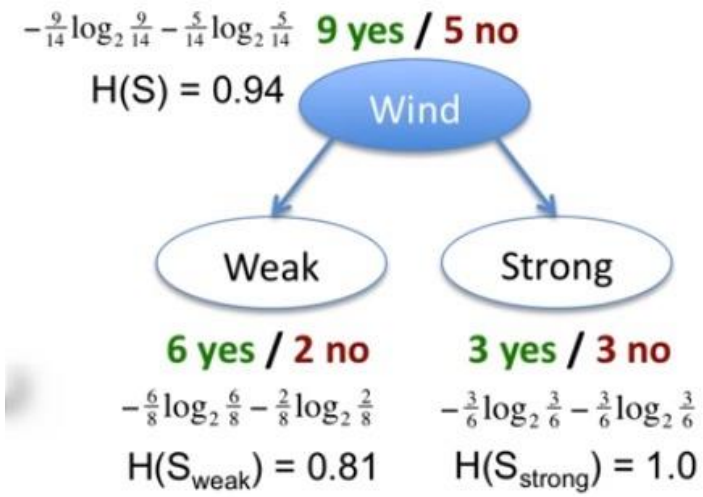
Information Gain

- Objective of Splits is to produce pure sets
  - Information Gain arises out of a split (Reduction in Entropy). Gain quantifies the Certainty obtained after a split

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$

  - Information Gain performed recursively



Gain (S, Wind)

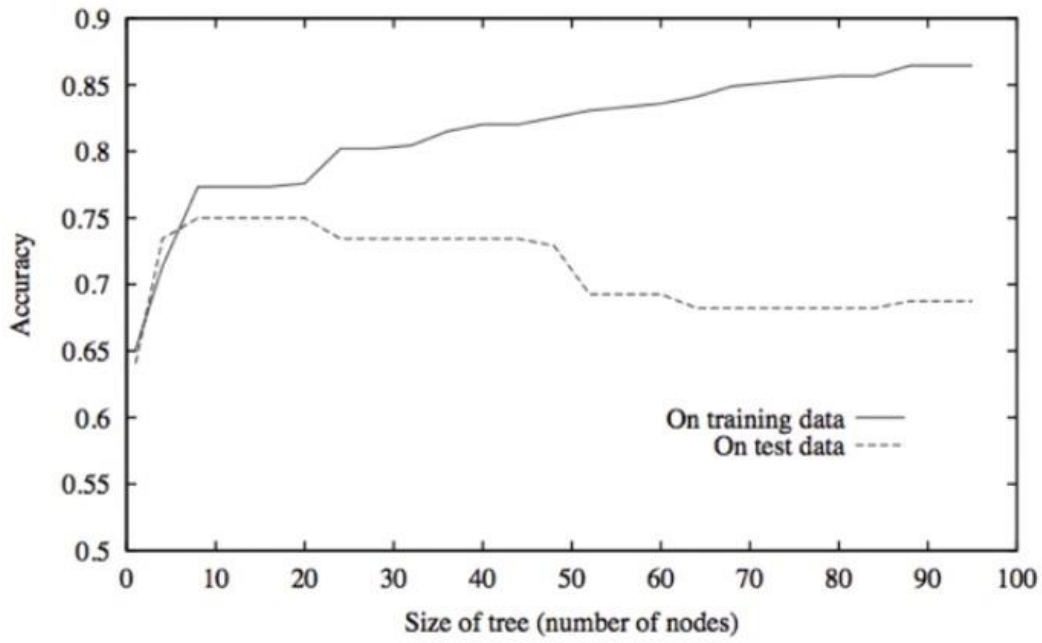
$= H(S) - {}^{8}/_{14}\, H(S_{weak}) - {}^{6}/_{14}\, H(S_{weak})$

$= 0.94 - {}^{8}/_{14} * 0.81 - {}^{6}/_{14} * 1.0$

$= 0.049$

Overfitting in Decision Trees

- Always classifies perfectly
  - Use of singleton classes
- Classification of test data problematic
- To build an effective decision tree, stop its growth before it becomes more specific



Copyright © 2014 Victor Lavrenko

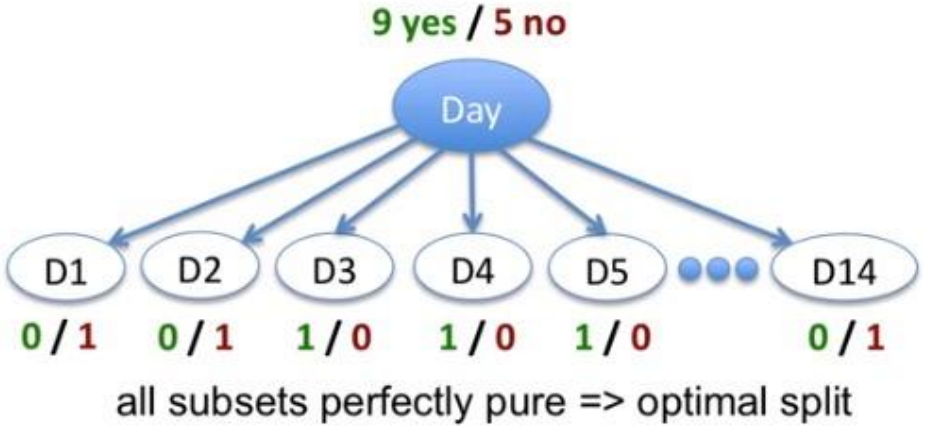
Figure credit: Tom Mitchell, 1997

Prevent Overfitting

- Avoid splitting when statistically insignificant
  - 2 samples vs. 100 samples
- Sub-Tree replacement
  - Construct validation set out of training set
  - Remove sub-trees (iterate over intermediate nodes) and test validation set
    - Greedy approach. Take sub-tree that gives the best result
  - Pick the tree that gives best results on validation set
  - Remove sub-trees while validation set result improves

Problems with Information Gain

- Biased towards attributes with many values
- Will not work with new data



- Use a Gain Ratio

$$SplitEntropy(S,A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}$$

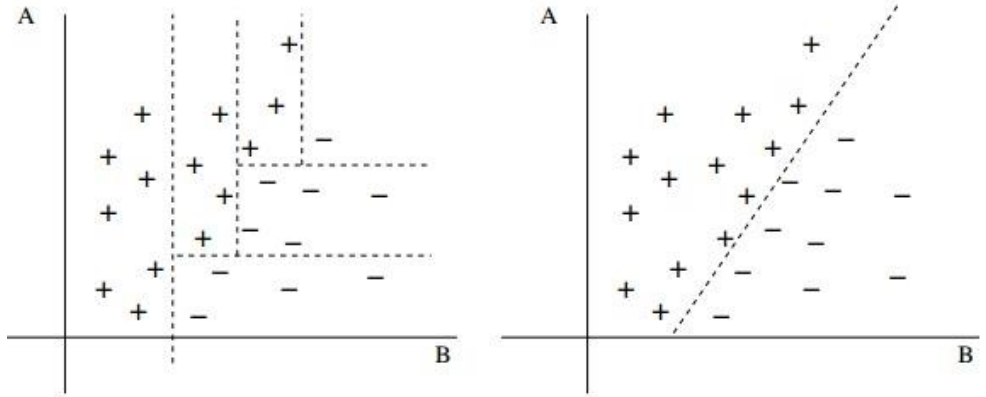
A … candidate attribute
V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

penalizes attributes with many values

Limitations of ID3

Strengths	Weaknesses
• An all-purpose classifier that does well on most problems • Highly automatic learning process, which can handle numeric or nominal features, as well as missing data • Excludes unimportant features • Can be used on both small and large datasets • Results in a model that can be interpreted without a mathematical background (for relatively small trees) • More efficient than other complex models	• Decision tree models are often biased toward splits on features having a large number of levels • It is easy to overfit or underfit the model • Can have trouble modeling some relationships due to reliance on axis-parallel splits • Small changes in the training data can result in large changes to decision logic • Large trees can be difficult to interpret and the decisions they make may seem counterintuitive



C5.0 and Overfitting

- It prunes using reasonable defaults.
  - Strategy is to post-prune the tree.
  - First grows a large tree that overfits the training data.
  - Nodes and branches that have little effect on the classification errors are removed.
    - Entire branches are moved further up the tree or replaced by simpler decisions.

C5.0 Boosting Accuracy

- Adaptive boosting: Many decision trees are built and the trees vote on the best class for each example
  - Combining a number of weak performing learners create a team that is much stronger than any of the learners alone.
    - Each learner has strengths and weaknesses and they may be better or worse in solving certain problems. Using a combination of several learners with complementary strengths and weaknesses can therefore dramatically improve the accuracy of a classifier.

C5.0 Making Mistakes Costly

- Allows us to assign a penalty to different types of errors, in order to discourage a tree from making more costly mistakes.
  - Penalties are designated in a cost matrix
- Predicted and actual values take two values, yes or no
  - Describe a 2 x 2 matrix, using a list of two vectors, each with two values.
  - Create Error Matrix

```
error_cost <- matrix(c(0, 1, 4, 0), nrow = 2, dimnames = matrix_dimensions)
```

  - Usage

```
> credit_cost <- C5.0(credit_train[-17], credit_train$default, costs = error_cost)
```

C5.0 Output

```
C5.0 [Release 2.07 GPL Edition]
-------------------------------

Class specified by attribute `outcome'

Read 900 cases (17 attributes) from undefined.data

Decision tree:

checking_balance in {> 200 DM,unknown}: no (412/50)
checking_balance in {< 0 DM,1 - 200 DM}:
:...credit_history in {perfect,very good}: yes (59/18)
    credit_history in {critical,good,poor}:
    :...months_loan_duration <= 22:
        :...credit_history = critical: no (72/14)
        :   credit_history = poor:
        :   :...dependents > 1: no (5)
        :   :   dependents <= 1:
        :   :   :...years_at_residence <= 3: yes (4/1)
        :   :       years_at_residence > 3: no (5/1)
```

	predicted default		
actual default	no	yes	Row Total
no	59 0.590	8 0.080	67
yes	19 0.190	14 0.140	33
Column Total	78	22	100

Classification Rules

- Classification rules represent knowledge in the form of logical if-else statements that assign a class to unlabeled examples.
  - Conists of an antecedent (combinations of feature values) and a consequent (class value)

    "if X, then Y"
  - Example

    "if the hard drive is making a clicking sound, then it is about to fail."
- Rule learners used similar to decision tree learners
- Like decision trees, used for applications that generate knowledge for future action
  - Identifying conditions that lead to a hardware failure in mechanical devices
  - Describing the key characteristics of groups of people for customer segmentation
  - Finding conditions that precede large drops or increases in the prices of shares on the stock market