# Data Mining & Machine Learning

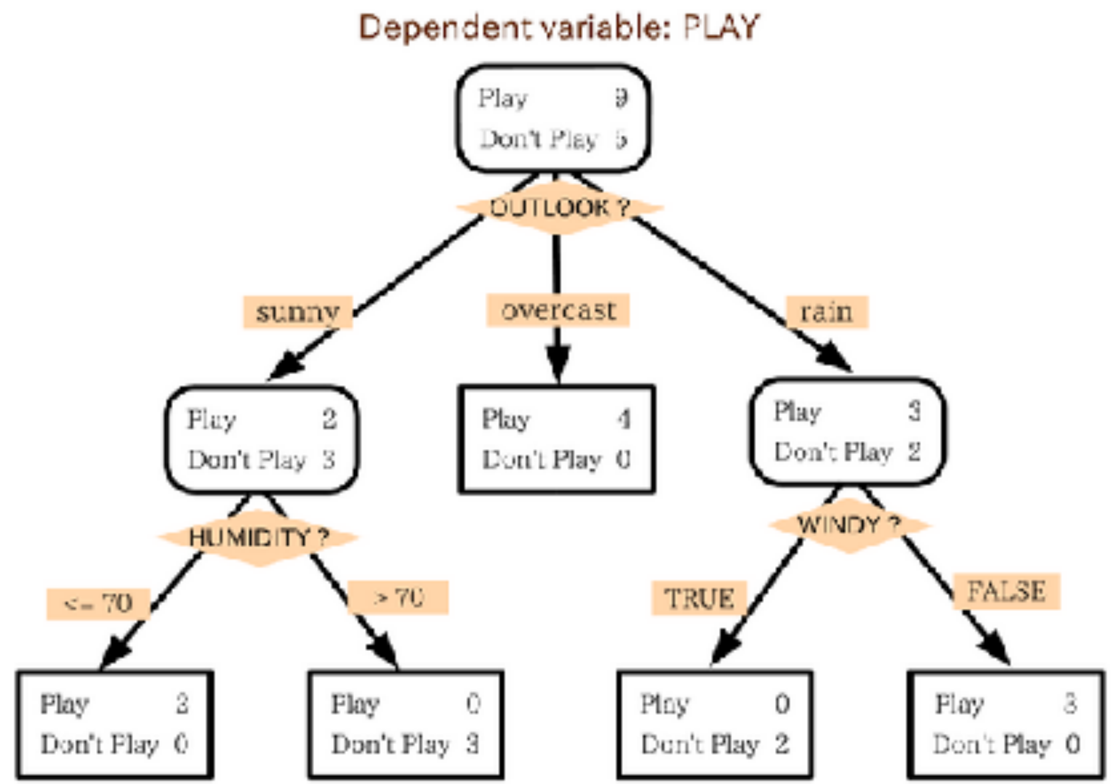
CS37300
Purdue University

Sept 18, 2017

# Decision trees

# Tree models

- Easy to understand knowledge representation
- Can handle mixed variables
- Recursive, divide and conquer learning method
- Efficient inference



# Tree learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select **best** attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- Other issues:
  - How to construct features
  - When to stop growing
  - Pruning irrelevant parts of the tree

| Fraud | Age | Degree | StartYr | Series7 |
|---|---|---|---|---|
| + | 22 | Y | 2005 | N |
| - | 25 | N | 2003 | Y |
| - | 31 | Y | 1995 | Y |
| - | 27 | Y | 1999 | Y |
| + | 24 | N | 2006 | N |
| - | 29 | N | 2003 | N |

**Score each attribute split for these instances:**
***Age, Degree, StartYr, Series7***

**Y** **choose split on *Series7*** **N**

| Fraud | Age | Degree | StartYr | Series7 |
|---|---|---|---|---|
| - | 25 | N | 2003 | Y |
| - | 31 | Y | 1995 | Y |
| - | 27 | Y | 1999 | Y |

| Fraud | Age | Degree | StartYr | Series7 |
|---|---|---|---|---|
| + | 22 | Y | 2005 | N |
| + | 24 | N | 2006 | N |
| - | 29 | N | 2003 | N |

**Y** **choose split on *Age>28*** **N**

**Score each attribute split for these instances:**
***Age, Degree, StartYr***

| Fraud | Age | Degree | StartYr | Series7 |
|---|---|---|---|---|
| - | 29 | N | 2003 | N |

| Fraud | Age | Degree | StartYr | Series7 |
|---|---|---|---|---|
| + | 22 | Y | 2005 | N |
| + | 24 | N | 2006 | N |

```
DecisionTree(examples, classLabel, attributes)
    features <- {}
    for each attribute
       for each attribute value
           create feature f
           features <- features + f
    create root node of tree
    growTree(root, examples, features)
```

---

```
growTree(node, examples, features)
    maxScore <- 0
    maxFeature <- null
    for each feature in features
       calculate score of feature on examples
           if score > maxScore & stopping criteria not met
           maxFeature <- feature; maxScore <- score
    if maxFeature is null
       nodeClassDist <- distribution of classLabel in examples   //to make predictions
       return //stop growing
    else
       nodeFeature <- maxFeature
       create nodes leftChild and rightChild
       lChildExamples <- examples that pass nodeFeatureTest
       rChildExamples <- examples that fail nodeFeatureTest
       //recurse on partitioned data
       growTree(leftChild, leftChildExamples, features)
       growTree(rightChild, rightChildExamples, features)
```

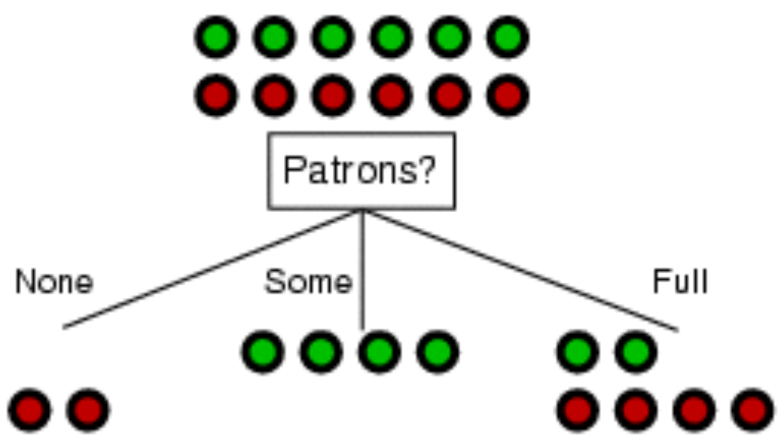
# Tree models

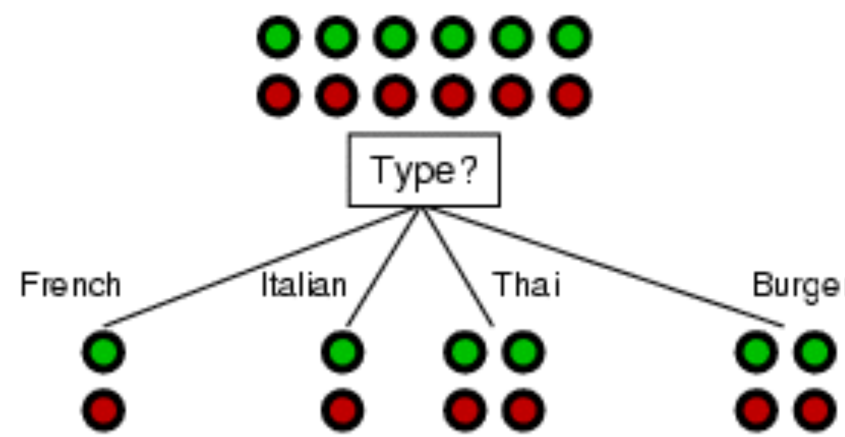
- Most well-known systems
  - CART: Breiman, Friedman, Olshen and Stone
  - ID3, C4.5: Quinlan
- How do they differ?
  - **Split scoring function**
  - Stopping criterion
  - Pruning mechanism
  - Predictions in leaf nodes

# Scoring functions: Local split value

# Choosing an attribute/feature

- Idea: a good feature splits the examples into subsets that distinguish among the class labels as much as possible... ideally into pure sets of "all positive" or "all negative"

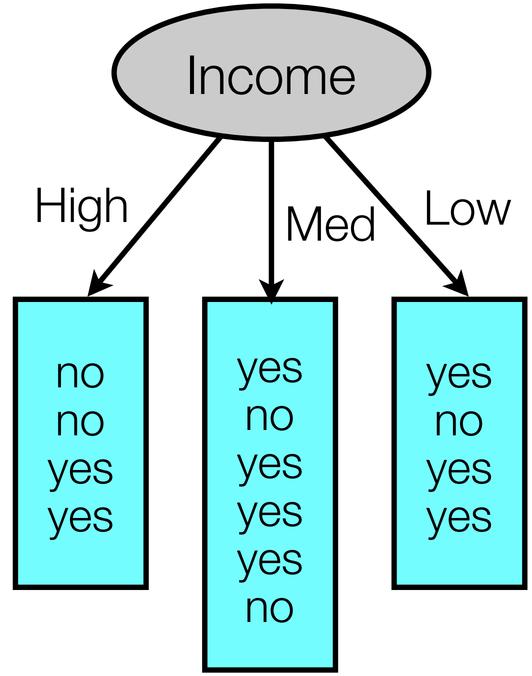


# Association between attribute and class label

Data

| age | income | student | credit_rating | buys_computer |
|---|---|---|---|---|
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |




**Contingency table**

Class label value

Attribute value

| | Buy | No buy |
|---|---|---|
| High | 2 | 2 |
| Med | 4 | 2 |
| Low | 3 | 1 |

# Information gain

- How much does a feature split decrease the entropy?

$$Gain(S, A) = \underline{Entropy(S)} - \sum_{v \in values(A)} \frac{|S_A|}{|S|} Entropy(S_A)$$

| age | income | student | credit_rating | buys_computer |
|---|---|---|---|---|
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

*Entropy(D)*

= -9/14 log 9/14 -5/14 log 5/14

= 0.9400

# Entropy

- Used to quantify the amount of randomness of a probability distribution.
- Definition: The **entropy** H(X) of a discrete random variable X is defined by:

$$H(X) = -\sum_{x} p(x) \log_2 p(x)$$

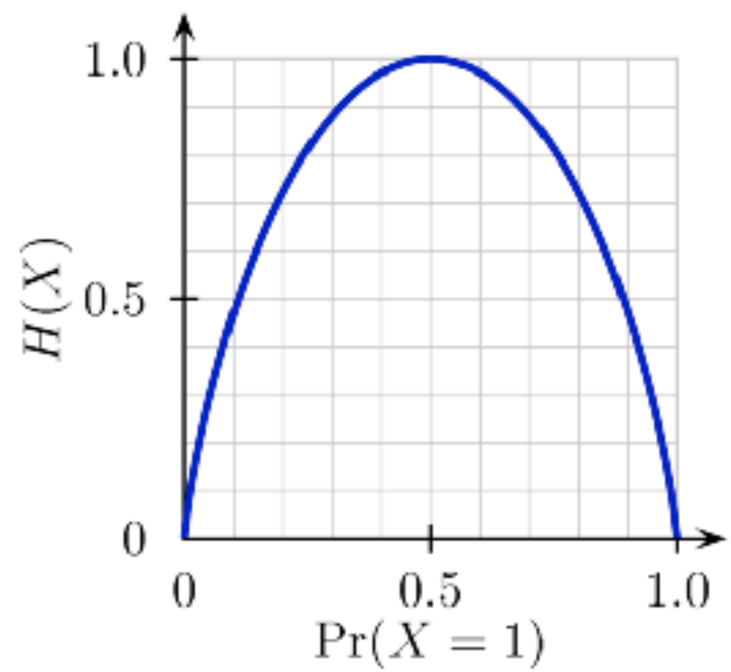
# Entropy of a random variable

A completely random binary variable with X=[0.5,0.5] has entropy:
H(X) = -(0.5 log 0.5 + 0.5 log 0.5) = -(-.05 + -0.5) = 1

A deterministic variable with X=[1,0] has entropy:
H(X) = -(1 log 1 + 0 log 0) = -(0+0) = 0

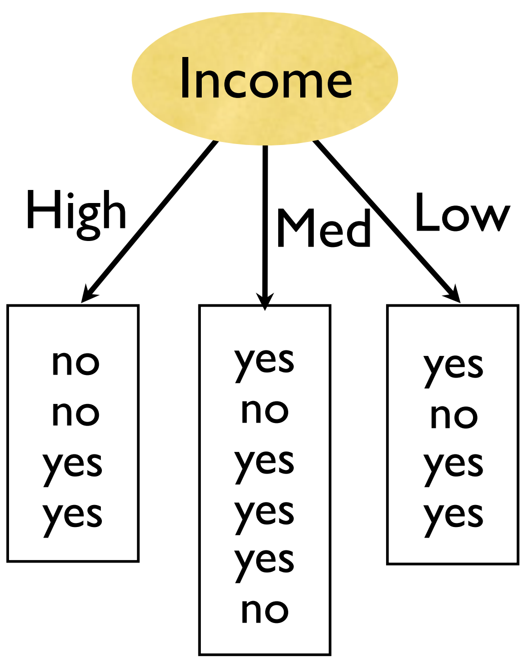
A biased variable with X=[0.75,0.25] has entropy:
H(X) = 0.8113

The entropy of a probability distribution ***p*** expresses the ***amount of uncertainty*** that we have about the values of X



# Information gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_A|}{|S|} Entropy(S_A)$$




*Entropy(Income=high)*
= -2/4 log 2/4 -2/4 log 2/4 = 1

*Entropy(Income=med)*
= -4/6 log 4/6 -2/6 log 2/6 = 0.9183

*Entropy(Income=low)*
= -3/4 log 3/4 -1/4 log 1/4 = 0.8113

*Gain(D,Income)*
= 0.9400 - (4/14 [1] + 6/14 [0.9183] + 4/14 [0.8113])
= 0.029

# Gini gain

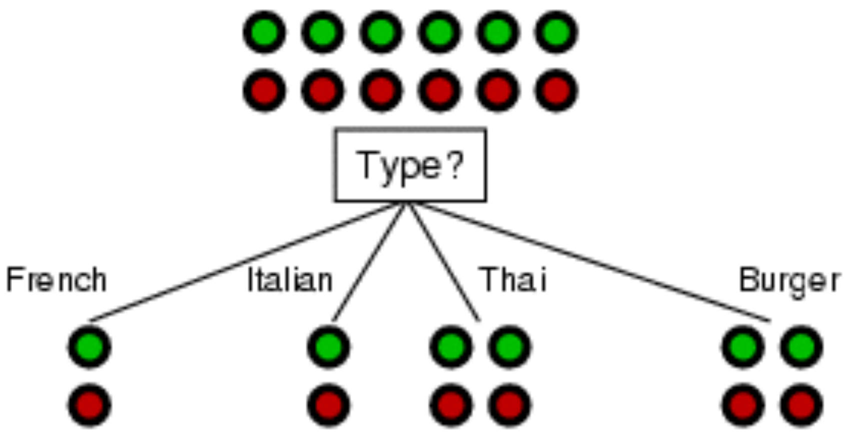
- Similar to information gain
- Uses gini index instead of entropy

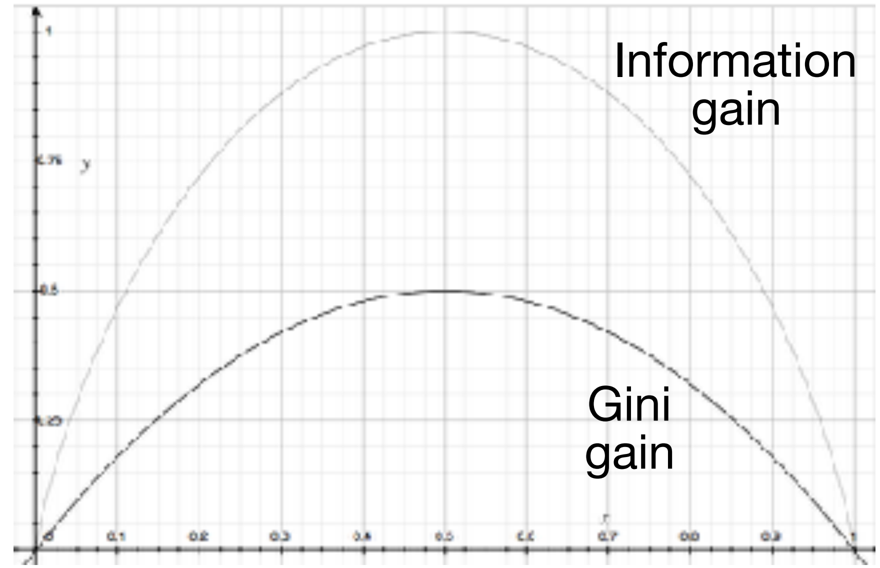
$$Gini(X) = 1 - \sum_{x} p(x)^2$$

- Measures decrease in gini index after split:

$$Gain(S, A) = Gini(S) - \sum_{v \in values(A)} \frac{|S_A|}{|S|} Gini(S_A)$$

# Comparing information gain to gini gain

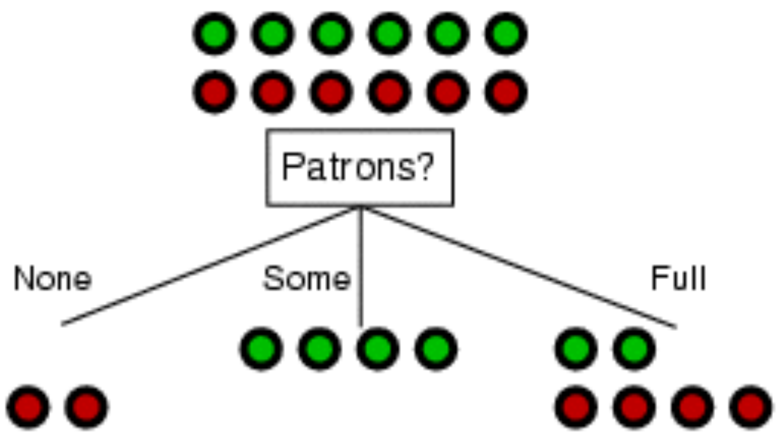


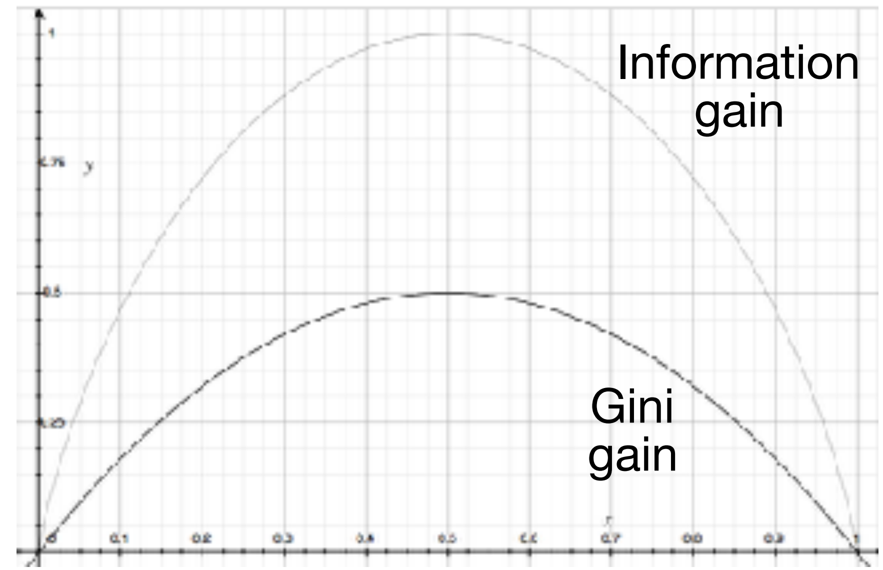




$$IG = 0$$

$$GG = 0$$

# Comparing information gain to gini gain

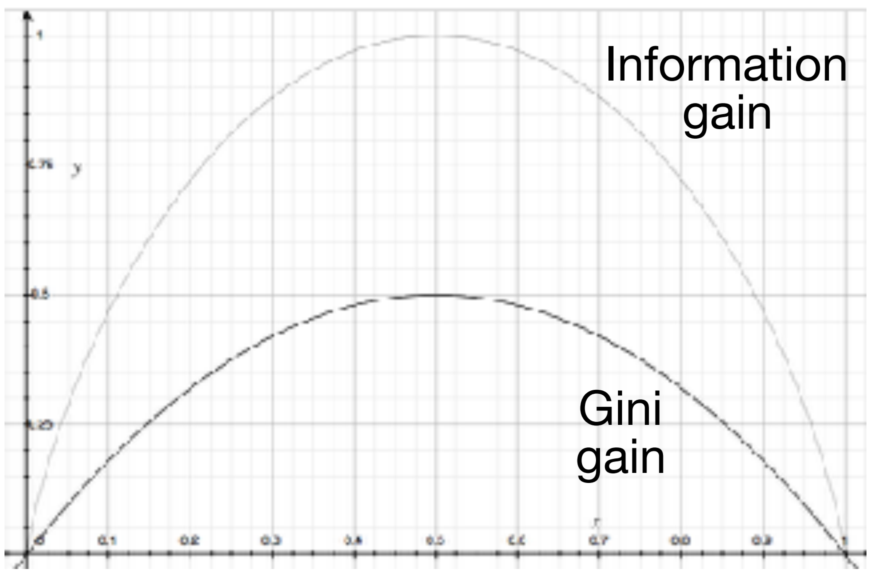



$$IG = 1.0 - \left[\frac{2}{12}\,0\right] - \left[\frac{4}{12}\,0\right] - \left[\frac{6}{12}\,0.919\right] = 0.541$$

$$GG = 0.5 - \left[\frac{2}{12}\,0\right] - \left[\frac{4}{12}\,0\right] - \left[\frac{6}{12}\,0.444\right] = 0.278$$

# How does score function affect feature selection?





$66\%$ split $:Entropy = 0.919$ $85\%$ split $:Entropy = 0.610$

$Gini \times 2 = 0.889$ $Gini \times 2 = 0.510$

**Lower scores produce larger gains**

# Chi-Square score

- Widely used to test independence between two categorical attributes (e.g., feature and class label)
- Considers counts in a contingency table and calculates the normalized squared deviation of observed (predicted) values from expected (actual) values

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

- Sampling distribution is known to be chi-square distributed, given that cell counts are above minimum thresholds

# Contingency tables

| | Buy | No buy | |
|---|---|---|---|
| High | 2 | 2 | 4 |
| Med | 4 | 2 | 6 |
| Low | 3 | 1 | 4 |
| | 9 | 5 | 14 |

# Calculating expected values for a cell

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

| | | Class | |
|---|---|---|---|
| | | + | – |
| Attribute | 0 | a | b |
| | 1 | c | d |

N

$$o_{(0,+)} = a$$

$$\begin{aligned}
e_{(0,+)} &= p(A = 0, C = +) \cdot N \\
&= p(A = 0)p(C = +|A = 0) \cdot N \\
&= p(A = 0)p(C = +) \cdot N \qquad \text{(assuming independence)} \\
&= \left[\frac{a+b}{N}\right] \cdot \left[\frac{a+c}{N}\right] \cdot N
\end{aligned}$$

# Example calculation

**Observed**

| | Buy | No buy |
|---|---|---|
| High | 2 | 2 |
| Med | 4 | 2 |
| Low | 3 | 1 |

**Expected**

| | Buy | No buy |
|---|---|---|
| High | 2.57 | 1.43 |
| Med | 3.86 | 2.14 |
| Low | 2.57 | 1.43 |

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = \left(\frac{(2-2.57)^2}{2.57}\right) + \left(\frac{(4-3.86)^2}{3.86}\right) + \left(\frac{(3-2.57)^2}{2.57}\right)$$

$$+ \left(\frac{(2-1.43)^2}{1.43}\right) + \left(\frac{(2-2.14)^2}{2.14}\right) + \left(\frac{(1-1.43)^2}{1.43}\right)$$

$$= 0.57$$

# Tree learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select **best** attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- Other issues:
  - How to construct features
  - ***When to stop growing***
  - ***Pruning irrelevant parts of the tree***

# When to stop growing

- Full growth methods
  - All samples for at a node belong to the same class
  - There are no attributes left for further splits
  - There are no samples left
- What impact does this have on the quality of the learned trees?
  - Trees **overfit** the data and accuracy decreases
  - Pruning is used to avoid overfitting

# Pruning

- Postpruning
  - Use a separate set of examples to evaluate the utility of pruning nodes from the tree (after tree is fully grown)
- Prepruning
  - Apply a statistical test to decide whether to expand a node
  - Use an explicit measure of complexity to penalize large trees (e.g., Minimum Description Length)

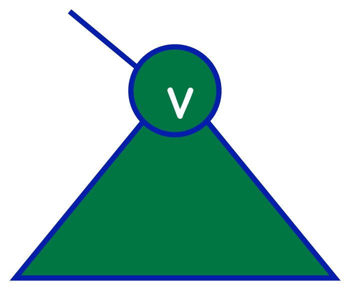
# Algorithm comparison

- CART
  - Evaluation criterion:
    **Gini index**
  - Search algorithm:
    Simple to complex,
    hill-climbing search
  - Stopping criterion:
    When leaves are pure
  - Pruning mechanism:
    **Cross-validation to select gini threshold**

- C4.5
  - Evaluation criterion:
    **Information gain**
  - Search algorithm:
    Simple to complex,
    hill-climbing search
  - Stopping criterion:
    When leaves are pure
  - Pruning mechanism:
    **Reduce error pruning**

# Example: reduced error pruning

- Use **pruning set** to estimate accuracy in sub-trees and for individual nodes
- Let T be a sub-tree rooted at node v




- Define:

$$\text{Gain from prunning at } v = \#\text{misclassification in } T - \#\text{misclassification at } v$$

- Repeat: Prune at node with largest gain until until only negative gain nodes remain
- “Bottom-up restriction”: T can only be pruned if it does not contain a sub-tree with lower error than T

Source: www.ailab.si/blaz/predavanja/uisp/slides/uisp05-PostPruning.ppt

# Pre-pruning methods

- Stop growing tree at some point during top-down construction when there is no longer sufficient data to make reliable decisions
- Approach:
  - Choose threshold on feature score
  - Stop splitting if the best feature score is below threshold

# Determine chi-square threshold analytically

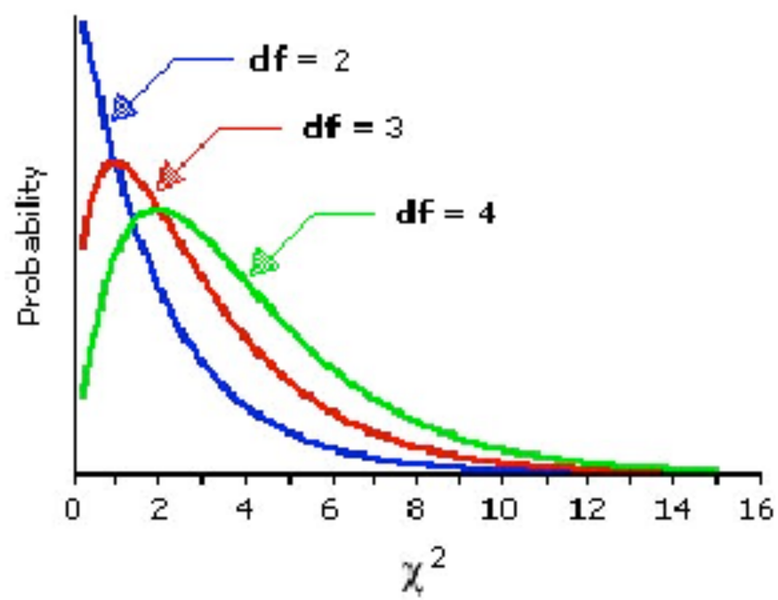
- Stop growing when chi-square feature score is not **statistically significant**

- Chi-square has known sampling distribution, can look up significance threshold

  - Degrees of freedom= (#rows-1)(#cols-1)

  - 2X2 table:
    3.84 is 95% critical value

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$



How do these pruning approaches change the search procedure?