Classification with Decision Tree Induction

- This algorithm makes Classification Decision for a test sample with the help of tree like structure (Similar to Binary Tree OR k-ary tree)
  - Nodes in the tree are attribute names of the given data
  - Branches in the tree are attribute values
  - Leaf nodes are the class labels

- Supervised Algorithm (Needs Dataset for creating a tree)

- Greedy Algorithm (favourite attributes first)

Building Decision Tree

- Two step method
  - Tree Construction
    1. Pick an attribute for division of given data
    2. Divide the given data into sets on the basis of this attribute
    3. For every set created above - repeat 1 and 2 until you find leaf nodes in all the branches of the tree - Terminate
  - Tree Pruning (Optimization)
    - Identify and remove branches in the Decision Tree that are not useful for classification
      - Pre-Pruning
      - Post Pruning

Assumptions and Notes for Basic Algorithm

- Attributes are categorical
  - if continuous-valued, they are discretized in advanced
- Examples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- At start, all the training examples are at the root

Algorithm at work….
(Tree Construction - Step 1 & 2)

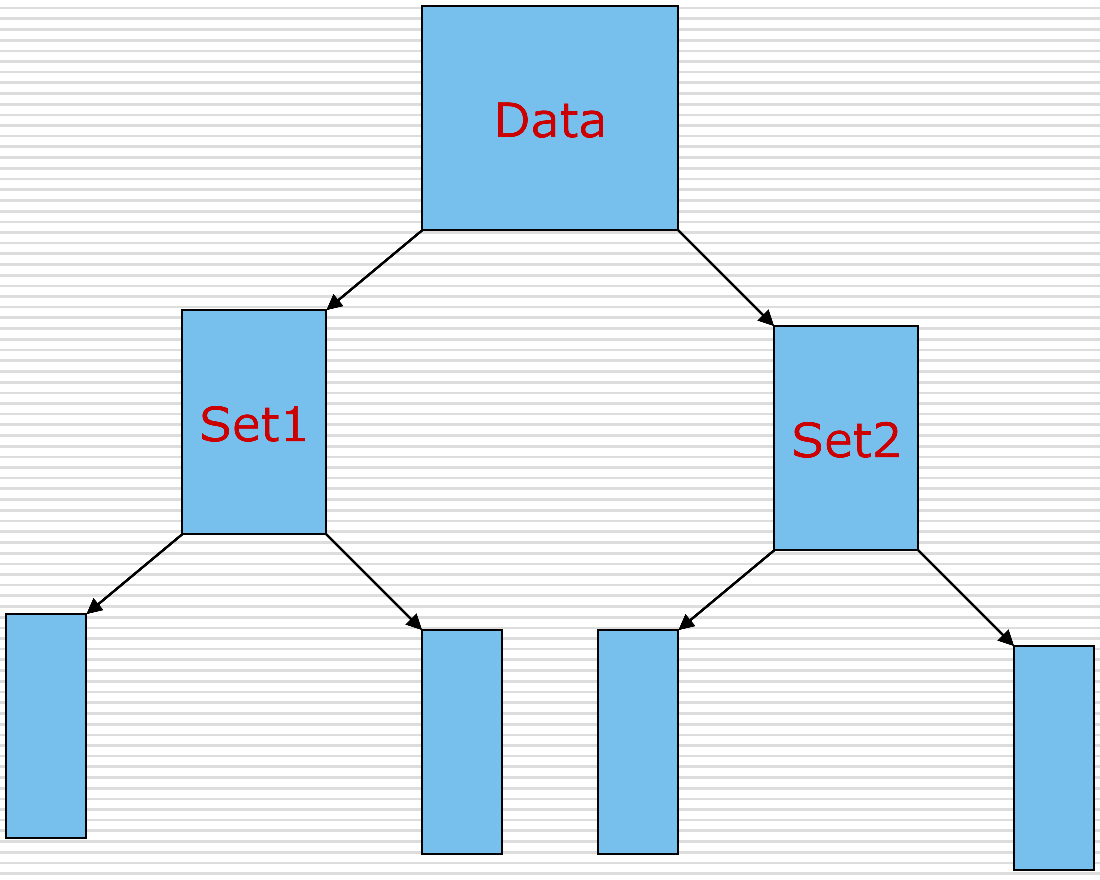
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent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

Given data

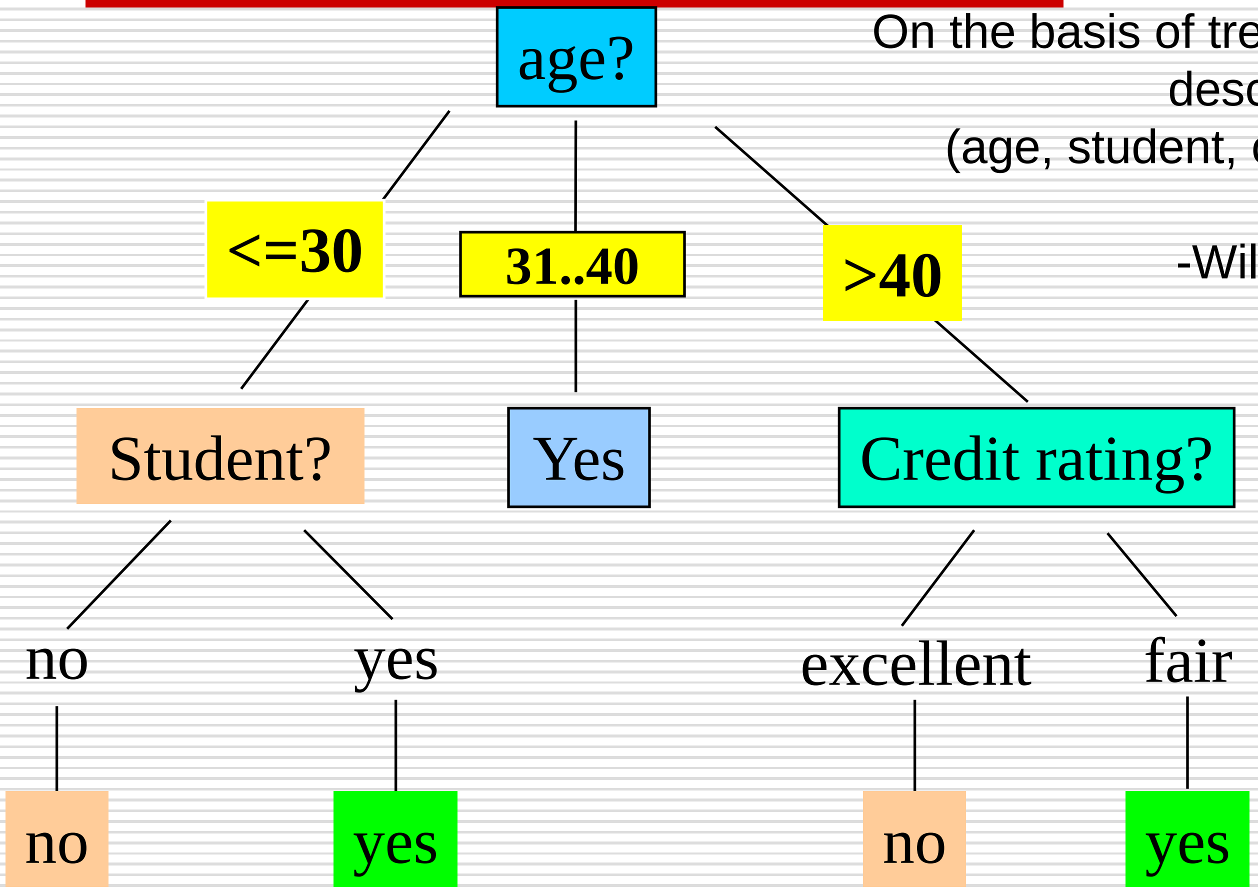
Three Data Sets formed after division at root node on the basis of “age” attribute

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

Algorithm in action….



Final Decision Tree



On the basis of tree constructed in the manner described, classify a test sample (age, student, creditrating, buys_computer) **(<=30, yes, excellent, ?)**

-Will this student buy computer?

Tree Construction (Termination Conditions)

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left

Attribute Selection Advancements

- We want to find the most "useful" attribute in classifying a sample. Two measures of usefulness –
  - Information Gain
    - Attributes are assumed to be categorical
  - Gini Index (IBM IntelligentMiner)
    - Attributes are assumed to be contineous
    - Assume there exist several possible split values for each attribute

How to calculate Information "Gain"

- In a given Dataset, assume there are two classes, *P* and *N* (yes and no from example)
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Entropy

- Entropy measures the impurity of a set of samples.
  - It is lowest, if there is at most one class present, and it is highest, if the proportions of all present classes are equal. That is,
    - If all examples are positive or all negative, entropy is low (zero).
    - If half are positive and half are negative, entropy is high (1.0)

Information Gain in Decision Tree Induction

- Assume that using attribute A a set *S* will be partitioned into sets {*S1*, *S2* , …, *Sv*}
  - If *Si* contains *pi* examples of *P* and *ni* examples of *N*, the entropy, or the expected information needed to classify objects in all subtrees *Si* is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on *A. This is the expected reduction in entropy if we go with A.*

$$Gain(A) = I(p, n) - E(A)$$

Play-tennis example: which attribute do we take first

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

I (Humidity[9+,5-]) = .940

Humidity = high [3+,4-] E=0.985
Humidity=normal [6+,1-] E = .592
Gain(S, Humidity) = .940 – 7/14(.985) – (7/14).592 = .151

Windy = false [6+,2-], E = .811
Windy = true [3+,3-], E = 1.0

Gain (S, Windy) = .940 – (8/14)(.811 – (6/14)(1.0) = .048

Humidity split into two classes , one with a great split of 6+ and 1-. The other was not so great of 3+,3-
Wind split into two classes, one with an Ok split of 6+2-
And the other was terrible of 3+,3- (max entropy of 1.0).

So Humidity is the best attribute between these two.

Gain(S,outlook) = .246
Gain(S,humidity) = .151
Gain(S,wind) = .048
Gain(S,Temperature) = .029

Gini Index (IBM IntelligentMiner

- If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as $$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$
- where *pj* is the relative frequency of class *j* in *T.*
- If a data set *T* is split into two subsets *T1* and *T2* with sizes *N1* and *N2* respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest *ginisplit*(*T*) is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Extracting Classification Rules

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand

- Example

  IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"
  IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
  IF *age* = "31…40" THEN *buys_computer* = "*yes*"
  IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"
  IF *age* = "<=30" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Overfitting

- Generated Decision Tree is said to *overfit* the training data if,
  - *It results in poor accuracy to classify test samples*
  - *It has too many branches, that reflect anomalies due to noise or outliers*
- Two approaches to avoid overfitting –
  - Tree Pre-Pruning – Halt tree construction early – that is, do not split a node if the goodness measure falls below a threshold
    - It is difficult to choose appropriate threshold
  - Tree Post-Pruning - Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Classifier Accuracy Estimation

- Why estimate a classifier accuracy?
  - Comparing classifiers for the given dataset (Different classifiers will favor different domain of datasets)
  - One needs to estimate how good the prediction will be.
- Methods of estimating accuracy
  - **Holdout** – randomly partition the given data into two independent sets and use one for training (typically 2/3rd) and the other for testing (1/3rd)
  - **k-fold cross-validation** – randomly partition the given data into 'k' mutually exclusive subsets (folds). Training and testing is performed k times.

Accuracy Improvement

- Methods
  - Bagging (Bootstrap aggregation) – Number of trees are constructed on subsets of given data and majority voting is taken from these trees to classify a test sample.
  - Boosting – attaching weights (importance) to the training samples and optimizing the weights during training and further using these weights to classify the test sample. Advantage – avoids outliers