CSE 4950/6950
Decision Tree

Instructor: AKM Kamrul Islam
aislam5@cs.gsu.edu
Dept. of Computer Science
Georgia State University

(Materials are highly adapted from different online sources)

Types of Classifier

- We can divide the large variety of classification approaches into roughly two main types

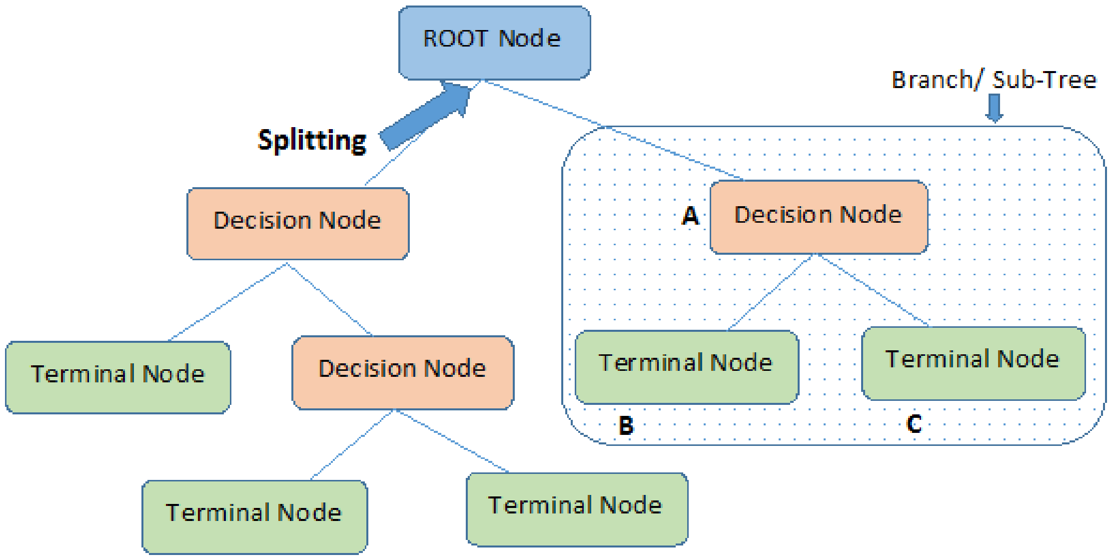
  1. Instance based classifiers
     - Use observation directly (no models)
     - e.g. K nearest neighbors

  2. Generative:
     - build a generative statistical model
     - e.g., Bayesian networks

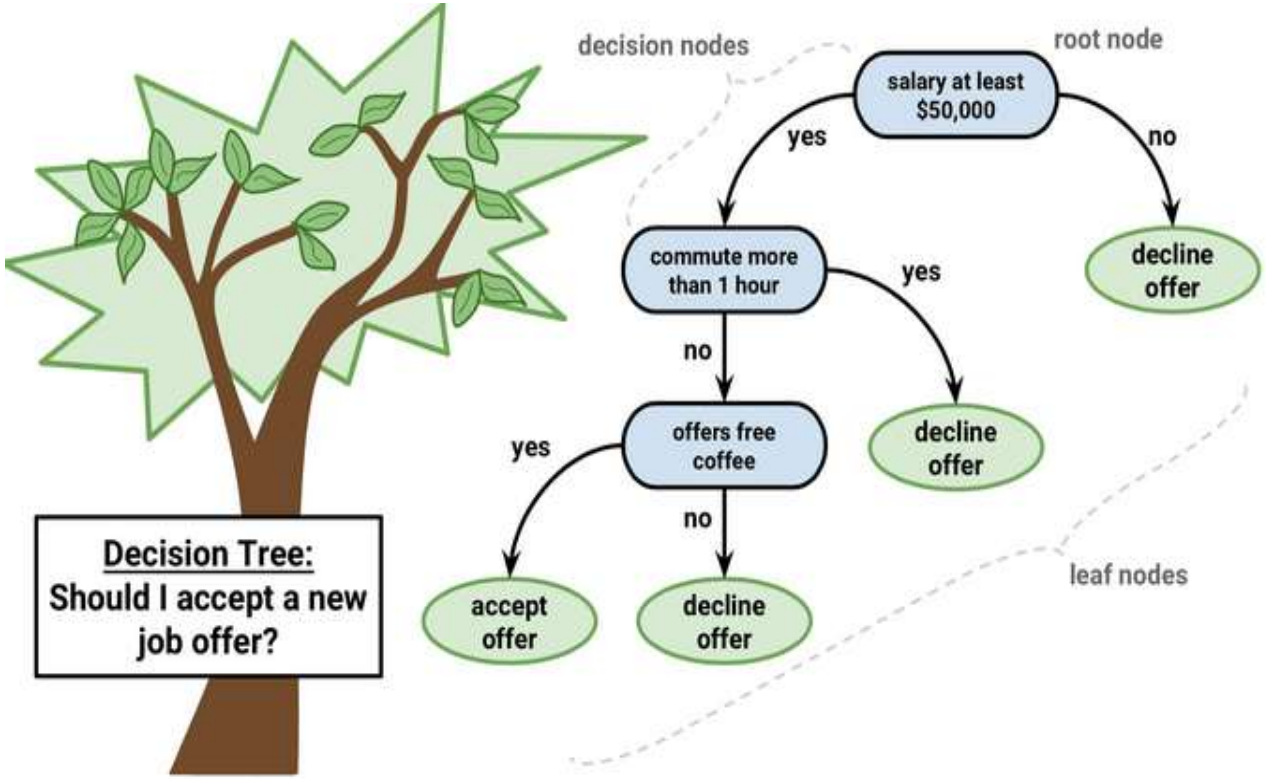
  3. Discriminative
     - directly estimate a decision rule/boundary
     - e.g., decision tree

Decision Tree

- Decision tree classify instances by sorting them down the tree from root to some leaf node
- Each node in the tree specifies a test of some attribute of the instance
- Each branch corresponds to one of the possible values for this attribute
- Each leaf associated with classification



A is a parent node of B and C

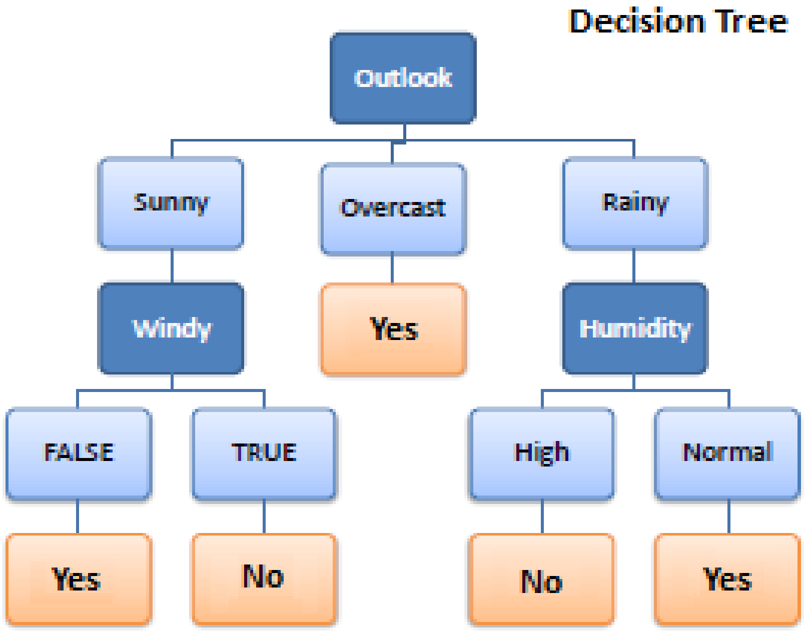


Decision Tree

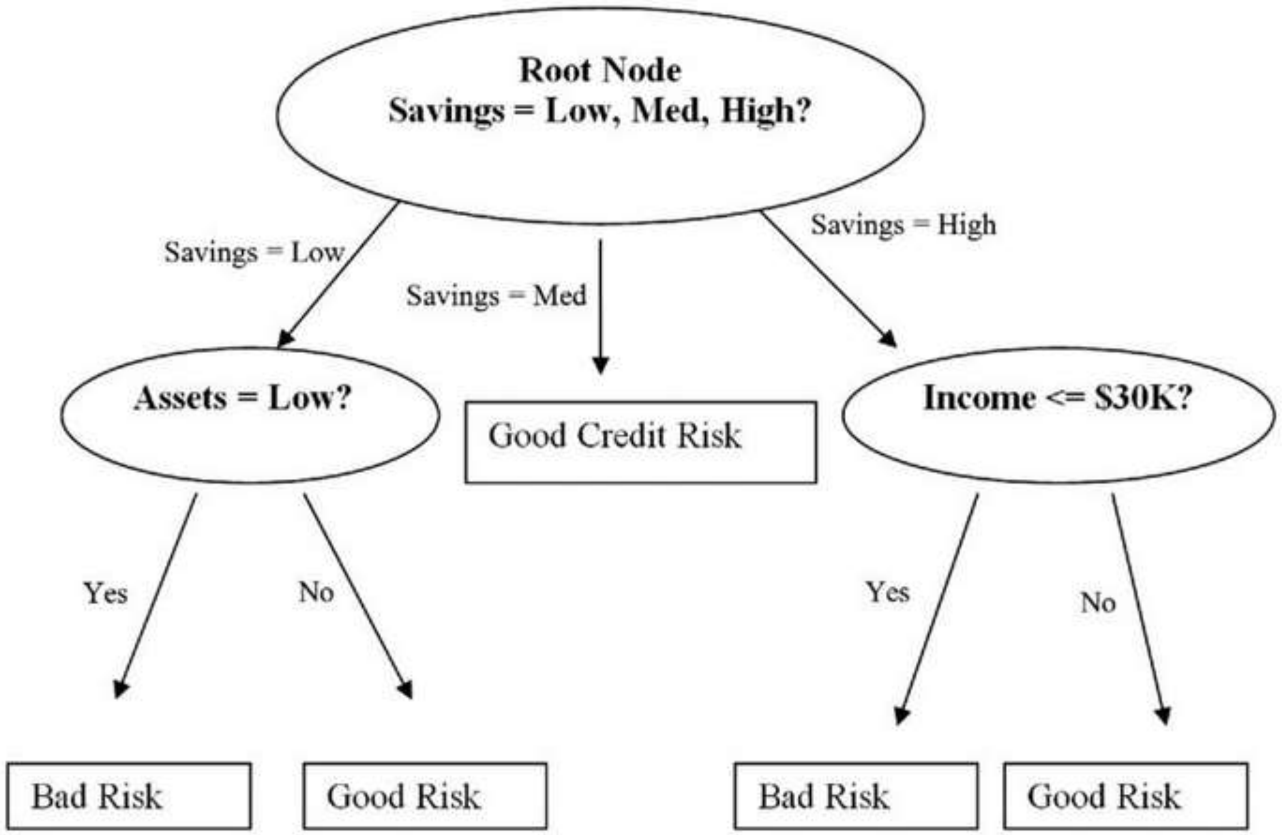
- Decision tree builds classification or regression models in the form of a tree structure.
- It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.
- The final result is a tree with **decision nodes** and **leaf nodes**.
- Example: A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

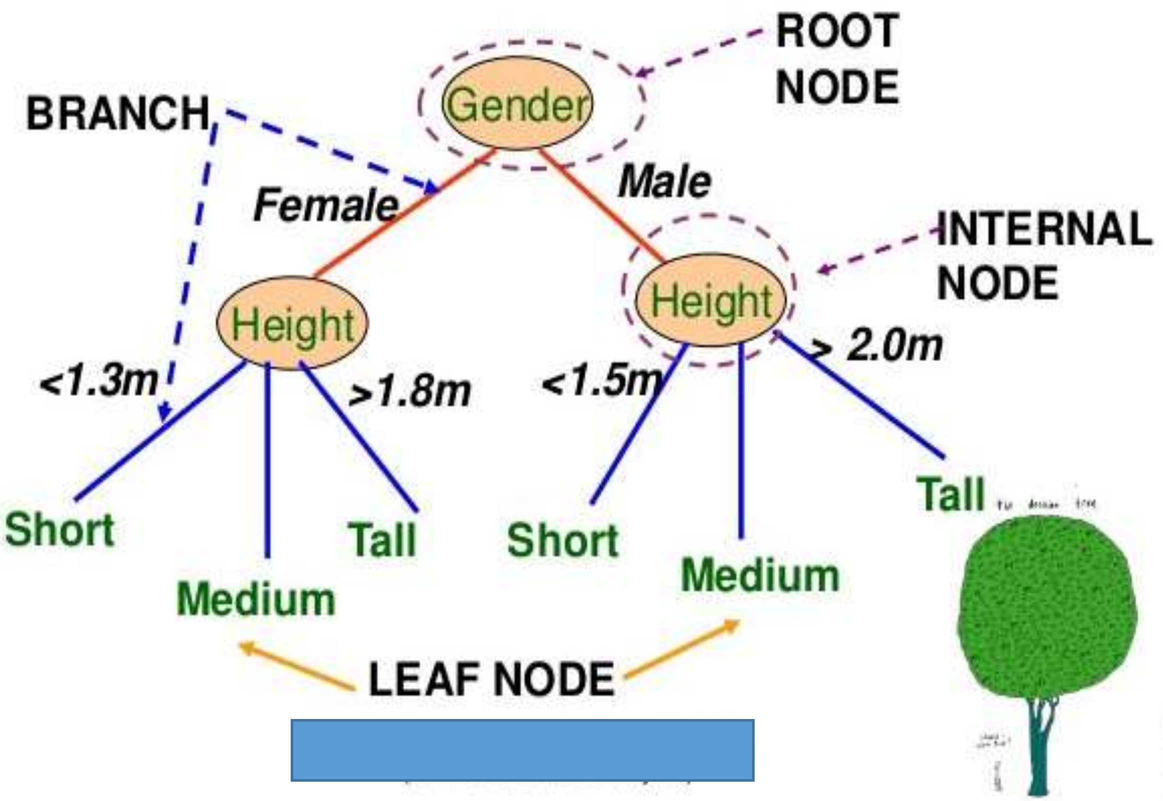
Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



Decision Tree





Decision Tree

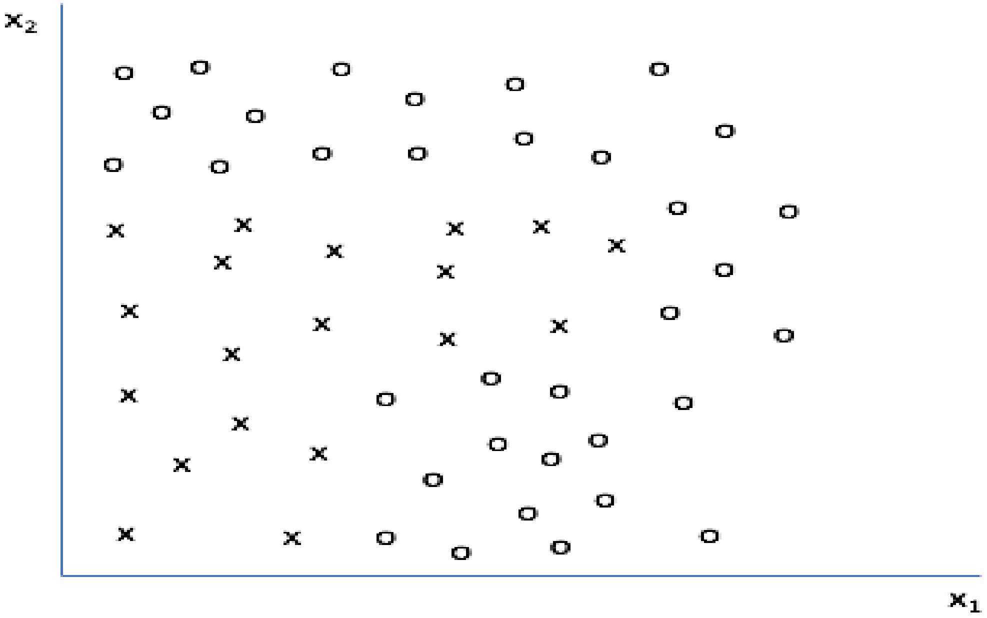
Here are some useful terms for describing a decision tree:

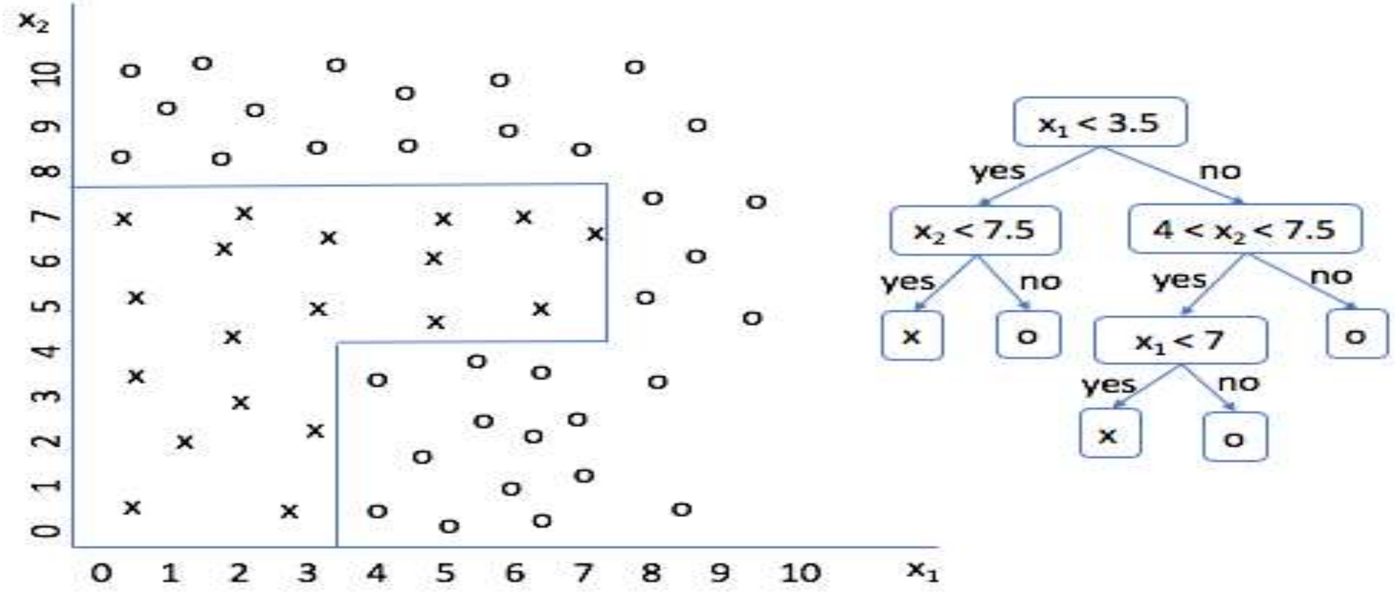
- **Root Node:** A root node is at the beginning of a tree. It represents entire population being analyzed. From the root node, the population is divided according to various features, and those sub-groups are split in turn at each decision node under the root node.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, it’s a decision node.
- **Leaf Node or Terminal Node:** Nodes that do not split are called leaf or terminal nodes.
- **Pruning:** Removing the sub-nodes of a parent node is called pruning. A tree is grown through splitting and shrunk through pruning. You can say opposite process of splitting.
- **Branch or Sub-Tree:** A sub-section of decision tree is called branch or a sub-tree, just as a portion of a graph is called a sub-graph.
- **Parent Node and Child Node:** These are relative terms. Any node that falls under another node is a child node or sub-node, and any node which precedes those child nodes is called a parent node.

DT Classification

- Let's look at a two-dimensional feature set and see how to construct a decision tree from data.
- The goal is to construct a decision boundary such that we can distinguish from the individual classes present.

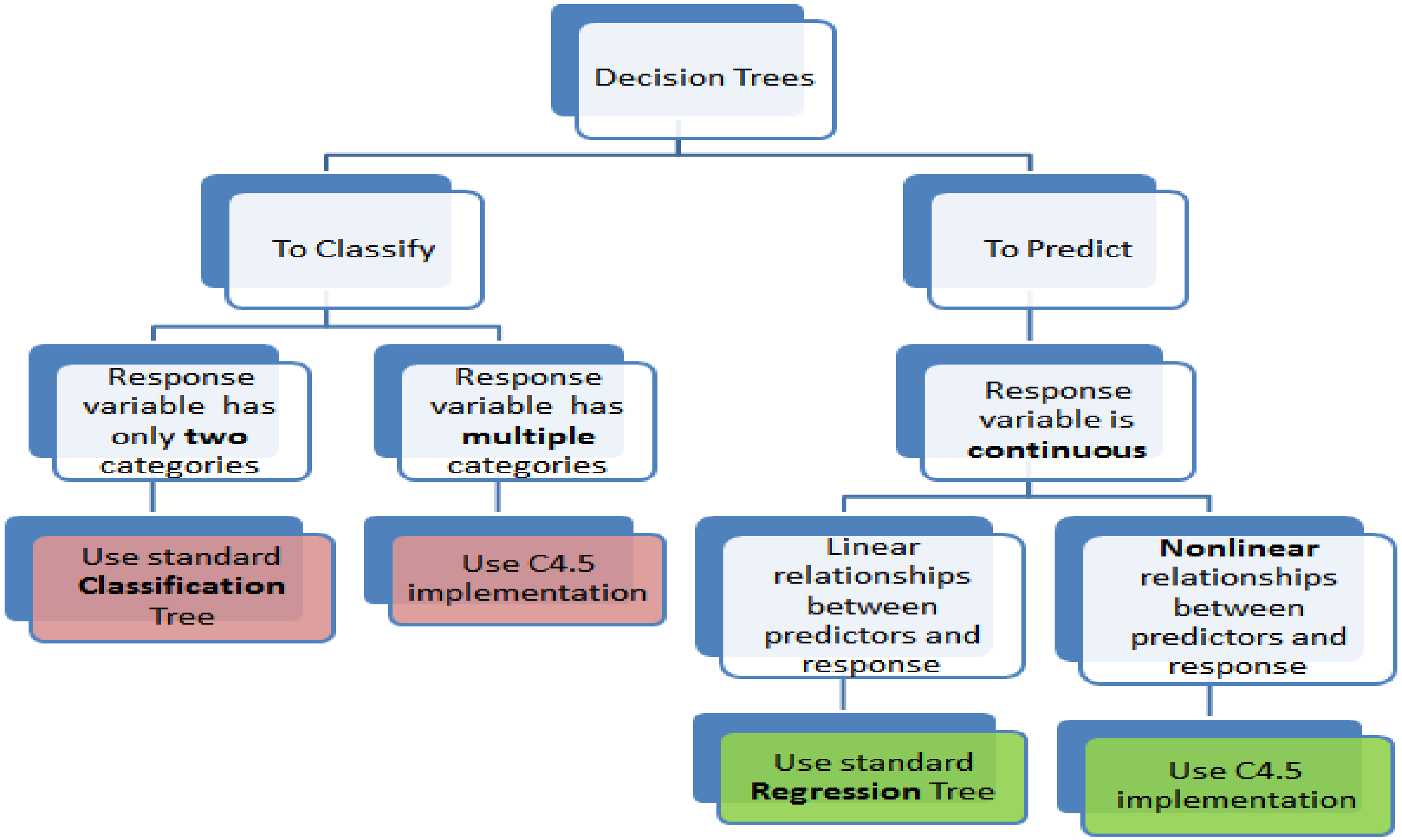
Any ideas on how we could make a decision tree to ***classify*** a new data point as "x" or "o"? Here's what I did.





Types of Decision Trees

This tree below summarizes at a high level the types of decision trees available.



ID3 Algorithm

- The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking.
- ID3 uses *Entropy* and *Information Gain* to construct a decision tree.
- Determine which attribute should be tested first. The best attribute will be used as root.
- For each branch, repeat the ID3 process
- In ZeroR model there is no predictor
- In OneR model we try to find the single best predictor, naive Bayesian includes all predictors using Bayes' rule and the independence assumptions between predictors but decision tree includes all predictors with the dependence assumptions between predictors.

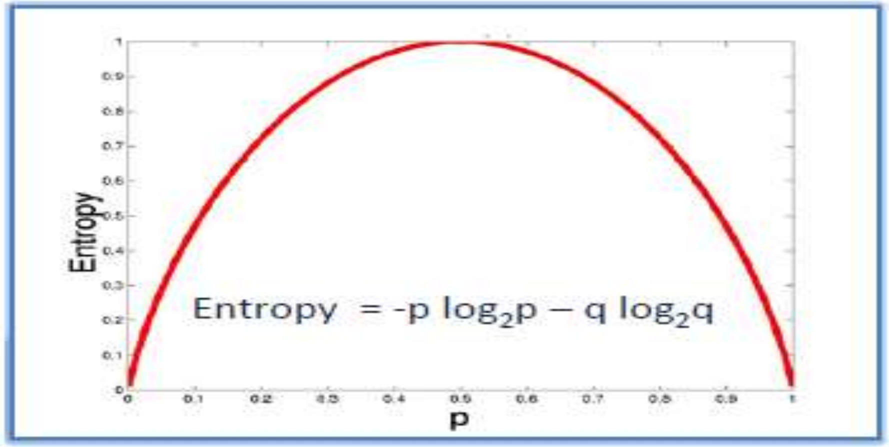
ID3 Algorithm

1. ID3 (Examples, Target_attribute, Attribute)
2. Create a Root node for the tree
3. If all examples are positive, Return the single-node tree, with label=+
4. If all examples are negative, Return the single-node tree, with label=-
5. If Attributes is empty, Return the single-node tree Root, with label=most common value of Target_attribute in Examples
6. Otherwise
   - A←the attribute from Attributes that best classifies Examples
   - The decision attribute for Root ←A
   - For each possible value, vi, of A,
     - Add a new tree branch below Root, Corresponding to the test A=vi
     - Let examples vi be the subset of Examples that have value vi for A
     - If examples vi is empty
       – Then below this new branch and a leaf node with label=most common value if Target_attribute in Examples
       – Else below this new branch add the subtree
         » ID(Examples vi, Target_attribute, Attributres-{A}))

7. End
8. Return Root

Entropy

- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).
- ID3 algorithm uses entropy to calculate the homogeneity of a sample.
- If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

- To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

  a) Entropy using the frequency table of one attribute

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

$= \text{Entropy}(0.36, 0.64)$

$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$

$= 0.94$

Entropy

- Suppose S is a collection of 14 examples of some Boolean concept, including 9 positive and 5 negative examples. Then the entropy of S relative to this Boolean classification is
  - Entropy([9+,5-])=-(9/14)log2(9/14)-(5/14)log2(5/14)=0.940
- Interpretation
  - Higher Entropy => Higher Uncertainty
  - Lower Entropy => Lower Uncertainty

- Suppose S is a collection of 20 examples with performance label, including 8 excellent, 10 good, and 2 poor examples. Then the entropy of S relative to this classification is
  - Entropy([8(E),10(G),2(P)]) = -(8/20)log2(8/20)-(10/20)log2(10/20)-(2/20)log2(2/20) = 1.3610

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

b) Entropy using the frequency table of two attributes

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Georgia State University

Information Gain

- How well a given attribute separates the training examples.
- Select candidate attributes at each step while growing the tree.
- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target

$$\begin{aligned} \text{Entropy(PlayGolf)} &= \text{Entropy}(5,9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94 \end{aligned}$$

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Information Gain

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

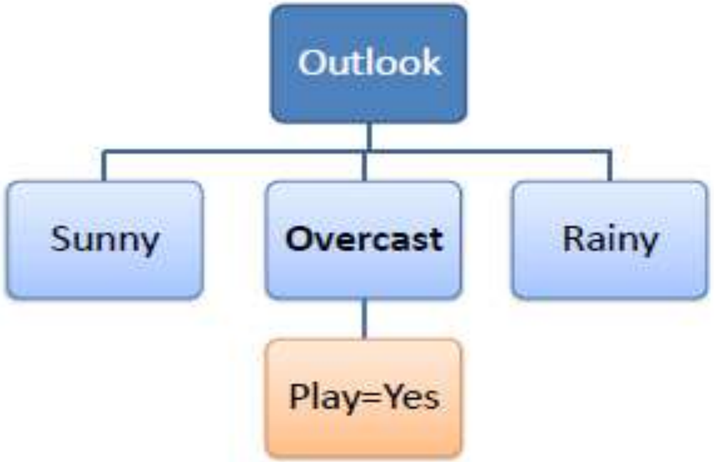
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes

Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Information Gain

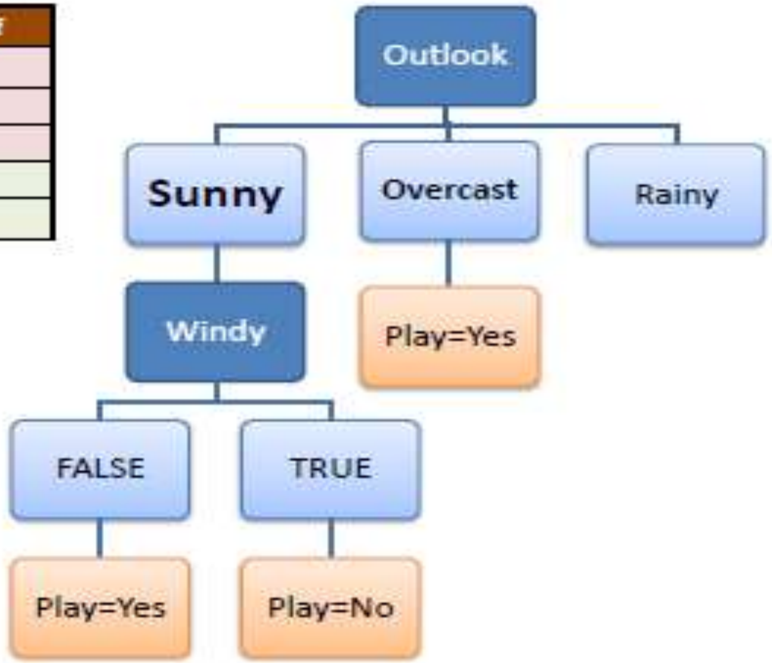
Step 4a: A branch with entropy of 0 is a leaf node

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

Example

$P(Li=yes) = 2/3$
$E(Li) = .91$
$E(Li \mid T) = 0.61$
$E(Li \mid Le) = 0.61$
$E(Li \mid D) = 0.36$
$E(Li \mid F) = 0.85$

$IG(Li \mid T) = .91-.61 = 0.3$

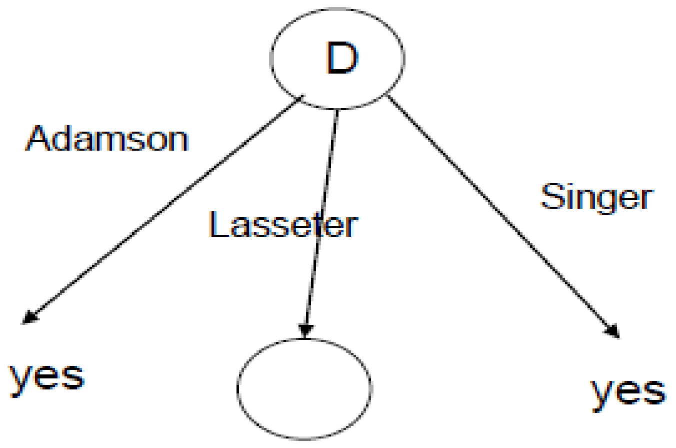
$IG(Li \mid Le) = .91-.61 = 0.3$

$\boxed{IG(Li \mid D) = .91-.36 = 0.55}$

$IG(Li \mid Le) = .91-.85 = 0.06$

Movie	Type	Length	Director	Famous actors	Liked ?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	animated	long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
M7	animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

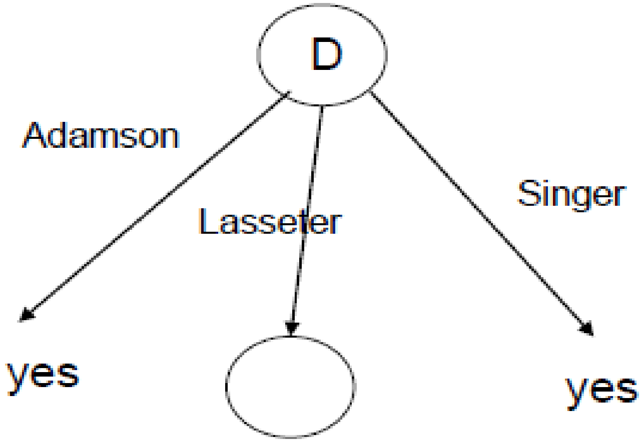
Example



Movie	Type	Length	Director	Famous actors	Liked ?
m2	Animated	Short	Lasseter	No	No
m4	animated	Long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m9	Drama	Medium	Lasseter	No	Yes

We only need to focus on the records (samples) associated with this node

Example



We eliminated the 'director' attribute. All samples have the same director

Movie	Type	Length	Famous actors	Liked ?
m2	Animated	Short	No	No
m4	animated	long	Yes	No
m5	Comedy	Long	Yes	No
m9	Drama	Medium	No	Yes

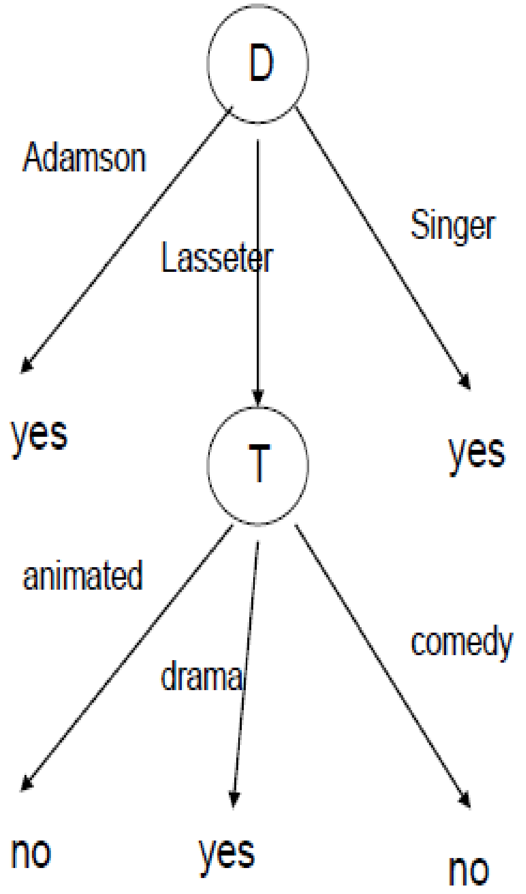
$P(Li=yes) = 1/4 \quad H(Li) = .81$

$H(Li \mid T) = 0 \quad \boxed{IG(Li \mid T) = 0.81}$

$H(Li \mid Le) = 0 \quad IG(Li \mid Le) = 0.81$

$H(Li \mid F) = 0.5 \quad IG(Li \mid F) = .31$

Example



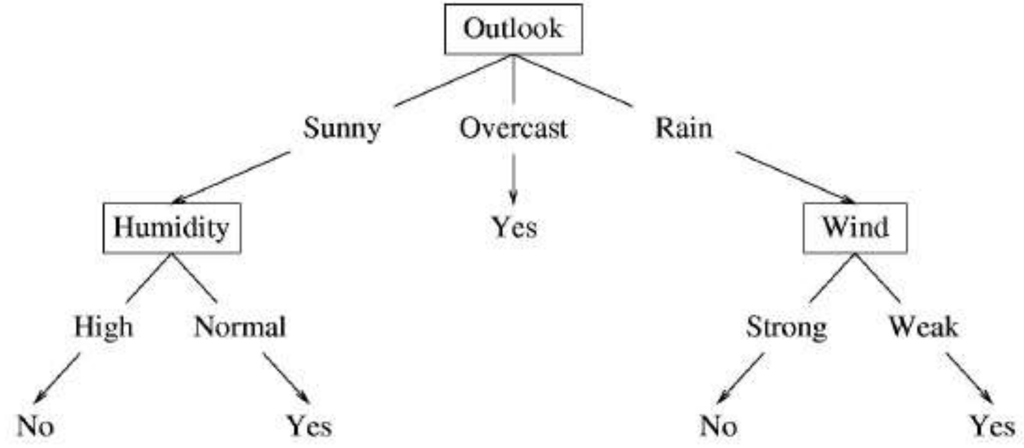
Movie	Type	Length	Famous actors	Liked ?
m2	Animated	Short	No	No
m4	animated	long	Yes	No
m5	Comedy	Long	Yes	No
m9	Drama	Medium	No	Yes

Movie	Type	Length	Director	Famous actors	Liked ?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	animated	long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
M7	animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

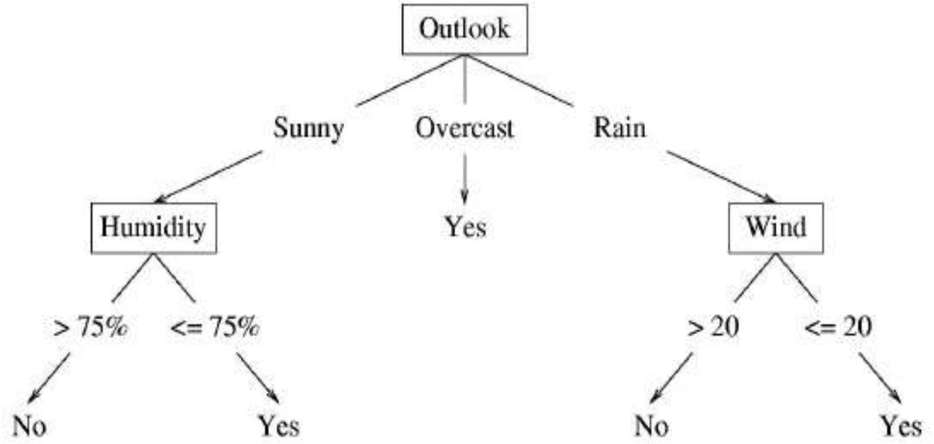
Example

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

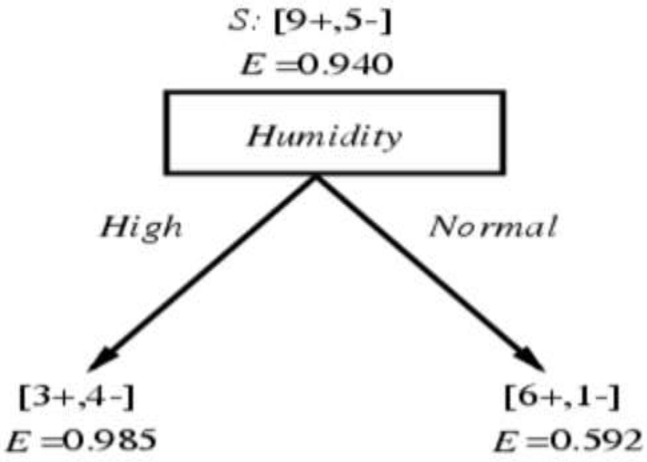
- A possible decision tree for the data:



- If features are continuous, internal nodes can test the value of a feature against a threshold

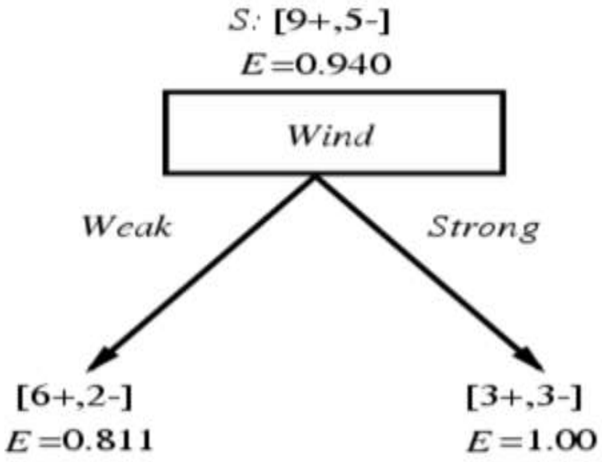


Which attribute is the best classifier?



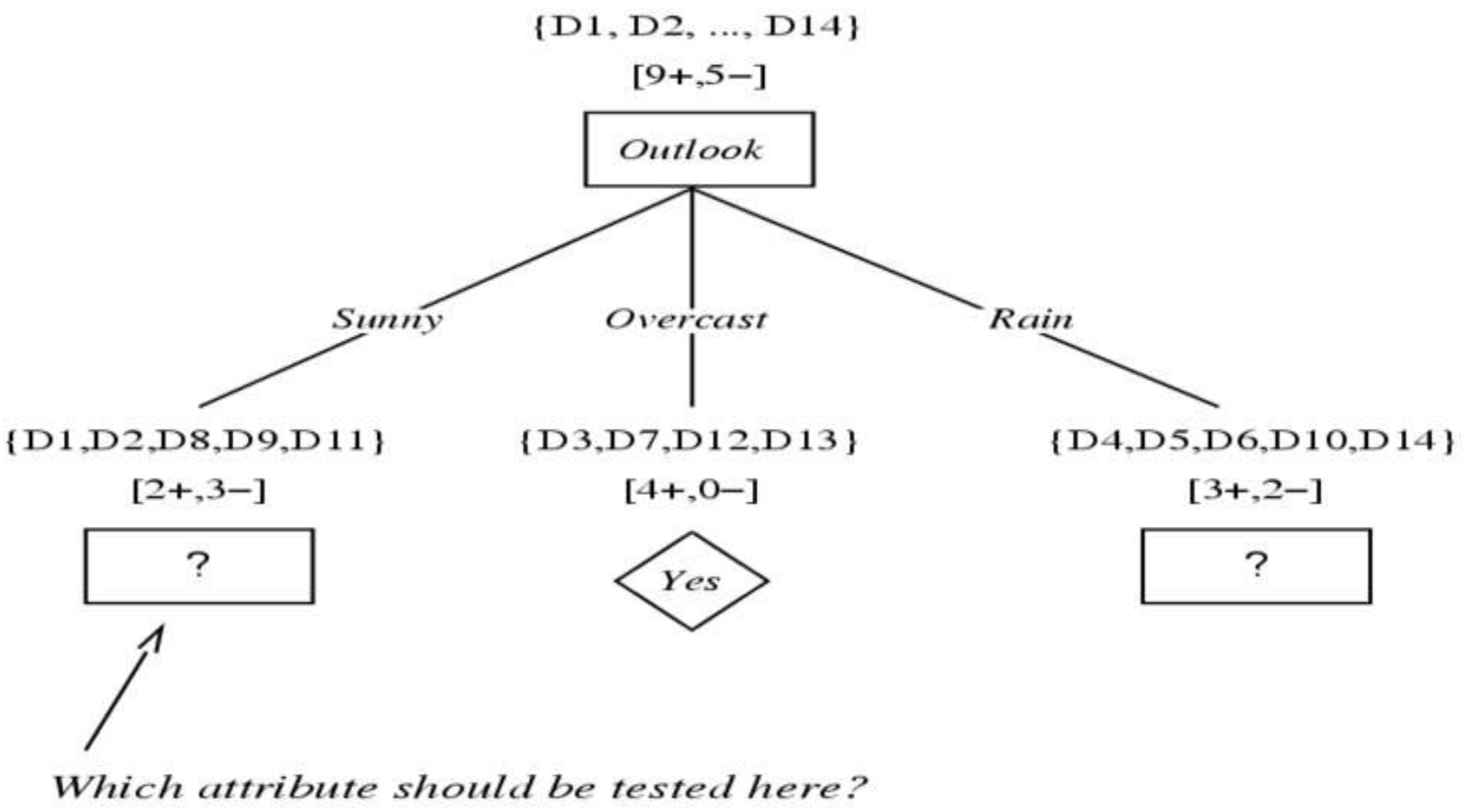
Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151



Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048



$S_{sunny} = \{D1, D2, D8, D9, D11\}$

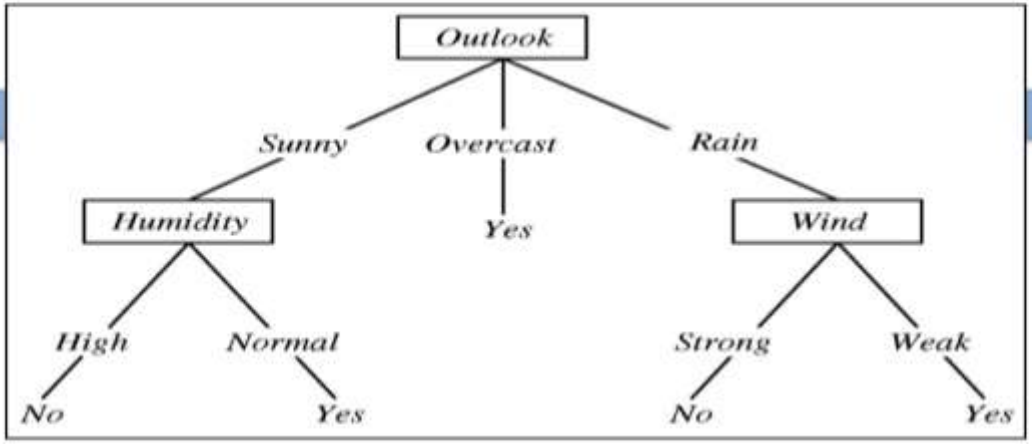
$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

Slide by Tom Mitchell

Converting a Tree to Rule



IF (*Outlook* = *Sunny*) AND (*Humidity* = *High*)
THEN *PlayTennis* = *No*

IF (*Outlook* = *Sunny*) AND (*Humidity* = *Normal*)
THEN *PlayTennis* = *Yes*

. . .

26

Hypothesis Space Search in Decision Tree Learning

Search a space of hypotheses for one that fits the training examples

- Hypothesis space of decision tree is a complete space of finite discrete-valued functions
- ID3 maintains only a single current hypothesis
- ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis

Inductive bias of ID3

- Shorter trees are preferred over larger trees
- Trees that place high information gain attributes close to the root are preferred over those that do not
- It only can achieve local optimal, but not global optimal

Computing Information-Gain for Continuous-Valued Attributes

Let attribute A be a continuous-valued attribute

- Must determine the ***best split point*** for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*

    (ai+ai+1)/2 is the midpoint between the values of ai and ai+1
  - The point with the ***minimum expected information requirement*** for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	8	High	Weak	No
D2	Sunny	10	High	Strong	No
D3	Overcast	16	High	Weak	Yes
D4	Rain	20	High	Weak	Yes
D5	Rain	6	Normal	Weak	Yes
D6	Rain	12	Normal	Strong	No
D7	Overcast	18	Normal	Strong	Yes

Temperature: 6 8 10 12 16 18 20

6 Possible Split-points: 7 9 11 14 17 19

Split-Points: 11

$A \le 11$: D1, D2, D5 (1+,2-)

$A > 11$: D3, D4, D6, D7 (3+,1-)

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{3}{7}\left(-\frac{1}{3}\log 2\left(\frac{1}{3}\right) - \frac{2}{3}\log 2\left(\frac{2}{3}\right)\right) + \frac{4}{7}\left(-\frac{3}{4}\log 2\left(\frac{3}{4}\right) - \frac{1}{4}\log 2\left(\frac{1}{4}\right)\right) = 0.8572$$

Split-Points: 14

$A \le 14$: D1, D2, D5, D6 (1+,3-)

$A > 14$: D3, D4, D7 (3+,0-)

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{4}{7}\left(-\frac{3}{4}\log 2\left(\frac{3}{4}\right) - \frac{1}{4}\log 2\left(\frac{1}{4}\right)\right) + \frac{3}{7}\left(-1 * \log 2(1) - 0 * \log 2(0)\right) = 0.4636$$

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- The attribute with the maximum gain ratio is selected as the splitting attribute

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

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

$Gain(income) = 0.029$
$Gain(student) = 0.151$
$Gain(credit\_rating) = 0.048$

Ex. $SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$

gain_ratio(income) = 0.029/1.557 = 0.019

Gini Index (CART, IBM Intelligent Miner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

GeorgiaStateUniversity

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

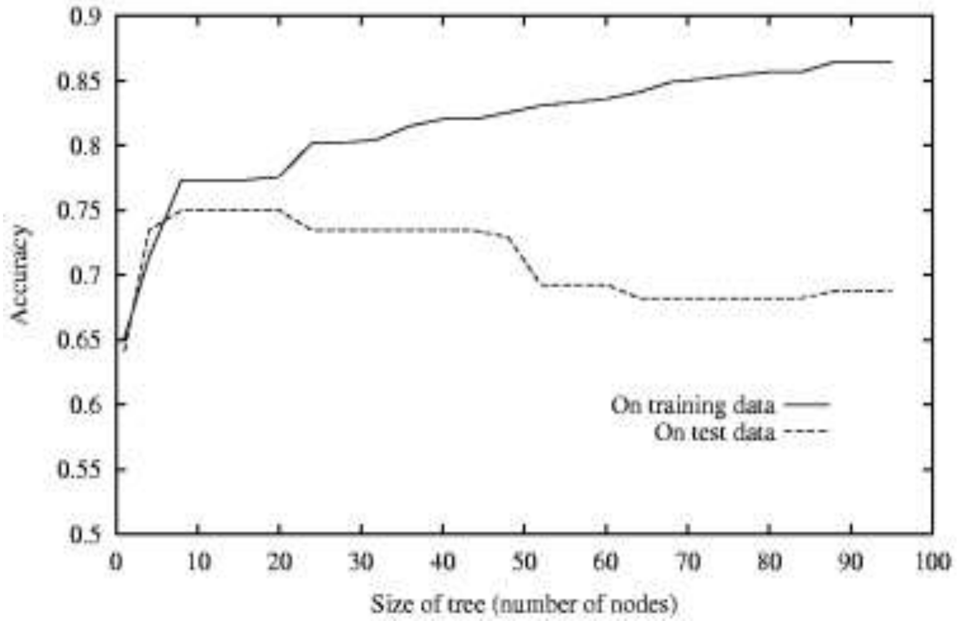
Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Avoiding Overfitting the Data

- Given a hypothesis space H, a hypothesis hεH is said to overfit the training data if there exists some alternative hypothesis h’εH, such that h has smaller error than h’ over the training examples, but h’ has a smaller error than h over the entire distribution of instance

- Training examples contain random errors or noise
  –<Outlook=Sunny, Temperature=Hot, Humidity=Normal, Wind=Strong, PlayTennis = No>?
- Some attribute happens to partition the examples perfectly, whereas the simpler h’ will not.



- Approaches to avoiding overfitting in decision tree learning
  –S: Stop growing the tree earlier
  –P: Allow the tree to overfit the data and then post-prune tree
- S is more straightforward, but it is hard to estimate precisely when to stop growing the tree
- P method has been found to be more successful in practice

Avoiding overfitting: Tree pruning

- Split data into train and test set
- Build tree using training set
  - For all internal nodes (starting at the root)
  - remove sub tree rooted at node
  - assign class to be the most common among training set
  - check test data error
  - if error is lower, keep change
  - otherwise restore subtree, repeat for all nodes in subtree

Why Prefer Short Hypotheses?

Occam's Razor

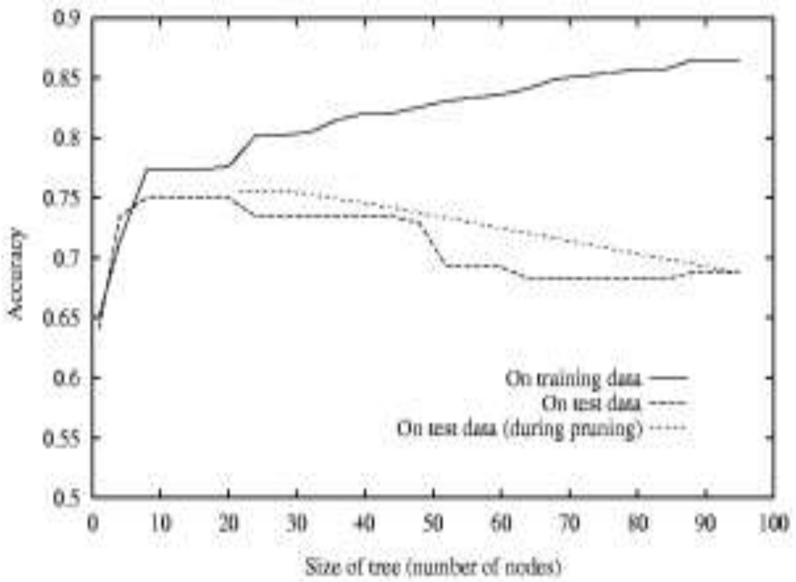
–Prefer the simplest hypothesis that fits the data

–500 nodes decision tree or 5 nodes decision tree?

–Less likely that one will find a short hypothesis that coincidentally fits the training data.

Reduced Error Pruning

- Pruning the decision node consists of removing the sub tree rooted at that node, making it a leaf node
- Node are removed only if the resulting pruned tree performs no worse than the original over the validation set.

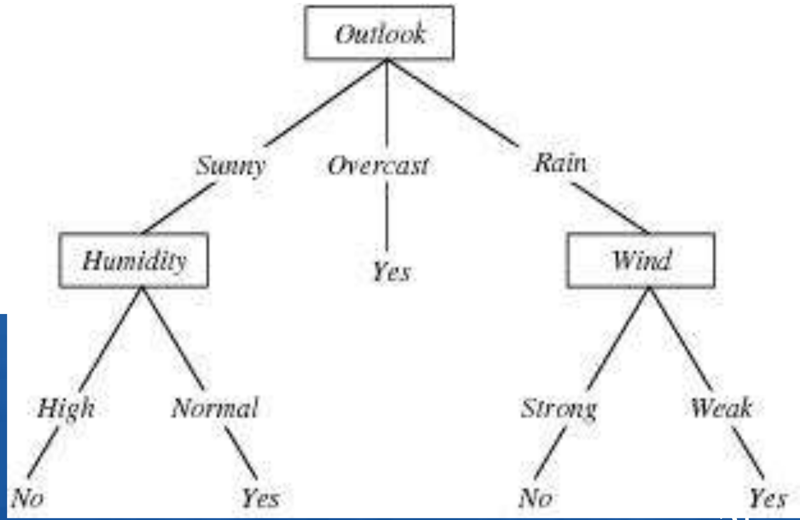


Rule Post-Pruning

- Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible
- Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node
- Prune each rule by removing any precondition that result in improving its estimated accuracy
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

if (outlook=sunny)ƛ(Humidity=High) then PlayTennis=No

•See whether remove this rule will improve the overall performance



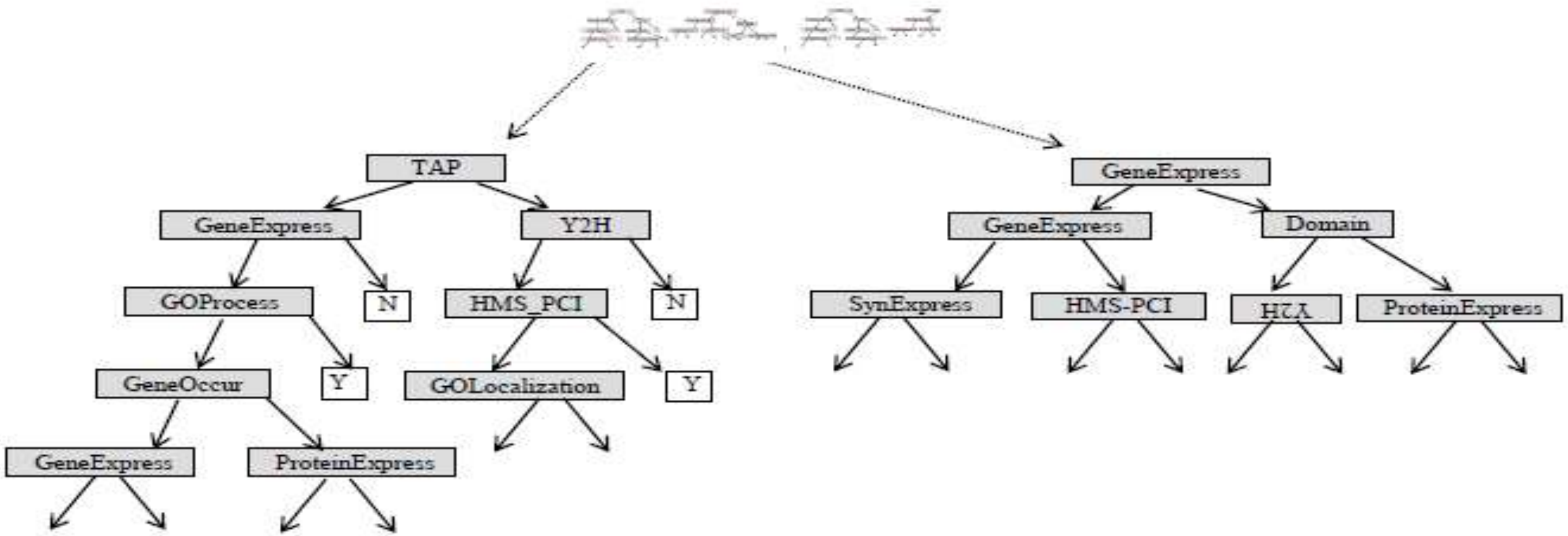
Important points

- Discriminative classifiers
- Entropy
- Information gain
- Building decision trees

- The algorithm we gave reaches homogonous nodes (or runs out of attributes)
- This is dangerous: For datasets with many (non relevant) attributes the algorithm will continue to split nodes
- This will lead to overfitting!

Random forest

- A collection of decision trees
- For each tree we select a subset of the attributes (recommended square root of |A|) and build tree using just these attributes
- An input sample is classified using majority voting



Decision Tree Classifier Building in Scikit-learn

Importing Required Libraries

Let's first load the required libraries.

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

Loading Data

Let's first load the required Pima Indian Diabetes dataset using pandas' read CSV function.

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'labe
# load dataset
pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)
```

```
pima.head()
```

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Feature Selection

Need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split the dataset by using function train_test_split(). Need to pass 3 parameters features, target, and test_set size.

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Building Decision Tree Model

Let's create a Decision Tree Model using Scikit-learn.

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating Model

Let's estimate, how accurately the classifier or model can predict the type of cultivars.
Accuracy can be computed by comparing actual test set values and predicted values.

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6753246753246753
```

Well, you got a classification rate of 67.53%, considered as good accuracy. You can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.

Visualizing Decision Trees

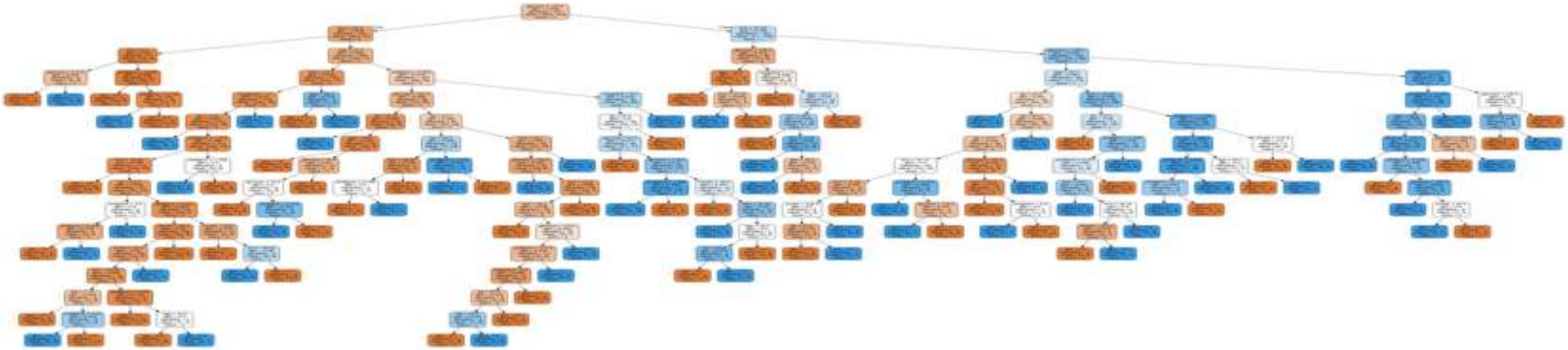
Use Scikit-learn's *export_graphviz* function for display the tree within a Jupyter notebook. For plotting tree, need to install graphviz and pydotplus.

```
pip install graphviz
```

```
pip install pydotplus
```

export_graphviz function converts decision tree classifier into dot file and pydotplus convert this dot file to png or displayable form on Jupyter.

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```



Optimizing Decision Tree Performance

In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. Plot a decision tree on the same data with max_depth=3. Other than pre-pruning parameters, attribute selection measure such as entropy can be used.

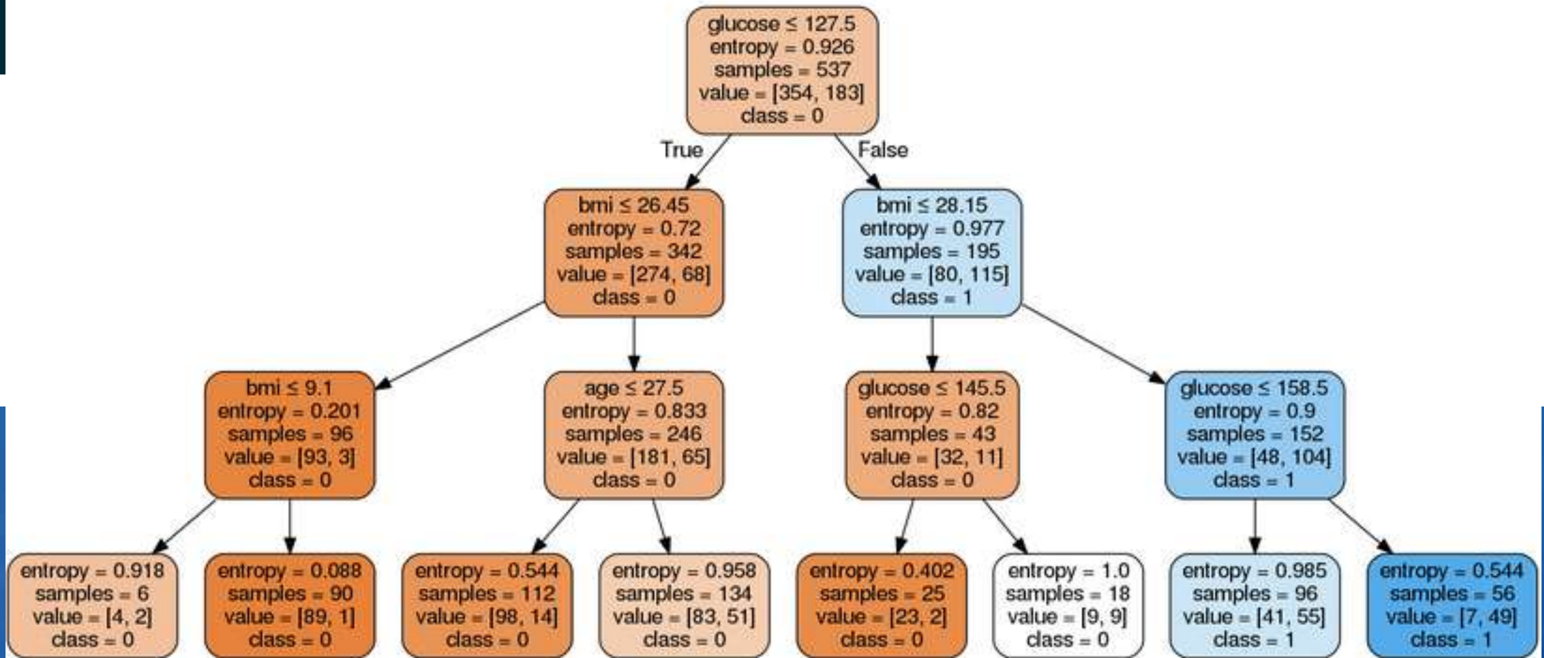
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7705627705627706
```

Well, the classification rate increased to 77.05%, which is better accuracy than the previous model.

Visualizing Decision Trees

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,class_names=['0','1
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```




Thank You!