Decision Tree

Ehab Hassan
Amr Barakat
Abdelrahman Barakat
Michael Lahzy
Moataz Al Adl

Agenda

1. Introduction.
2. Classification Methods.
    1. Information Gain.
    2. Gini Index
3. Advantages & Disadvantages
4. Applications.
5. Demo

Introduction

1. Learning.
2. Classification.
3. What is Decision Tree?
4. Simple Learning Example.

Learning

- Learning is essential for unknown environments, i.e. when designer lacks omniscience
- Learning is useful as a system construction method, i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Types of Learning

- **Supervised learning:**
  - Correct answer for each example. Answer can be a numeric variable, categorical variable etc.
  - Both inputs and outputs are given
  - The outputs are typically provided by a friendly teacher.
- **Unsupervised learning:**
  - Correct answers not given – just examples
  - The agent receives some evaluation of its actions (such as a fine for stealing bananas), but is not told the correct action (such as how to buy bananas).
- **Reinforcement learning:**
  - Occasional rewards
  - The agent can learn relationships among its percepts, and the trend with time

Classification

- **Objective**
  - To build a model of the classifying attribute based upon the other attributes
- **Classification given**
  - A set of example record, called a training set that is a set of records consists of several attributes
    - Attributes are either
    1. Continuous (i.e. Age: 23,24,25)
    2. Categorical (i.e. Gender: Female, Male)

Classification Models

1. Neural networks
2. Statistical models (linear/quadratic discriminants)
3. **Decision trees**
4. Genetic models

Decision Tree Definition

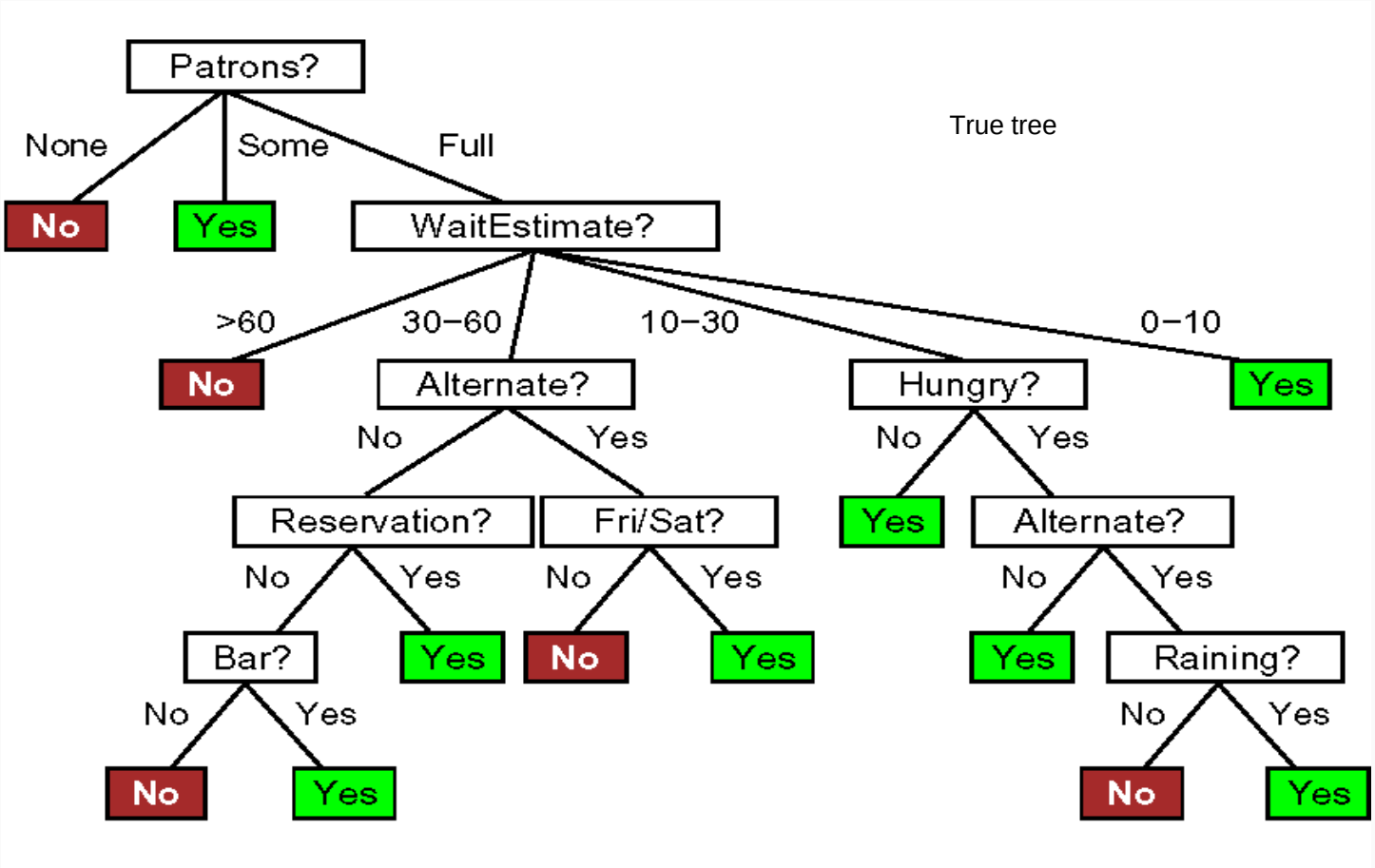
- A decision tree takes as input an object or situation described by a set of properties, and outputs a yes/no "**decision**".
- Decision tree induction is one of the simplest and yet most successful forms of machine learning. We first describe the representation—the hypothesis space—and then show how to learn a good hypothesis.
- Each node tests the value of an input attribute
- Branches from the node correspond to possible values of the attribute
- Leaf nodes supply the values to be returned if that leaf is reached

Simple Example

Decision: Whether to wait for a table at a restaurant.

1. Alternate: whether there is a suitable alternative restaurant.
2. Bar: whether the restaurant has a Bar for waiting customers.
3. Fri/Sat: true on Fridays and Saturdays Sat.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in it (None, Some, Full).
6. Price: the restaurant's rating (★, ★★ , ★★★).
7. Raining: whether it is raining outside Raining.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (Indian, Chinese, Thai, Fast food):
10. Wait Estimate: 10 mins , 10 30, 30 60, >60.

Simple Example



Training Set

	Alternate	Bar	Fri/Sat	Hungry	Patrons	Price	Raining	Res	Waiting	Type	Decision
X1	Yes	No	Yes	No	Some	Yes	No	Yes	30-60	It	Yes
X2	No	Yes	No	No	Full	No	No	No	10-30	Fr	No
X3	Yes	No	Yes	Yes	None	Yes	No	Yes	>60	Am	Yes
X4	No	Yes	No	Yes	Some	Yes	Yes	Yes	0-10	Ch	Yes
. . .											
Xn											

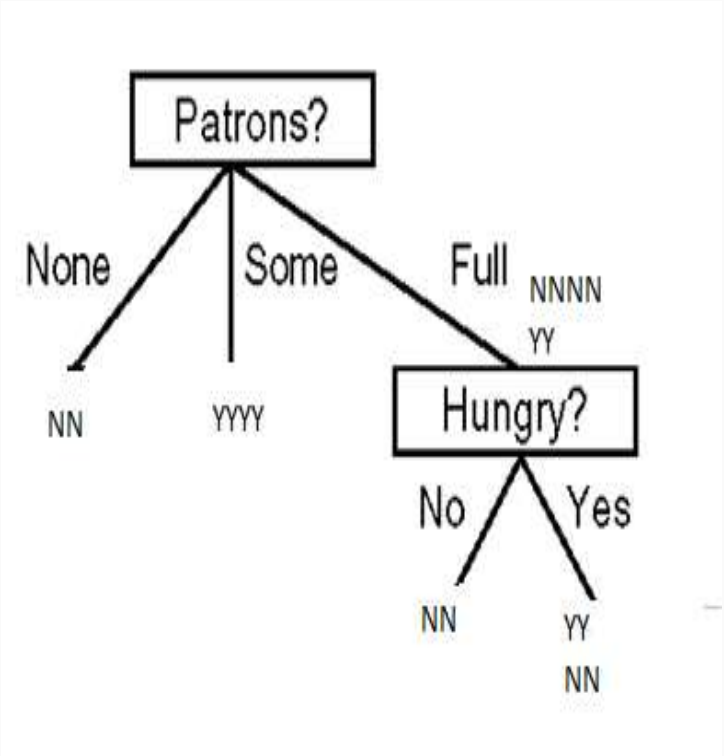
. This is our Training Set But the function is not completely specified and can be noisy

. We take this Set and produce a small and succinct tree

. Not all variable affect the decision so it will be small by utilize the don't care situations

Decision tree learning example

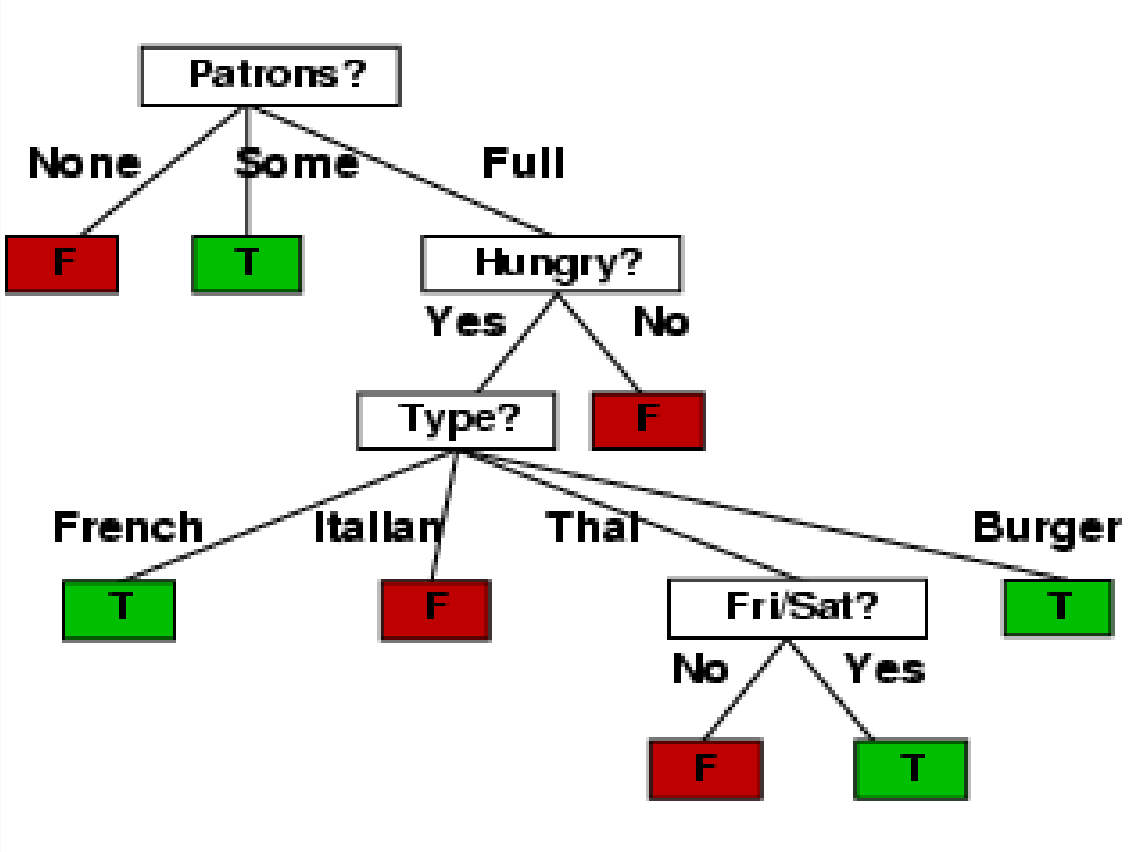
- Induced tree (from examples)
- Cannot make it more complex than what the data supports.



Decision Tree Learning Algorithm

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

The Trained Tree



Classification Methods

1. Information Gain
   - ID3
   - C4.5
   - C 5
   - J 48
2. Gini Index
   - SPRINT
   - SLIQ

Classification Methods
Information Gain

- Tree Representation.
- Attribute Selection Measure.
- Example
- Algorithms:
  - ID3
  - C4.5
  - C 5
  - J 48

Tree Represntation

To draw a decision tree from a dataset of some attributes:

- Each node corresponds to a **splitting attribute**.
- Each arc is a possible value of that attribute.
- **Splitting attribute** is selected to be the most informative among the attributes.
- **Entropy** is a factor used to measure how informative is a node.
- The algorithm uses the criterion of **information gain** to determine the goodness of a split.
- The attribute with the greatest information gain is taken as the splitting attribute, and the data set is split for all distinct values of the attribute values of the attribute.

Entropy

- A measure that characterize the purity of a dataset.
- Given a training set D with two classes, P and S of some target.

Let $p = \frac{number\ of\ P}{number\ of\ records}$,and $s = \frac{number\ of\ S}{number\ of\ records}$

$$\boldsymbol{E(D) = -p\log_2 p - s\log_2 s}$$

- For example if we have a collection D of 14 records including 9 records of **P** and 5 of **S**.

$p = \frac{9}{14} = 0.643$,and $s = \frac{5}{14} = 0.357$

$$\boldsymbol{E(D) = -(0.643)\log_2(0.643) - (0.357)\log_2(0.357) = 0.94}$$

#	Decision
1	P
2	S
3	P
4	S
5	P
6	P
7	P
8	S
9	S
10	P
11	P
12	S
13	P
14	P

Entropy

- If all records belongs to the same class.
  - Entropy = 0
- If records are equally distributed over collection class.
  - Entropy = 1
- General Formula of Entropy for a dataset of (k) classes:

$$E(D) = \sum_{i=1}^{k} -p_i \log_2 p_i$$

For example if k=4

$$E(D) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - p_3 \log_2 p_3 - p_4 \log_2 p_4$$

Information Gain

Given Entropy of a collection we can define a measure of the effectiveness of an attribute in classification.

Information Gain of an attribute is the expected reduction in entropy caused by partitioning the collection according this attribute.

$$\boldsymbol{Gain(D,V) = E(D) - \sum \frac{D_v}{D} \times E(D_v)}$$

$$Gain(D,V) = E(D) - \left(\frac{8}{14}E(X) + \frac{6}{14}E(Y)\right) = 0.94 - (0.892) = 0.048$$

#	Decision	Att. (V)
1	P	X
2	S	X
3	P	Y
4	S	Y
5	P	X
6	P	X
7	P	X
8	S	Y
9	S	X
10	P	Y
11	P	X
12	S	Y
13	P	Y
14	P	X

RAFAEL NADAL Example

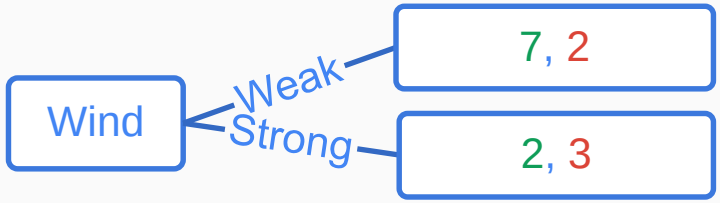
Consider that we had observed Nadal's training status for two weeks with respect to weather conditions to be able to expect if he'll **Play** in a certain day or **Stay** home.

Action	Wind	Temp	Outlook	Humidity
Play (P)	Weak	Hot	Sunny	High
Play	Strong	Hot	Sunny	High
Stay (S)	Weak	Hot	Rain	High
Play	Weak	Mid	Overcast	High
Stay	Strong	Cold	Rain	Normal
Play	Weak	Cold	Overcast	Normal
Stay	Strong	Cold	Rain	Normal
Play	Weak	Mid	Sunny	Normal
Play	Weak	Cold	Sunny	Normal
Play	Strong	Mid	Overcast	Normal
Stay	Weak	Mid	Sunny	High
Stay	Strong	Mid	Rain	High
Play	Weak	Hot	Overcast	Normal
Play	Weak	Cold	Rain	High

Nadal training statistics with weather conditions

Will he play if:

Action	Wind	Temp	Outlook	Humidity
?	Weak	Hot	Sunny	Normal



$$E(Weak) = -\left(\frac{7}{9}\right)\log_2\left(\frac{7}{9}\right) - \left(\frac{2}{9}\right)\log_2\left(\frac{2}{9}\right) = 0.764$$

$$E(Strong) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.970$$

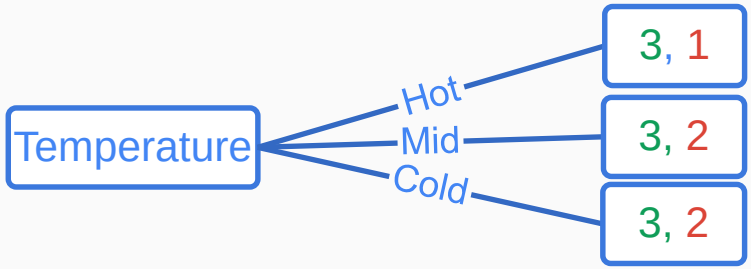
$$Gain(D, Wind) = E(D) - \left(\frac{9}{14}E(Weak) + \frac{5}{14}E(Strong)\right)$$

$$= 0.94 - (0.491 + 0.346) = 0.94 - 0.837 = 0.102$$

Action	Wind	Temp	Outlook	Humidity
Play (P)	Weak	Hot	Sunny	High
Play	Strong	Hot	Sunny	High
Stay (S)	Weak	Hot	Rain	High
Play	Weak	Mid	Overcast	High
Stay	Strong	Cold	Rain	Normal
Play	Weak	Cold	Overcast	Normal
Stay	Strong	Cold	Rain	Normal
Play	Weak	Mid	Sunny	Normal
Play	Weak	Cold	Sunny	Normal
Play	Strong	Mid	Overcast	Normal
Stay	Weak	Mid	Sunny	High
Stay	Strong	Mid	Rain	High
Play	Weak	Hot	Overcast	Normal
Play	Weak	Cold	Rain	High

Nadal training statistics with weather conditions

Action	Wind	Temp	Outlook	Humidity
Gain	0.102			



$$E(Hot) = -\left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) = 0.811$$

$$E(Mid) = E(Cold) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.970$$

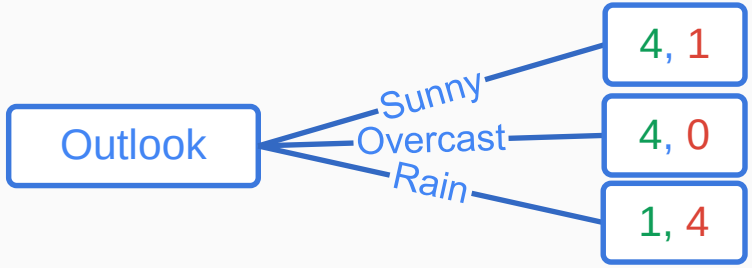
$$Gain(D, Temp) = E(D) - \left(\frac{4}{14}E(H) + \frac{5}{14}E(M) + \frac{5}{14}E(C)\right)$$

$$= 0.94 - (0.231 + 0.346 + 0.346) = 0.94 - 0.932 = 0.008$$

Action	Wind	Temp	Outlook	Humidity
Play (P)	Weak	Hot	Sunny	High
Play	Strong	Hot	Sunny	High
Stay (S)	Weak	Hot	Rain	High
Play	Weak	Mid	Overcast	High
Stay	Strong	Cold	Rain	Normal
Play	Weak	Cold	Overcast	Normal
Stay	Strong	Cold	Rain	Normal
Play	Weak	Mid	Sunny	Normal
Play	Weak	Cold	Sunny	Normal
Play	Strong	Mid	Overcast	Normal
Stay	Weak	Mid	Sunny	High
Stay	Strong	Mid	Rain	High
Play	Weak	Hot	Overcast	Normal
Play	Weak	Cold	Rain	High

Nadal training statistics with weather conditions

Action	Wind	Temp	Outlook	Humidity
Gain	0.102	0.008		



$$E(Sunny) = 0.722$$

$$E(Overcast) = 0$$

$$E(Rain) = 0.722$$

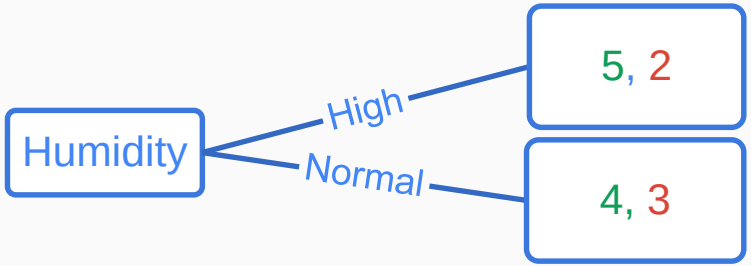
$$Gain(D, Outlook) = E(D) - \left(\frac{5}{14}E(S) + \frac{4}{14}E(O) + \frac{5}{14}E(R)\right)$$

$$= 0.94 - (0.258 + 0 + 0.258) = 0.94 - 0.516 = 0.424$$

Action	Wind	Temp	Outlook	Humidity
Play (P)	Weak	Hot	Sunny	High
Play	Strong	Hot	Sunny	High
Stay (S)	Weak	Hot	Rain	High
Play	Weak	Mid	Overcast	High
Stay	Strong	Cold	Rain	Normal
Play	Weak	Cold	Overcast	Normal
Stay	Strong	Cold	Rain	Normal
Play	Weak	Mid	Sunny	Normal
Play	Weak	Cold	Sunny	Normal
Play	Strong	Mid	Overcast	Normal
Stay	Weak	Mid	Sunny	High
Stay	Strong	Mid	Rain	High
Play	Weak	Hot	Overcast	Normal
Play	Weak	Cold	Rain	High

Nadal training statistics with weather conditions

Action	Wind	Temp	Outlook	Humidity
Gain	0.102	0.008	0.424	



$$E(High) = 0.863$$

$$E(Normal) = 0.985$$

$$Gain(D, Humidity) = E(D) - \left(\frac{7}{14}E(H) + \frac{7}{14}E(N)\right)$$

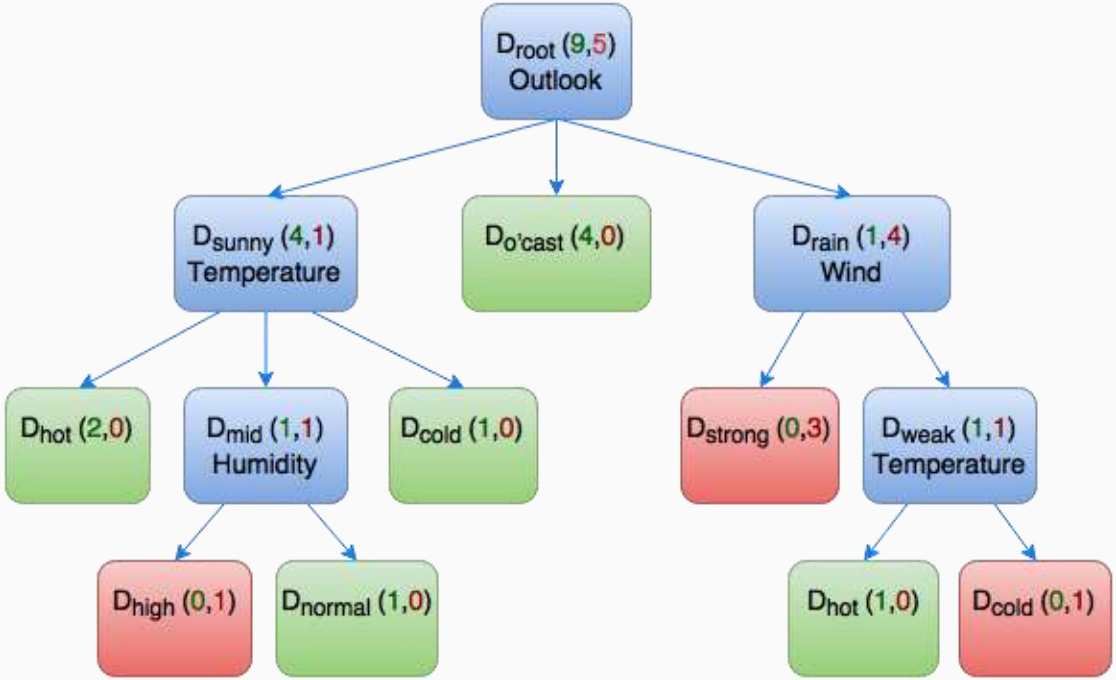
$$= 0.94 - (0.431 + 0.493) = 0.94 - 0.924 = 0.016$$

Action	Wind	Temp	Outlook	Humidity
Play (P)	Weak	Hot	Sunny	High
Play	Strong	Hot	Sunny	High
Stay (S)	Weak	Hot	Rain	High
Play	Weak	Mid	Overcast	High
Stay	Strong	Cold	Rain	Normal
Play	Weak	Cold	Overcast	Normal
Stay	Strong	Cold	Rain	Normal
Play	Weak	Mid	Sunny	Normal
Play	Weak	Cold	Sunny	Normal
Play	Strong	Mid	Overcast	Normal
Stay	Weak	Mid	Sunny	High
Stay	Strong	Mid	Rain	High
Play	Weak	Hot	Overcast	Normal
Play	Weak	Cold	Rain	High

Nadal training statistics with weather conditions

Information gains at root node are:

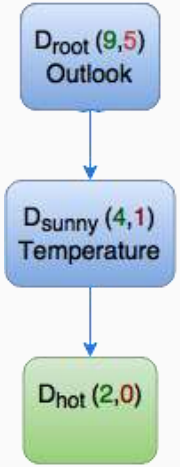
Action	Wind	Temp	Outlook	Humidity
Gain	0.102	0.008	**0.424**	0.016

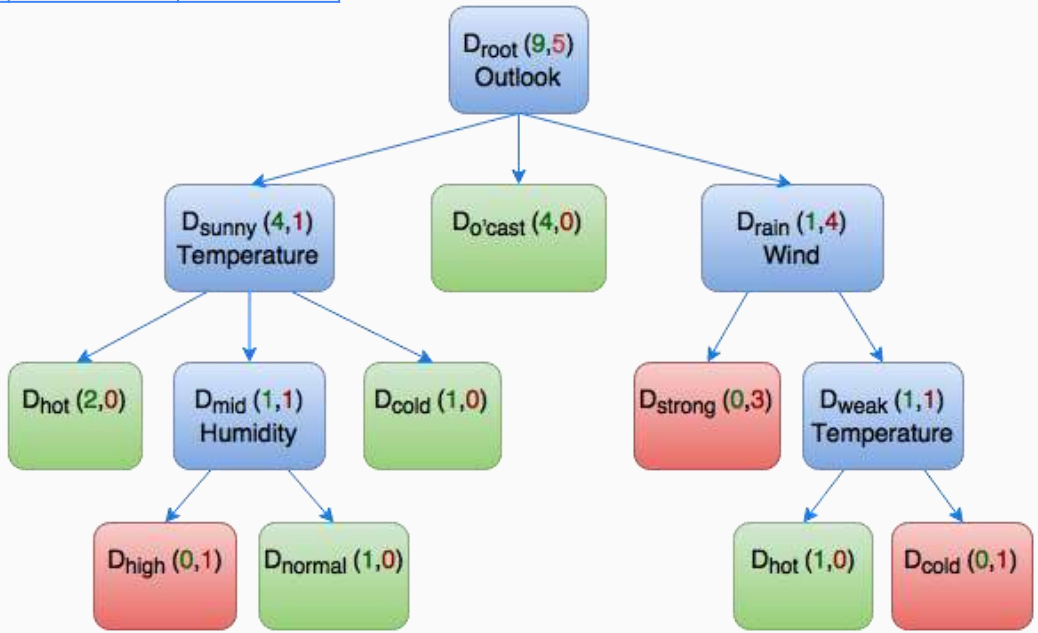


Decision for Test dataset

Will he play if:

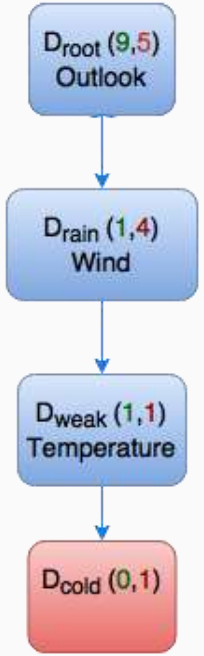
Action	Wind	Temp	Outlook	Humidity
?	Weak	Hot	Sunny	Normal





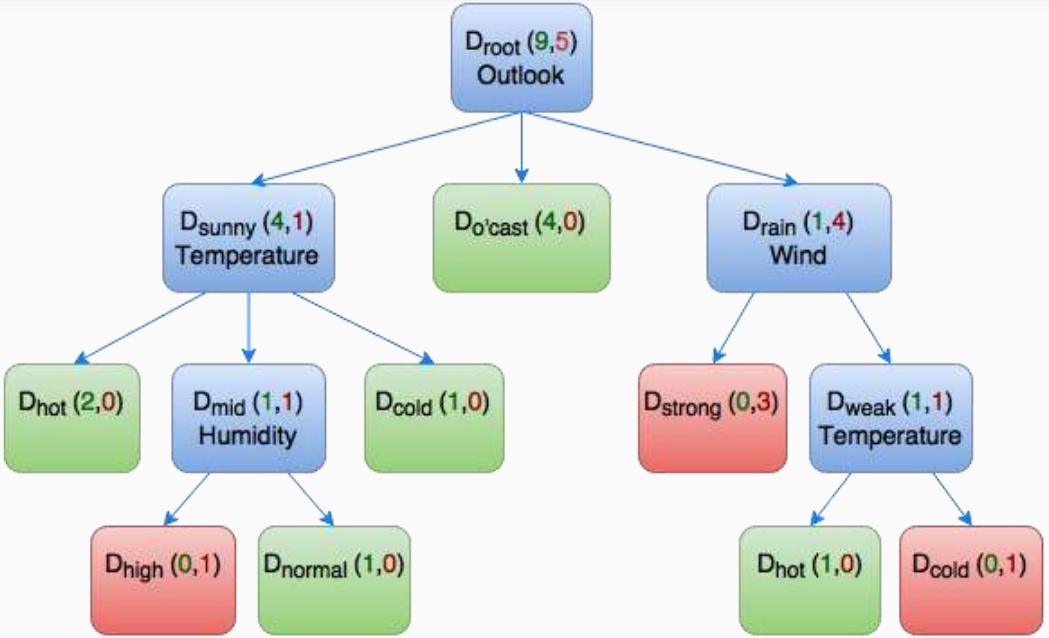
Will he play if:

Action	Wind	Temp	Outlook	Humidity
?	Weak	Hot	Rain	Normal



Rules Extraction

- We have five leaf nodes., In a decision tree, each leaf node represents a rule.
- We have the following rules corresponding to the tree given in figure.



1. **if** it is *SUNNY* and Temperature is *HOT* **then** PLAY.
2. **if** it is *SUNNY* and Temperature is *COLD* **then** STAY.
3. **if** it is *SUNNY* and Temperature is *MID* and Humidity is *NORMAL* **then** PLAY.
4. **if** it is *OVERCAST* **then** PLAY.
5. **if** it is *RAIN* and Wind is *WEAK* and Temperature is *HOT* **then** PLAY.

ID3 Algorithm

ID algorithm uses Information Gain methodology to create a tree:

- This decision tree is used to classify new unseen test cases by working down the decision tree using the values of this test case to arrive at a terminal node that tells you what class this test case belongs to.

ID3 Pseudo Code

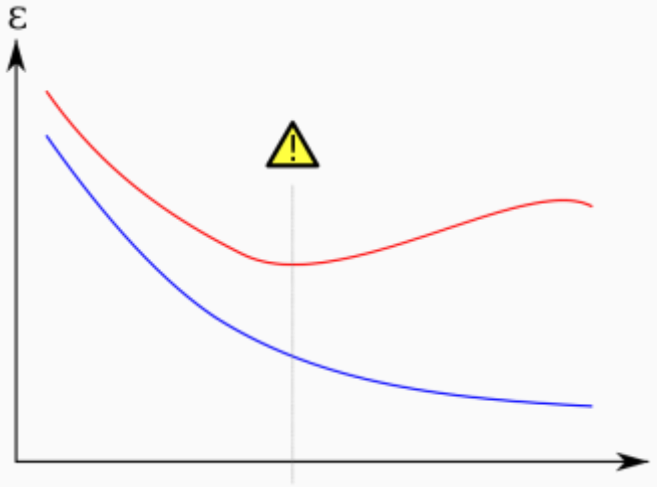
```
ID3 (Dataset, Target_Attribute, Attributes)
Create a root node for the tree
If all Dataset are positive,
        Return the single-node tree Root, with label = +.
If all Dataset are negative,
         Return the single-node tree Root, with label = -.
If number of predicting attributes is empty,
         Return the single node tree Root, with label = most common value of the
target attribute.
Else Begin
        A ← The Attribute that best classifies Dataset.
        Decision Tree attribute for Root = A.
        Foreach possible value, v_i, of A,
        Add a new tree branch below Root, corresponding to the test A = v_i.
        Let Dataset(v_i) be the subset of Dataset that have the value v_i for A
                If below this new branch add the subtree
                        ID3 (Dataset(v_i), Target_Attribute, Attributes – {A})
End
Return Root
```

Shortage of ID3

ID3 suffers from the problem of **over-fitting** like many other techniques, so we limit the level of the decision tree and convert the nodes on the last levels to leaf nodes, if they aren't already based on simple heuristics for assigning the class.

Overfitting

- Overfitting is a common problem in machine learning.
- Training error is shown in blue, test error in red, both as a function of the number of tree nodes.
- If the validation error increases while the training error steadily decreases then a situation of **overfitting** occurred.
- The best predictive and fitted model would be where the validation error has its global minimum.



C4.5 Algorithm

C4.5 algorithm acts similar to ID3 but improves a few of ID3 behaviors:.

- A possibility to use continuous data.
- Using unknown (missing) values.
- Ability to use attributes with different weights.
- Pruning the tree after being created.

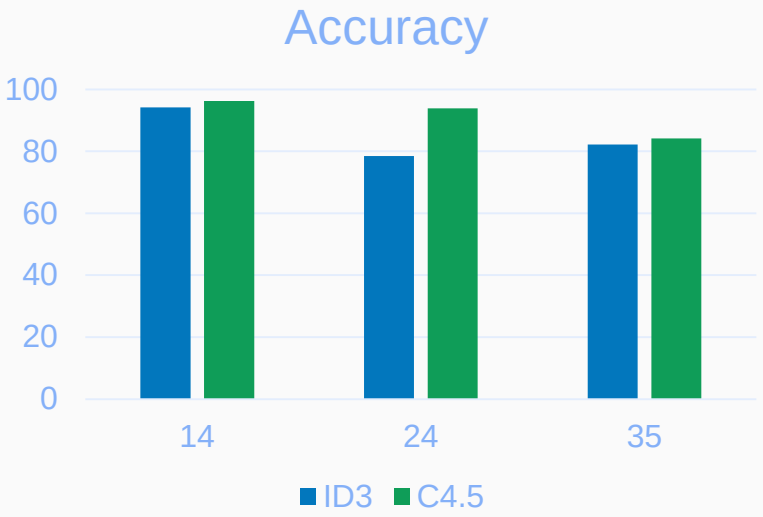
C4.5 Pseudo Code

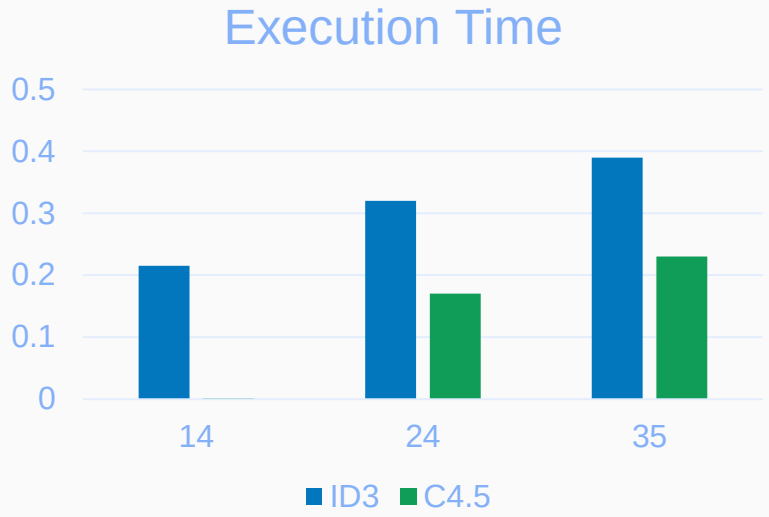
```
Check for base cases
For each attribute a
        Find the normalized information gain ratio from splitting on a
Let a_best be the attribute with the highest normalized information gain
Create a decision node that splits on a_best
Recur on the sublists obtained by splitting on a_best, and add those nodes as
children of node
```

Improvement in C4.5 from ID3

Right table presents a comparison of ID3 and C4.5 accuracy and time consuming with different data set size, this comparison is presented graphically in below figures

Dataset Size	Accuracy		Time	
	ID3	C4.5	ID3	C4.5
14	94.15	96.2	0.215	0.0015
24	78.47	83.85	0.32	0.17
35	82.2	84.12	0.39	0.23





C5 Algorithm

- C5 algorithm follows the rules of algorithm of C4.5.
- C5 algorithm has many features like:
    - The large decision tree can be viewing as a set of rules which is easy to understand.
    - Gives the acknowledge on noise and missing data.
    - Problem of over fitting and error pruning is solved.
    - In classification technique the C5 classifier can predict which attributes are relevant and which are not relevant in classification.

Improvement in C5 from C4.5

- **Speed:** C5.0 is significantly faster than C4.5 (several orders of magnitude)
- **Memory usage:** C5.0 is more memory efficient than C4.5
- **Smaller decision trees:** C5.0 gets similar results to C4.5 with considerably smaller decision trees.
- **Weighting:** C5.0 allows you to weight different cases and misclassification types.
- **Winnowing:** a C5.0 option automatically winnows the attributes to remove those that may be unhelpful.

J48 Algorithm

- J48 is an open source Java implementation of the C4.5 algorithm in the WEKA data mining

Classification Methods
Gini Index

- Used in CART, SLIQ, SPRINT to determine the best split.
- **Tree Growing Process:**
    - Find each predictor’s best split.
    - Find the node’s best split : Among the best splits found in step 1, choose the one that maximizes the splitting criterion.
    - Split the node using its best split found in step 2 if the stopping rules are not satisfied.

Finding the Best Split: GINI Index

- **GINI Index for a given node t:**

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

p(j|t) is the relative frequency of j at node t

- **Maximum (1-1/n):** records equally distributed in n classes
- **Minimum 0:** all records in one class

Computing GINI Index

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0, **P(C2)** = 6/6=1

GINI = 1 - $P(C1)^2$ - $P(C2)^2$ = 1 - 0 - 1 = 0

C1	**2**
C2	**4**

P(C1) = 2/6 = 1/3, **P(C2)** = 4/6=2/3

GINI = 1 - $(1/3)^2$ - $(2/3)^2$ = 0.444

C1	**3**
C2	**3**

P(C1) = 3/6 = 1/2, **P(C2)** = 3/6 = 1/2

GINI = 1 - $(1/2)^2$ - $(1/2)^2$ = 1 - 1/4 - 1/4 = 0.5

Finding the Best Split: GINI Index

When a node p is split into k partitions (children), the quality of split is computed as:

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

ni = number of records at child i,
n = number of records at node p

Computing GINI Index for Binary Attributes

- Splits into two partitions
- Effect of Weighing partitions: Larger and Purer Partitions are sought for.
- Example (wind condition in nadal example):
  - **GINI(Weak)** = $1 - (7/9)^2 - (2/9)^2$ = **0.346**
  - **GINI(Strong)** = $1 - (2/5)^2 - (3/5)^2$ = **0.48**
  - **GINI(Children)** = $9/14 * 0.346 + 5/14 * 0.48$ = **0.393**

Wind?	Weak	Strong
Play	7	2
Stay	2	3
Gini	0.346	0.48
Gini(Children) = 0.393		

Computing GINI Index for Categorical Attributes

- **Multiway Split**
  - For each distinct value, gather counts for each class in the dataset
- Example (temperature condition in nadal example):
  - **GINI(Hot)** $=1 - (3/4)^2 - (1/4)^2 =$ **0.375**
  - **GINI(Mid)** $= 1 - (3/5)^2 - (2/5)^2 =$ **0.48**
  - **GINI(Cold)** $= 1 - (3/5)^2 - (2/5)^2 =$ **0.48**
  - **GINI(Children)** $= 4/14 * 0.375 + 5/14 * 0.48 + 5/14 * 0.48 =$ **0.45**

Temp.?	Hot	Med	Cold
Play	3	3	3
Stay	1	2	2
Gini	0.375	0.48	0.48
Gini(Children) = 0.45			

Computing GINI Index for Categorical Attributes

- **Two way Split**
  - Join 2 or more nodes so we have only 2 nodes
- Example (outlook condition in nadal example):

Temp.?	Hot	Med/Cold
Play	3	6
Stay	1	4
Gini	0.375	0.48
Gini(Children) = 0.45		

Temp.?	Hot/Med	Cold
Play	6	3
Stay	3	2
Gini	0.444	0.48
Gini(Children) = 0.457		

Temp.?	Hot/Cold	Med
Play	6	3
Stay	3	2
Gini	0.444	0.48
Gini(Children) = 0.457		

Two way split will result in a larger gini index because it merge purer subsets resulting in more impure new set

Computing GINI Index for Continuous Attributes

- Assume we have a continuous attribute **weight** which ranges from 60-100 kg:
  - It would be computationally expensive to go through the data set scanning to see what values are below **w** for each value of **w**
  - Instead of brute force approach, we sort the data based on the weight attribute
  - Then choose a candidate split position that are identified by taking the midpoints between two adjacent sorted values
  - And employ a split approach using these values as we did in categorical attributes

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	Strong	Hot	Sunny	High	Play
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	Weak	Mid	Overcast	High	Play
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	Strong	Mid	Rain	High	Stay
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Wind

GINI(Weak) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (7/9)^2 - (2/9)^2$

$= \mathbf{0.346}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	**Weak**	**Hot**	**Sunny**	**High**	**Play (P)**
Day 2	Strong	Hot	Sunny	High	Play
Day 3	**Weak**	**Hot**	**Rain**	**High**	**Stay (S)**
Day 4	**Weak**	**Mid**	**Overcast**	**High**	**Play**
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	**Weak**	**Cold**	**Overcast**	**Normal**	**Play**
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	**Weak**	**Mid**	**Sunny**	**Normal**	**Play**
Day 9	**Weak**	**Cold**	**Sunny**	**Normal**	**Play**
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	**Weak**	**Mid**	**Sunny**	**High**	**Stay**
Day 12	Strong	Mid	Rain	High	Stay
Day 13	**Weak**	**Hot**	**Overcast**	**Normal**	**Play**
Day 14	**Weak**	**Cold**	**Rain**	**High**	**Play**

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Wind

GINI(Weak) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (7/9)^2 - (2/9)^2$

$=$ **0.346**

GINI(Strong) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (2/5)^2 - (3/5)^2$

$=$ **0.480**

GINI-Split(Wind) $= 0.346 * (9/14) + 0.48 * (5/14) =$ **0.394**

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	**Strong**	**Hot**	**Sunny**	**High**	**Play**
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	Weak	Mid	Overcast	High	Play
Day 5	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	**Strong**	**Mid**	**Overcast**	**Normal**	**Play**
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	**Strong**	**Mid**	**Rain**	**High**	**Stay**
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Temp

GINI(Hot) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (3/4)^2 - (1/4)^2$

$= \mathbf{0.375}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	**Weak**	**Hot**	**Sunny**	**High**	**Play (P)**
Day 2	**Strong**	**Hot**	**Sunny**	**High**	**Play**
Day 3	**Weak**	**Hot**	**Rain**	**High**	**Stay (S)**
Day 4	Weak	Mid	Overcast	High	Play
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	Strong	Mid	Rain	High	Stay
Day 13	**Weak**	**Hot**	**Overcast**	**Normal**	**Play**
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Temp

GINI(Hot) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (3/4)^2 - (1/4)^2$

$= \mathbf{0.375}$

GINI(Mid) $= 1 - (3/5)^2 - (2/5)^2$

$= \mathbf{0.48}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	Strong	Hot	Sunny	High	Play
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	**Weak**	**Mid**	**Overcast**	**High**	**Play**
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	**Weak**	**Mid**	**Sunny**	**Normal**	**Play**
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	**Strong**	**Mid**	**Overcast**	**Normal**	**Play**
Day 11	**Weak**	**Mid**	**Sunny**	**High**	**Stay**
Day 12	**Strong**	**Mid**	**Rain**	**High**	**Stay**
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Temp

GINI(Hot) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (3/4)^2 - (1/4)^2$

$=$ **0.375**

GINI(Mid) $= 1 - (3/5)^2 - (2/5)^2$

$=$ **0.48**

GINI(Cold) $= 1 - (3/5)^2 - (2/5)^2$

$=$ **0.48**

GINI-Split(Temp) $= 0.375 * (4/14) + 0.48 * (5/14) + 0.48 * (5/14) =$ **0.450**

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	Strong	Hot	Sunny	High	Play
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	Weak	Mid	Overcast	High	Play
Day 5	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 6	**Weak**	**Cold**	**Overcast**	**Normal**	**Play**
Day 7	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	**Weak**	**Cold**	**Sunny**	**Normal**	**Play**
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	Strong	Mid	Rain	High	Stay
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	**Weak**	**Cold**	**Rain**	**High**	**Play**

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Outlook

GINI(Sunny) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (4/5)^2 - (1/5)^2$

$= \mathbf{0.32}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	**Weak**	**Hot**	**Sunny**	**High**	**Play (P)**
Day 2	**Strong**	**Hot**	**Sunny**	**High**	**Play**
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	Weak	Mid	Overcast	High	Play
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	**Weak**	**Mid**	**Sunny**	**Normal**	**Play**
Day 9	**Weak**	**Cold**	**Sunny**	**Normal**	**Play**
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	**Weak**	**Mid**	**Sunny**	**High**	**Stay**
Day 12	Strong	Mid	Rain	High	Stay
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Outlook

GINI(Sunny) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (4/5)^2 - (1/5)^2$

$= \mathbf{0.32}$

GINI(Rain) $= 1 - (1/5)^2 - (4/5)^2$

$= \mathbf{0.32}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	Strong	Hot	Sunny	High	Play
Day 3	**Weak**	**Hot**	**Rain**	**High**	**Stay (S)**
Day 4	Weak	Mid	Overcast	High	Play
Day 5	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	**Strong**	**Mid**	**Rain**	**High**	**Stay**
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	**Weak**	**Cold**	**Rain**	**High**	**Play**

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Outlook

GINI(Sunny) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (4/5)^2 - (1/5)^2$

$= \mathbf{0.32}$

GINI(Rain) $= 1 - (1/5)^2 - (4/5)^2$

$= \mathbf{0.32}$

GINI(Over..) $= 1 - (4/4)^2 - (0/4)^2$

$= \mathbf{0}$

GINI-Split(O'look) $= 0.320 * (5/14) + 0.32 * (5/14) + 0 *(4/14) = \mathbf{0.229}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	Strong	Hot	Sunny	High	Play
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	**Weak**	**Mid**	**Overcast**	**High**	**Play**
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	**Weak**	**Cold**	**Overcast**	**Normal**	**Play**
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	**Strong**	**Mid**	**Overcast**	**Normal**	**Play**
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	Strong	Mid	Rain	High	Stay
Day 13	**Weak**	**Hot**	**Overcast**	**Normal**	**Play**
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Humidity

GINI(High) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (4/7)^2 - (3/7)^2$

$= \mathbf{0.49}$

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	**Weak**	**Hot**	**Sunny**	**High**	**Play (P)**
Day 2	**Strong**	**Hot**	**Sunny**	**High**	**Play**
Day 3	**Weak**	**Hot**	**Rain**	**High**	**Stay (S)**
Day 4	**Weak**	**Mid**	**Overcast**	**High**	**Play**
Day 5	Strong	Cold	Rain	Normal	Stay
Day 6	Weak	Cold	Overcast	Normal	Play
Day 7	Strong	Cold	Rain	Normal	Stay
Day 8	Weak	Mid	Sunny	Normal	Play
Day 9	Weak	Cold	Sunny	Normal	Play
Day 10	Strong	Mid	Overcast	Normal	Play
Day 11	**Weak**	**Mid**	**Sunny**	**High**	**Stay**
Day 12	**Strong**	**Mid**	**Rain**	**High**	**Stay**
Day 13	Weak	Hot	Overcast	Normal	Play
Day 14	**Weak**	**Cold**	**Rain**	**High**	**Play**

Nadal training statistics (GINI Index)

Step 1: Find the Gini for each predictor:

Gini Index For Humidity

GINI(High) $= 1 - P(play)^2 - P(stay)^2$

$= 1 - (4/7)^2 - (3/7)^2$

$=$ **0.49**

GINI(Norm.) $= 1 - (5/7)^2 - (2/7)^2$

$=$ **0.408**

GINI-Split(Hum.) $= 0.49 * (7/14) + 0.408 * (7/14) =$ **0.449**

Day	Wind	Temp	Outlook	Humidity	Action
Day 1	Weak	Hot	Sunny	High	Play (P)
Day 2	Strong	Hot	Sunny	High	Play
Day 3	Weak	Hot	Rain	High	Stay (S)
Day 4	Weak	Mid	Overcast	High	Play
Day 5	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 6	**Weak**	**Cold**	**Overcast**	**Normal**	**Play**
Day 7	**Strong**	**Cold**	**Rain**	**Normal**	**Stay**
Day 8	**Weak**	**Mid**	**Sunny**	**Normal**	**Play**
Day 9	**Weak**	**Cold**	**Sunny**	**Normal**	**Play**
Day 10	**Strong**	**Mid**	**Overcast**	**Normal**	**Play**
Day 11	Weak	Mid	Sunny	High	Stay
Day 12	Strong	Mid	Rain	High	Stay
Day 13	**Weak**	**Hot**	**Overcast**	**Normal**	**Play**
Day 14	Weak	Cold	Rain	High	Play

Nadal training statistics (GINI Index)

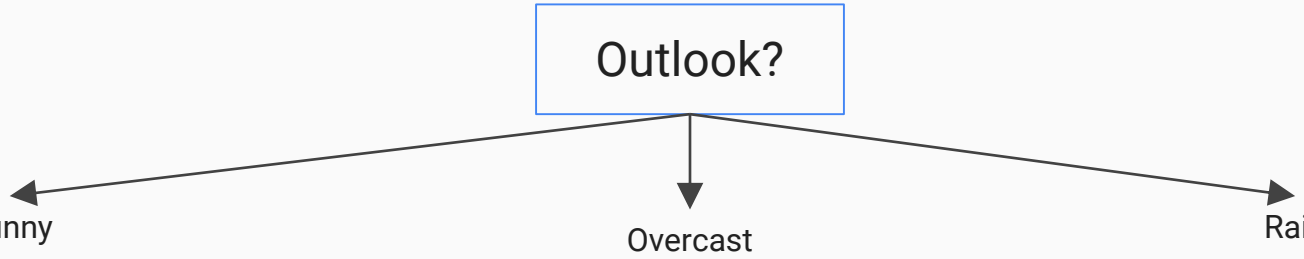
Step 2: Find the node's best split:

	Wind	Temp	Outlook	Humidity
Gini Split	0.394	0.450	**0.229**	0.449

The best split will use the outlook attribute

Nadal training statistics (GINI Index)

Step 3: Split the node



Day	Wnd	Tmp	Hu.	
1	W	H	H	P
2	S	H	H	P
8	W	M	N	P
9	W	C	N	P
11	W	M	H	S

Impure Set repeat from Step 1

Day	Wnd	Tmp	Hu.	
4	W	M	H	P
6	W	C	N	P
10	S	M	N	P
13	W	H	N	P

No further splitting (Pure set)

Day	Wnd	Tmp	Hu.	
3	W	H	H	S
5	S	C	N	S
7	S	C	N	S
12	S	M	H	S
14	W	C	H	P

Impure Set repeat from Step 1

Scaling Classification

- SLIQ
- SPRINT

Outline

Why do we need scaling?

Cover state of the art methods

Details on my research (which is one of the state of the art methods)

Problems Scaling Decision Trees

- Data doesn't fit in RAM
- Numeric attributes require repeated sorting
- Noisy datasets lead to very large trees
- Large datasets fundamentally different from smaller ones
  - Can't store the entire dataset
  - Underlying phenomenon changes over time

Current State-Of-The-Art

1. Disk based methods
   - Sprint
   - SLIQ
2. Sampling methods
   - BOAT
   - VFDT & CVFDT
3. Data Stream Methods
   - VFDT & CVFDT

SPRINT/SLIQ Details

- Split the dataset into one file per attribute
  - (value, record ID)
- Pre-sort each numeric attribute’s file
- Do one scan over each file, find best split point
- Use hash-tables to split the files maintaining sort order
- Recur

Summary

- Decision trees important, need some more work to scale to today's problems
- Disk based methods
    - About one scan per level of tree
- Sampling can produce equivalent trees much faster

Classification

- Objective
  - To build a model of the classifying attribute based upon the other attributes
- Classification given
  - A set of example record, called a training set that is a set of records consists of several attributes
    - Attributes are either
    1. Continuous (i.e. Age: 23,24,25)
    2. Categorical (i.e. Gender: Female, Male)

    One of the attributes called the classifying attribute

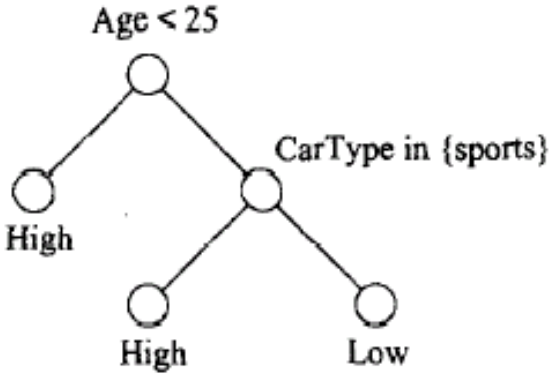
Example – Car Insurance

Continuous attribute

Categorical attribute

rid	Age	Car Type	Risk
0	23	family	High
1	17	sports	High
2	43	sports	High
3	68	family	Low
4	32	truck	Low
5	20	family	High

(a) Training Set



(b) Decision Tree

Gini-Index Classification Algorithm

- Two classification algorithms are presented in the paper
  - SLIQ
  - SPRINT

1. SLIQ
   - Handles large disk-resident data
   - Requires some data per record that stay in memory-resident all the time
2. SPRINT
   - Removes all of the memory restrictions
   - Fast and scalable and it can be easily parallelized

- How good it is?
  - Excellent scale-up, speedup and size-up properties

SPRINT ALGORITHM

- A decision tree classifier is built in two phases
  - Growth phase
  - Prune phase
- Growth phase is computationally much more expensive than pruning

```
Partition(Data S)
    if (all points in S are of the same class) then
        return;
    for each attribute A do
        evaluate splits on attribute A;
    Use best split found to partition S into S1 and S2;
    Partition(S1);
    Partition(S2);

Initial call: Partition(TrainingData)
```

Figure 2: General Tree-growth Algorithm

Tree Building Phase

General tree-growth algorithm (binary tree)

Partition(Data S)

```
        If (all points in S are of the same class) then
return;

        for each attribute A do

                evaluate splits on attribute A;

        Use best split to partition S into S1 and S2;

        Partition(S1);

        Partition(S2);
```

Growth Phase

- Key Issue
  - To find split points that define node tests.
  - Having chosen a split point, how to partition the data
- Data Structures
  - Attribute lists
  - Histograms

Data Structure (Attribute List)

- SPRINT creates an attribute list for each attribute
- Entries are called attribute records which contains
  - Attribute value
  - Class label
  - Index of the record

Age	Class	rid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Car Type	Class	rid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5

Figure 3: Example of attribute lists

Data Structure (Attribute List)

- The initial lists are associated with the root
- As the tree is grown, the attribute lists belonging to each node are partitioned and associated with the children

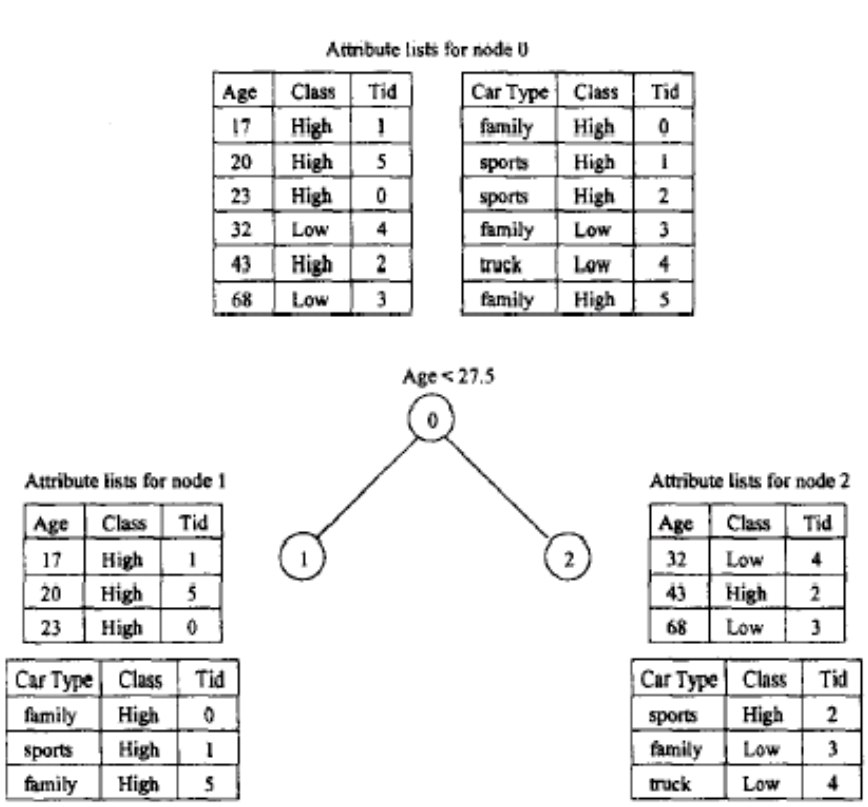


Figure 4: Splitting a node's attribute lists

Data Structure (Histograms)

- For continuous attributes, two histograms are associated with each decision-tree node. These histograms, denoted as C_{above} and C_{below}
  - C_{below}: maintains this distribution for attribute records that already been processed
  - C_{above}: maintains this distribution for attribute records that have not been processed

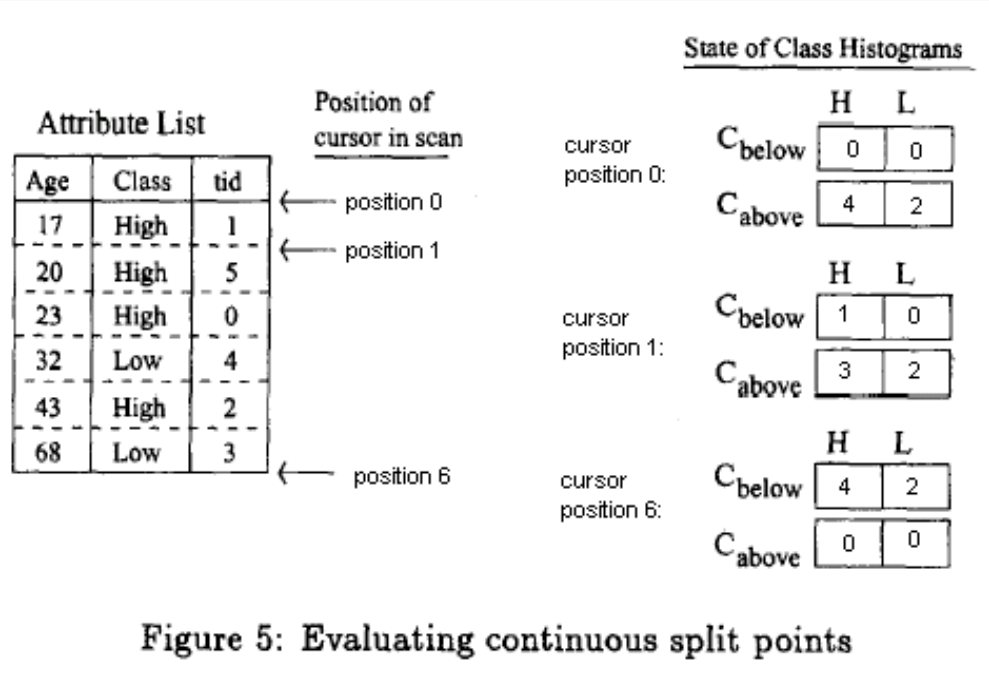


Figure 5: Evaluating continuous split points

Data Structure (Histograms)

- For categorical attributes, one histogram associated with a node. However, only one histogram is needed and called count matrix

Attribute List

Car Type	Class	tid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5

Count Matrix

	H	L
family	2	1
sports	2	0
truck	0	1

Figure 6: Evaluating categorical split points

Finding split points

- While growing the tree, the goal at each node is to determine the split point that “best” divides the training records belonging to the leaf
- In the paper, they use gini index

- $Gini(S)=1-\Sigma p_j^2$
  - Where p_j is the relative of class j in S

- $Gini_{split}(S) = n_1/n(S_1)+n_2/n(S_2)$

Example – Split point for the continuous attributes

Age	Class	Tid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Cursor Position 3:

	H	L
C_{above}	3	0
C_{below}	1	2

$$Gini_{split} = \frac{n_1}{n} gini(S_1) + \frac{n_2}{n} gini(S_2)$$

$$Example:$$

$$Gini_{split3} = \frac{3}{6} gini(S_1) + \frac{3}{6} gini(S_2)$$

$$gini(S_1) = 1 - [(\frac{1}{1})^2 + (\frac{0}{1})^2] = 0$$

$$gini(S_2) = 1 - [(\frac{1}{3})^2 + (\frac{2}{3})^2] = 0.44$$

$$Gini_{split3} = 0.22$$

Example – Split point for the continuous attributes

$$Gini_{split0} = 0.44$$

$$Gini_{split1} = 0.40$$

$$Gini_{split2} = 0.33$$

$$Gini_{split3} = 0.22$$

$$Gini_{split4} = 0.41$$

$$Gini_{split5} = 0.26$$

$$Gini_{split6} = 0.44$$

- After finding all the gini indexes we choose the lowest as the split point
- Therefore, we split at position 3 where the candidate split point is the mid-point between age 23 and 32 (i.e. Age < 27.5)

Example – Split point for the categorical attributes

	H	L
Family	2	1
Sports	2	0
Truck	0	1

Example

$$gini_{split(sport)} = \frac{2}{6} gini(S_1) + \frac{4}{6} gini(S_2)$$

$$gini(S_1) = 1 - (\frac{2}{2})^2 + (\frac{0}{2})^2 = 0$$

$$gini(S_2) = 1 - (\frac{2}{4})^2 + (\frac{2}{4})^2 = 0.5$$

$$gini_{split(sport)} = 0.33$$

$$gini_{split(family)} = 0.44$$

$$gini_{split(truck)} = 0.266$$

Example – Split point for the categorical attributes

- Once the best split point has been found for a node, we then execute the split by creating child nodes and dividing the attribute records between them

- For the rest of the attribute lists (i.e. CarType) we need to retrieve the information by using rids

Comparison with SLIQ

- SLIQ does not have separate sets of attribute lists for each node
- Advantage
  - Do not have to rewrite these lists during a split
  - Reassignment of records is easy
- Disadvantage
  - It must stay in memory all the time which limits the amount of data that can be classified by SLIQ

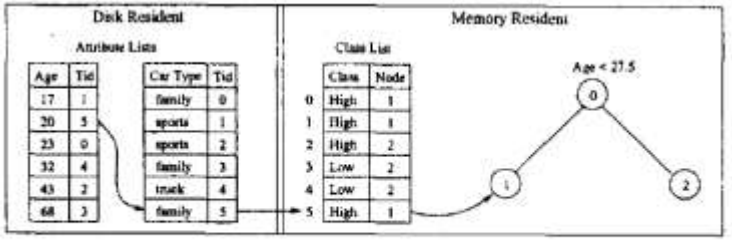


Figure 7: Attribute and Class lists in SLIQ

SPRINT was not to outperform SLIQ. Rather, the purpose of the algorithm is to develop an accurate classifier for datasets, and to develop a classifier efficiently. Furthermore, SPRINT is designed to be easily parallelizable, thus the workload can be shared among N processor

Conclusion

1. Decision Tree Advantages.
2. Decision Tree Disadvantages.
3. Decision Tree Applications.
4. Decision Tree Tool.
5. Demo

Decision Tree Advantages

- Decision trees are powerful and popular tools for classification and prediction.
- Simpler and ease of use.
- They are able to handle both numerical and categorical attributes.
- Easy to understand.
- State is recorded in memory.
- Provide a clear indication of which fields are most important for prediction or classification.
- Can be learned.

Decision Tree Disadvantages

- Each tree is “unique” sequence of tests, so little common structure.
- Perform poorly with many class and small data.
- Need as many examples as possible.
- Higher CPU cost - but not much higher.
- Learned decision trees may contain errors.
- Hugely impacted by data input.
- Duplicate in sub trees

Decision Tree Applications

- Medical diagnosis.
- Credit risk analysis.
- Library book use.
- Etc….

WEKA Software application

Waikato Environment for Knowledge Analysis (Weka)

- Is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand.
- It is a collection of machine learning algorithms for data mining tasks which can either be applied directly to a dataset or your own Java code.
- Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.
- Weka input is a dataset Attribute-Relation File Format file (ARFF file).

ARFF file contents

1. **Dataset name.**
2. **A header:** Describes the attribute types
3. **Data section:** Comma separated list of data

Comment

```
% This is a toy example, the UCI weather dataset.
% Any relation to real weather is purely conditional.
```

Dataset Name

```
@relation weather
```

Attributes

```
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE,FALSE}
@attribute play {yes, no}
```

Data

```
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

WEKA Demo

Thank You ☺