[redacted]

Decision Trees

What is a Decision Tree?

Decision Tree

- A tool using tree-like graph or model decisions
- Displays an algorithm that only contains conditional control statement
- Used to measure the probabilities of an even to occur
- Popular in machine learning, maketing and algorithms for Big Data
- Some can adapt to the situation : classification and regression trees

Decision trees elements

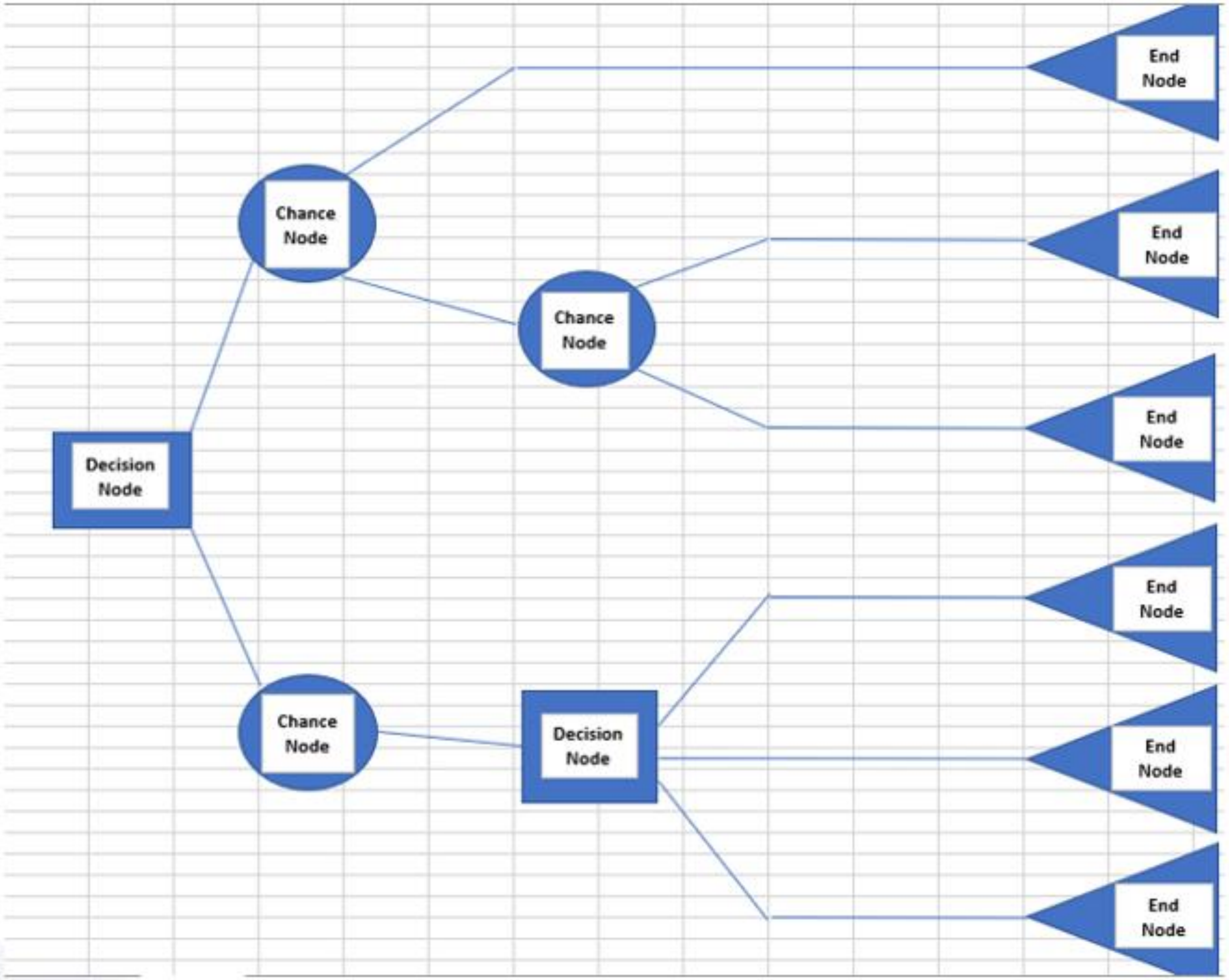
Nodes

- Decision nodes : A choice must be made
- Chances nodes : The choice depends on outer factors
- End nodes : Final result of the combination of events

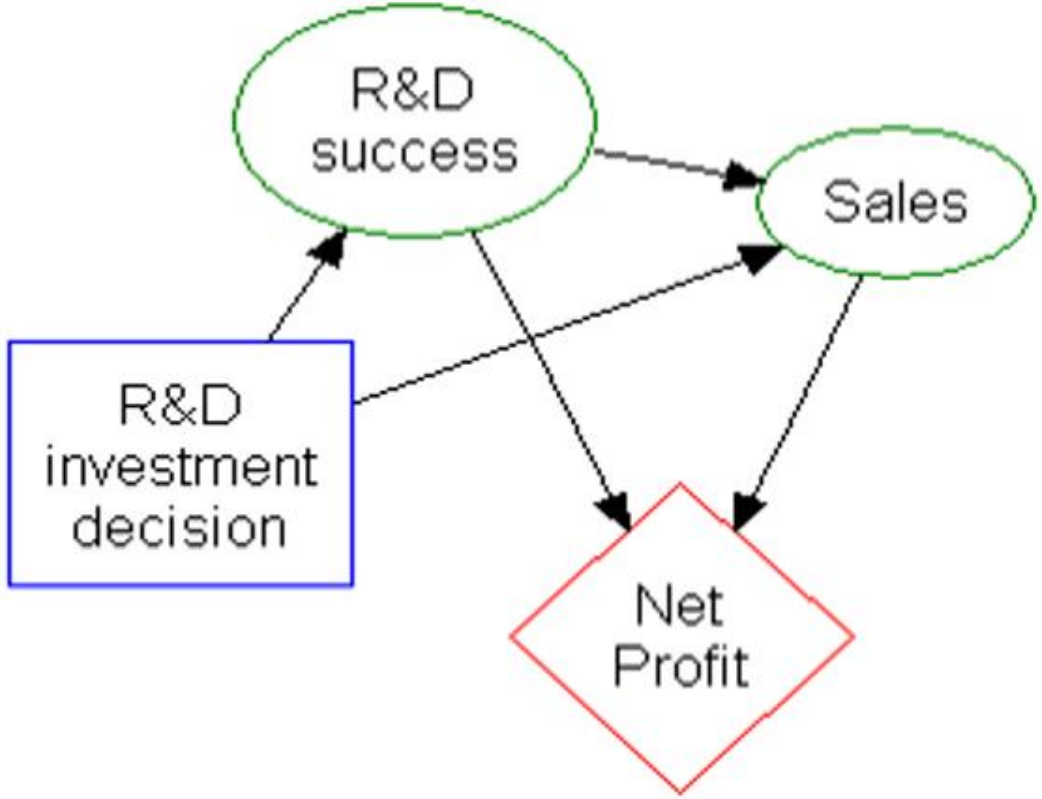
Branches

- Decision branches : Extending from a decision node
- Event branches : Extending from a chance node

Example of a decision tree



- Example of influence diagram



Advantages and Disadvantages

Advantages

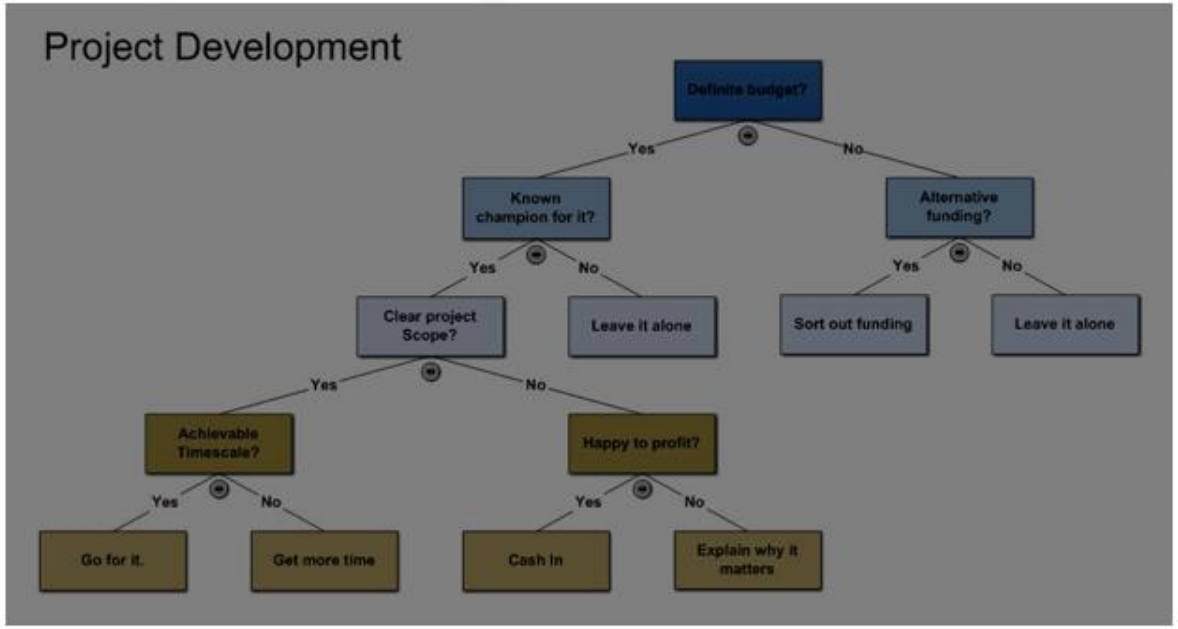
- Easy to use and to understand
- Flexibility due to the possibility to add new scenarios
- Ability to have value even with few hard data
- Ability to determine the worst and the best and to expect values according to different scenarios.
- The use of a white box model

Disadvantage

- Sometimes calculations can get complex, especially if any values are uncertain/missing

Simple decision tree

In this case the decision tree model is a binary tree



Linear decision tree

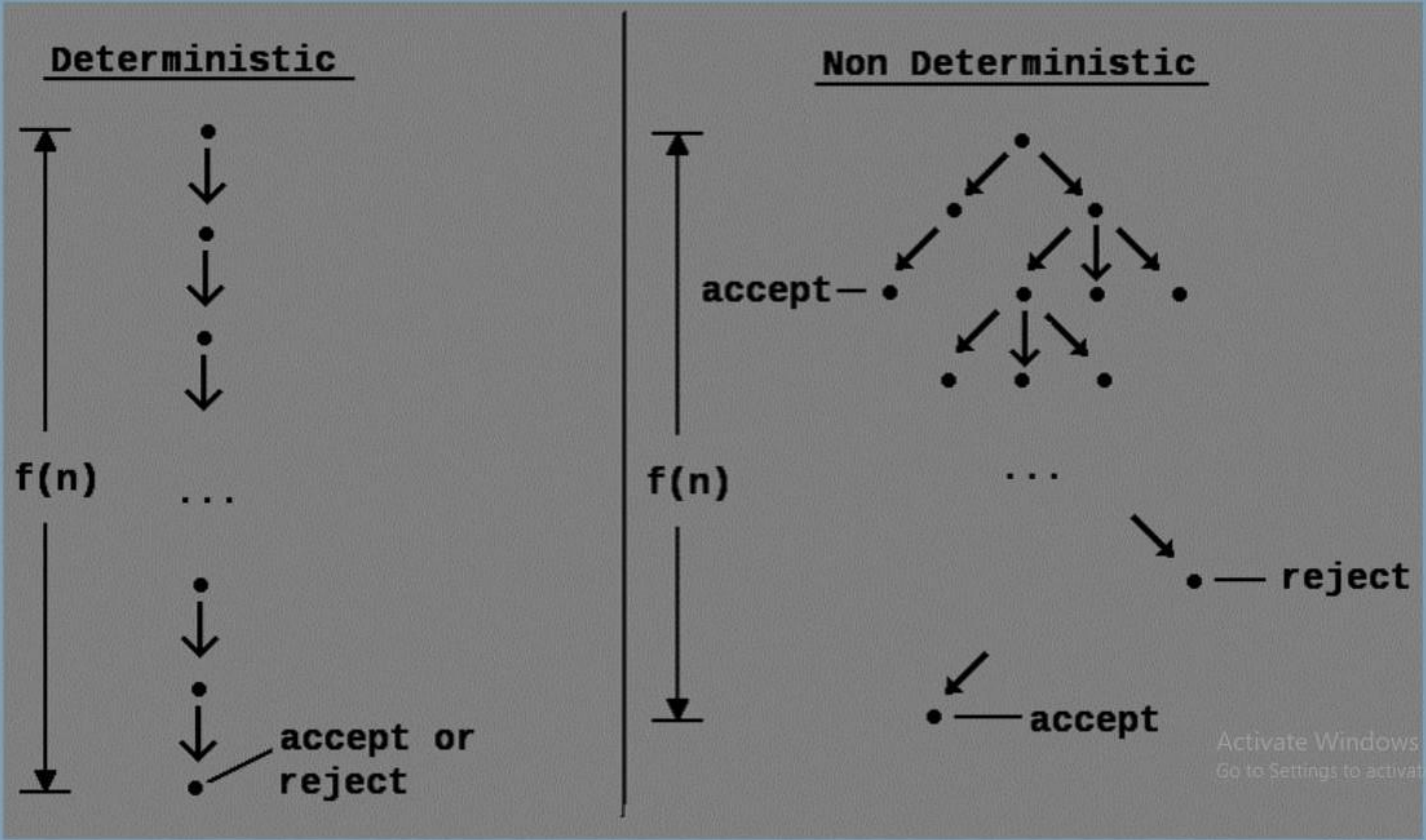
Linear decision trees have three output branches. A linear function f(x1,...,xi) is being tested and branching decisions are made based on the sign of the function (negative, positive, or 0).

Deterministic decision tree

A deterministic decision tree is a rooted ordered binary tree T.
Each internal node of T is labeled with a variable xi and each leaf
is labeled with a value 0 or 1. If the current node is a leaf then
the evaluation stops. Otherwise the variable xi that labels the
current node is queried. If xi =0, then left subtree will be
recursively evaluated, if xi = 1 then the right one.

Non deterministic decision tree

non-deterministically, decision trees may give more than a single answer to a question. In such cases, the final answer may be given as a majority vote amongst the occurrences of each possible answer



Deterministic
f(n)
...
accept or reject
Non Deterministic
f(n)
accept
...
reject
accept

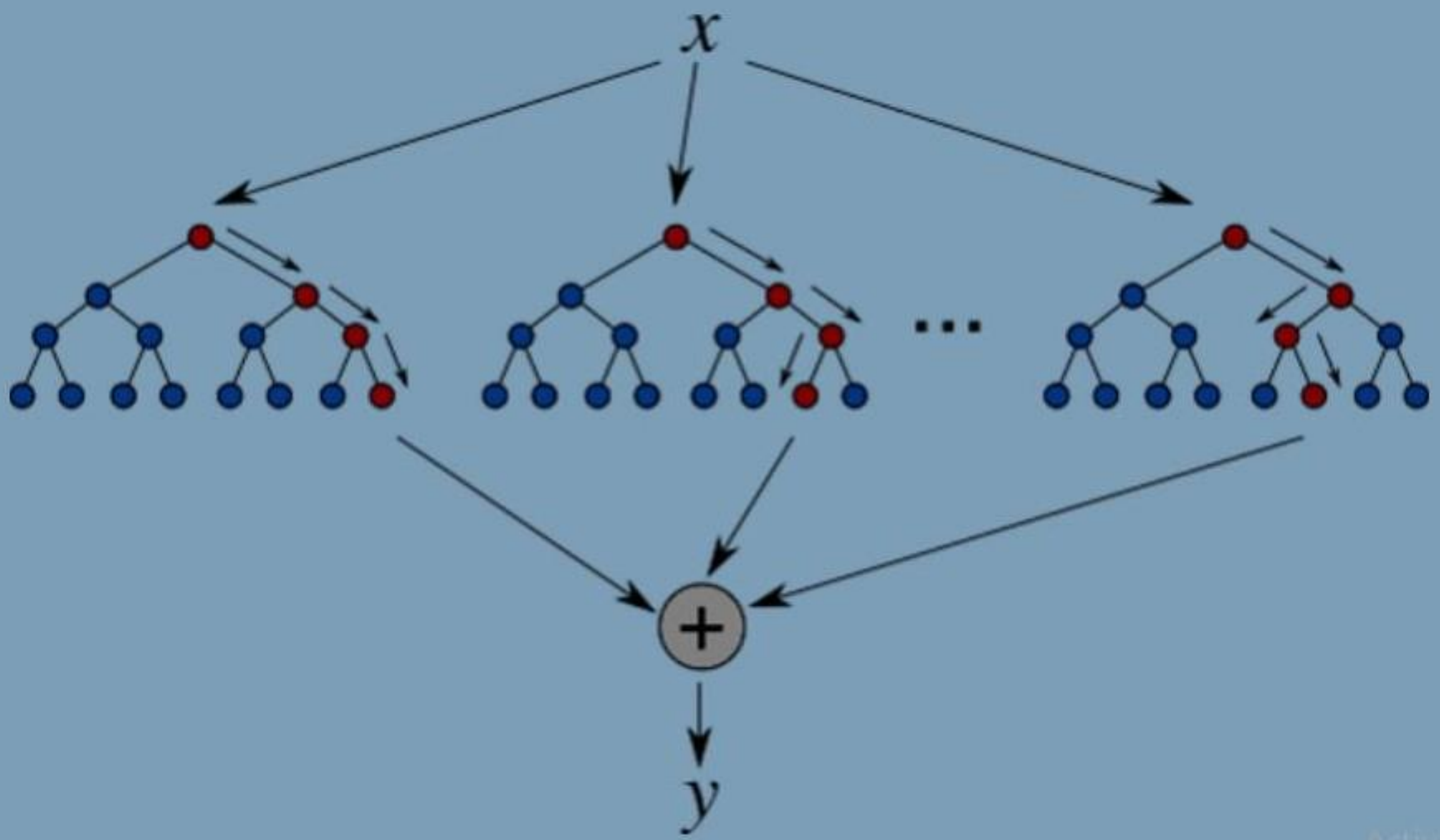
Randomized decision tree

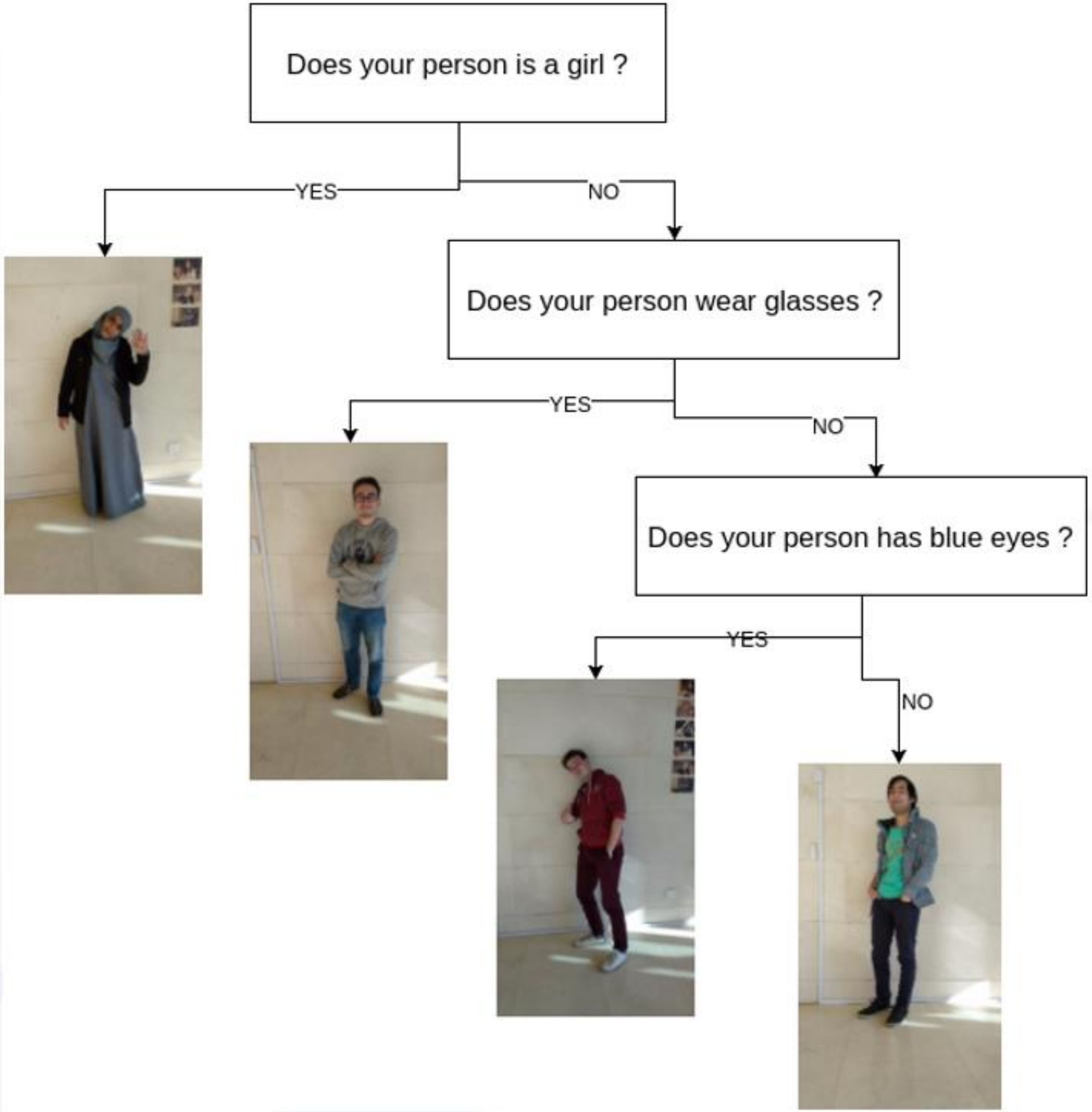
each tree of the forest is built on a bootstrap sample of the training data

at each tree node, the best split is chosen among a reduced set of variable (chosen at random)

each tree is fully grown (no pruning)

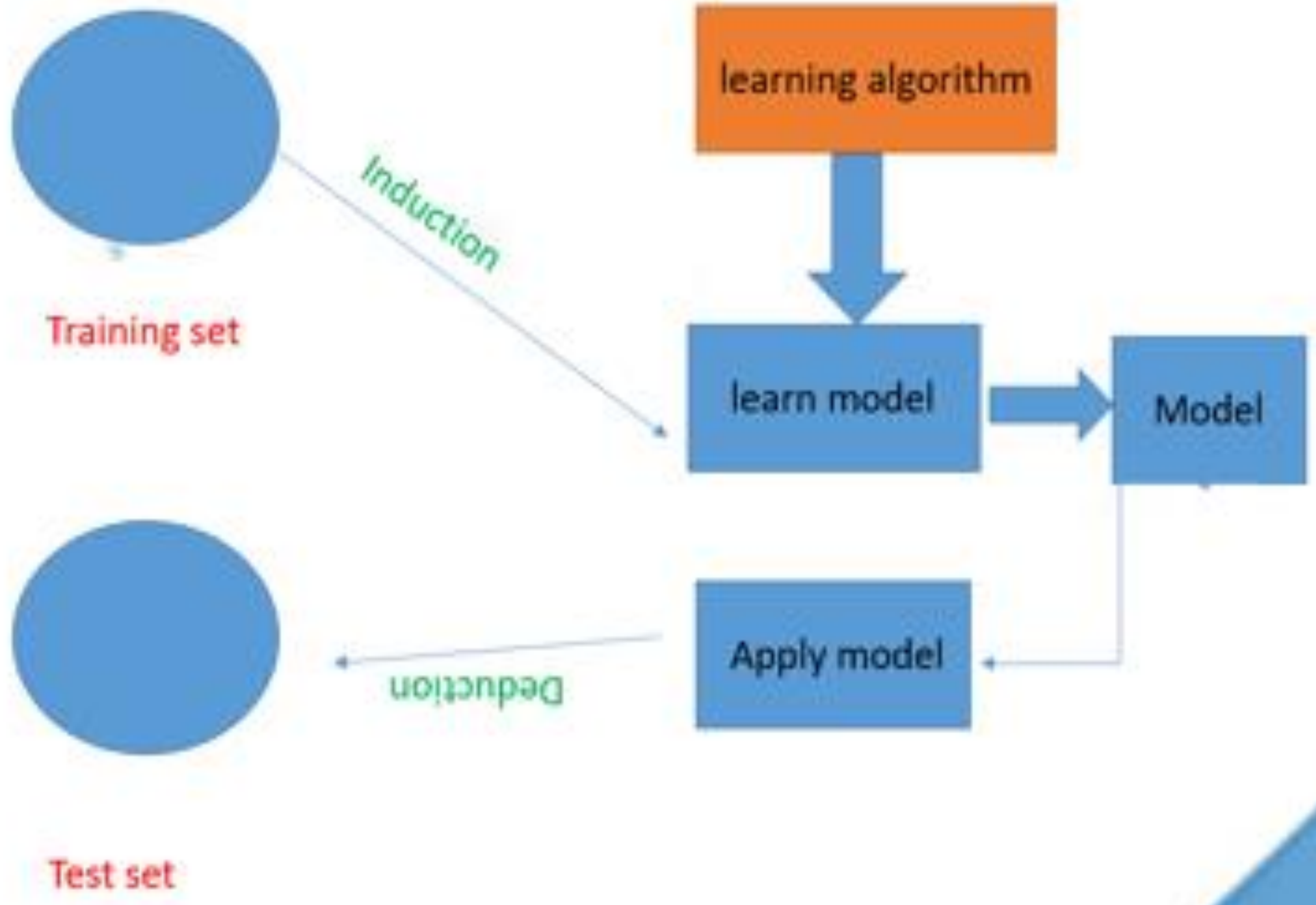
the prediction is made by a majority vote over all trees


x
y


Does your person is a girl ?
YES
NO
Does your person wear glasses ?
YES
NO
Does your person has blue eyes ?
YES
NO

Classification

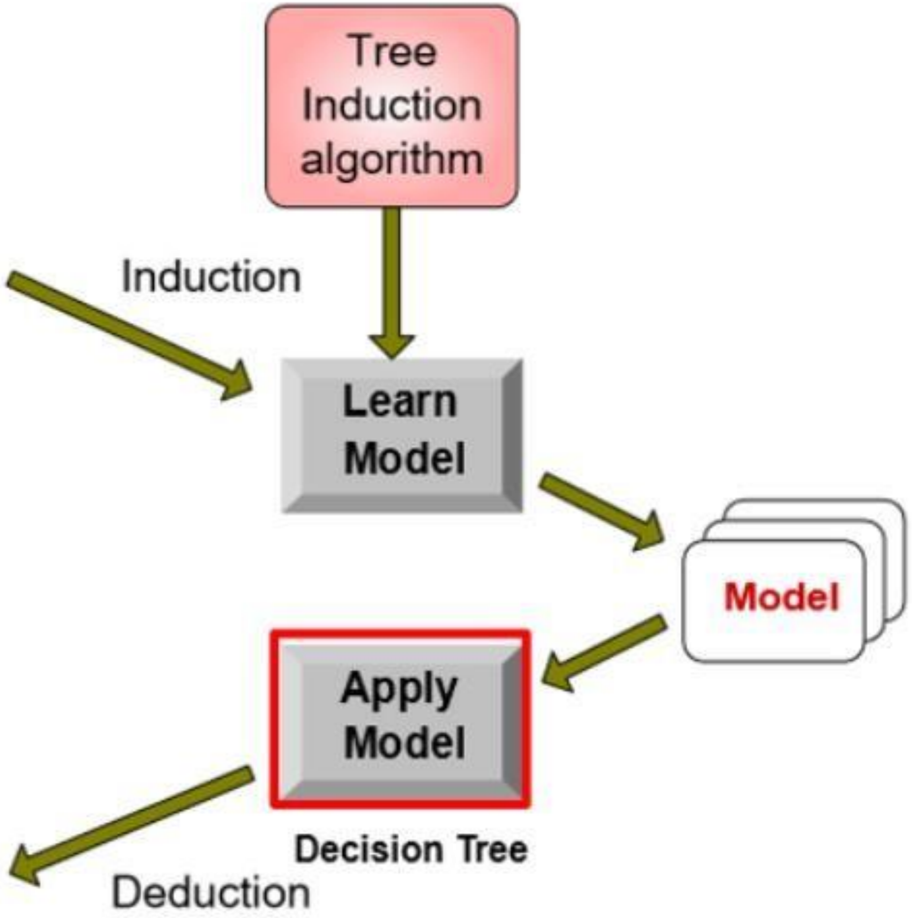
- General Approach to a Classification Problem



Example

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

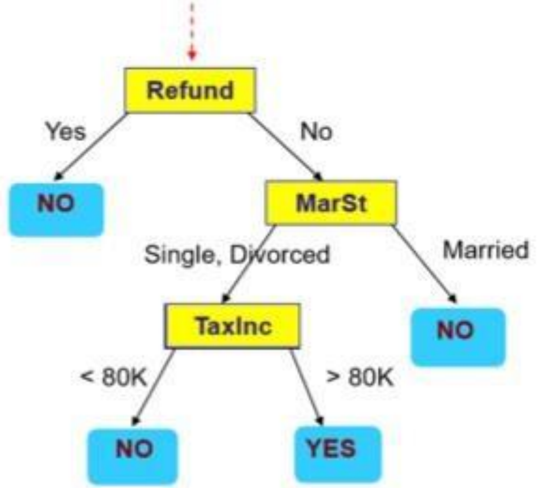
Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

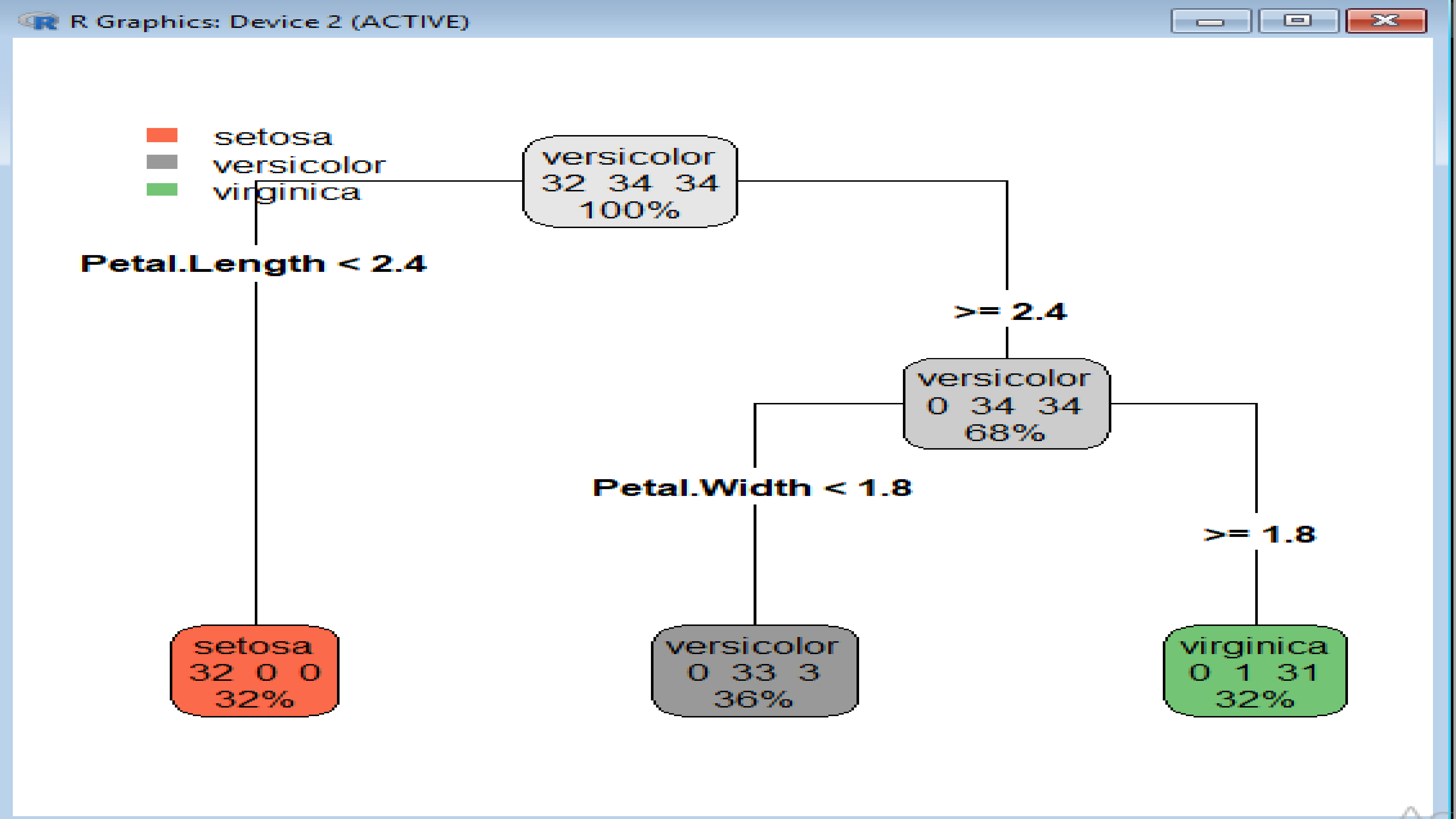
Example

Start from the root to tree

C:\Users\afefa\Desktop\gg.R - R Editor

```
library(rpart)
library(rpart.plot)
s<-sample(150,100)
iris_train<-iris[s,]
iris_test<-iris[-s,]
dtm<-rpart(Species~., iris_train, method="class")

rpart.plot(dtm,type=4, extra=101)
```


R Graphics: Device 2 (ACTIVE)
setosa
versicolor
virginica
versicolor
32 34 34
100%
Petal.Length < 2.4
>= 2.4
versicolor
0 34 34
68%
Petal.Width < 1.8
>= 1.8
setosa
32 0 0
32%
versicolor
0 33 3
36%
virginica
0 1 31
32%

Decision Tree Induction

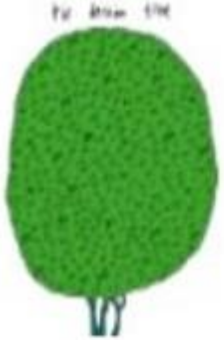
- Hunt’s Algorithm
- CART
- ID3, C4.5
- LIQ,SPRINT

Hunt's Algorithm

- Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, \ldots y_c\}$, where y is the target variable with c number of classes.
- The following is a recursive definition of Hunt's algorithm:

Step 1 :
If all the records in D_t belong to the same class y_t, then node **t** is a leaf node labeled as y_t.

Step 2 :
If **Dt** contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.



Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split

How to determine the Best Split

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

How to determine the Best Split

Measure of Impurity: GINI

- Gini Index for a given node t :

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

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

How to determine the Best Split

Examples for computing GINI

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

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	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

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

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

How to determine the Best Split

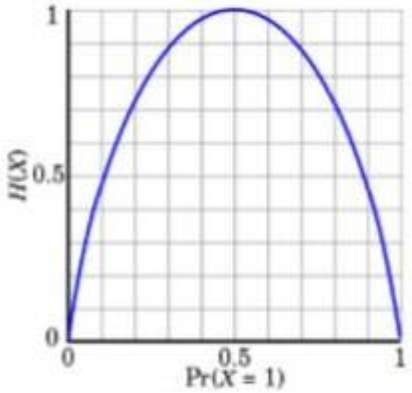
Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$\boxed{Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)}$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Measures homogeneity of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations



How to determine the Best Split

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

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

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

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

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

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

How to determine the Best Split

Splitting Based on INFO...

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Where the Parent Node, p is split into k partitions;
and n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Designed to overcome the disadvantage of Information Gain

How to determine the Best Split

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$\boxed{Error(t) = 1 - \max_i P(i \mid t)}$$

- Measures misclassification error made by a node.
  - ✓ Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - ✓ Minimum (0.0) when all records belong to one class, implying most interesting information

- ID3 algorithm

ID3 algorithm

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

     if subset is pure: STOP

     else: Split (child_node, {subset})
- Ross Quinlan (ID3: 1986), (C4.5: 1993)

- More details about ID3 algorithm

https://www.youtube.com/watch?v=_XhOdSLlE5c