Machine Learning

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir

https://www.aparat.com/mehran.safayani

https://github.com/safayani/machine_learning_course

Department of Electrical and computer engineering, Isfahan university of technology, Isfahan, Iran

Decision Tree

These slides are modified based on the original slides by Dr. Debasis Samanta

This presentation slides includes…

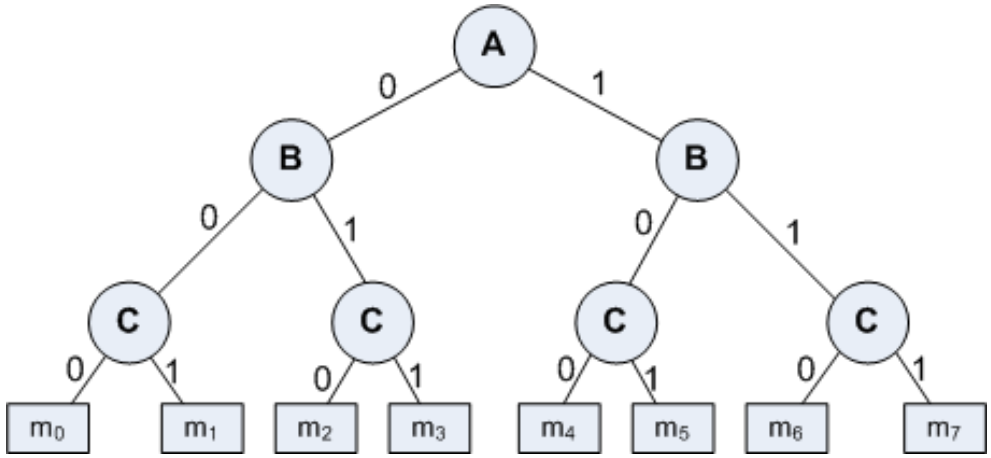
- Concept of Decision Tree
- Use of Decision Tree to classify data
- Basic algorithm to build Decision Tree
  - Some illustrations
- Concept of Entropy
- Decision Tree induction algorithms
  - ID3
  - CART
- Pruning of decision tree
- Random Forest
- Decision tree for regression

Basic Concept

- A Decision Tree is an important data structure known to solve many computational problems

Example 9.1: Binary Decision Tree

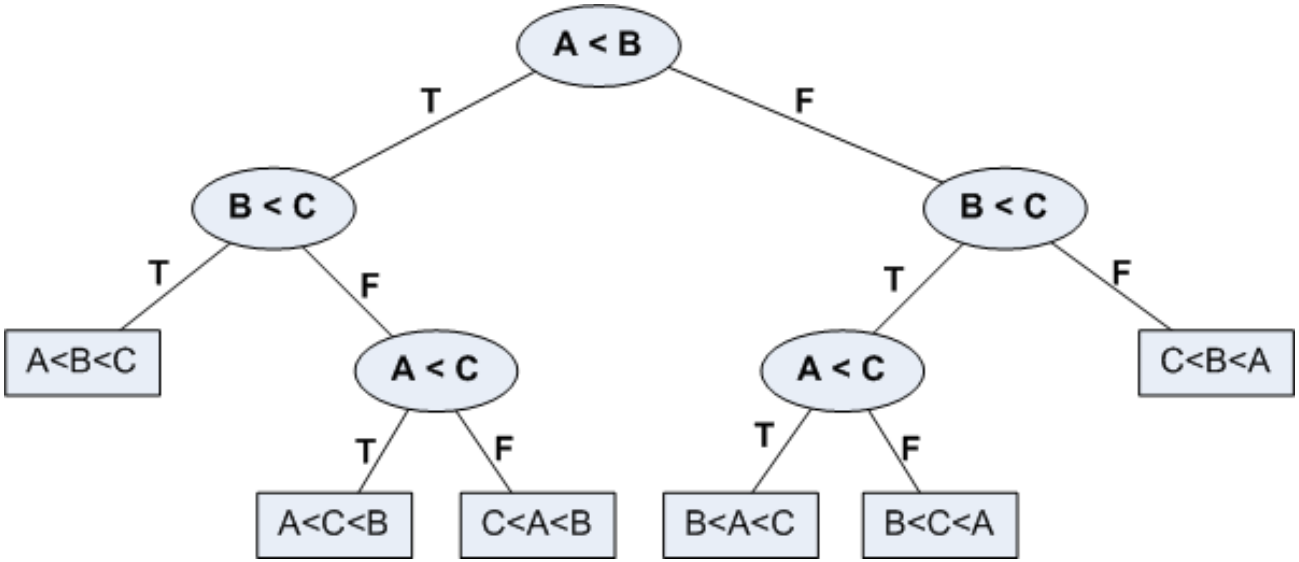
A	*B*	*C*	*f*
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

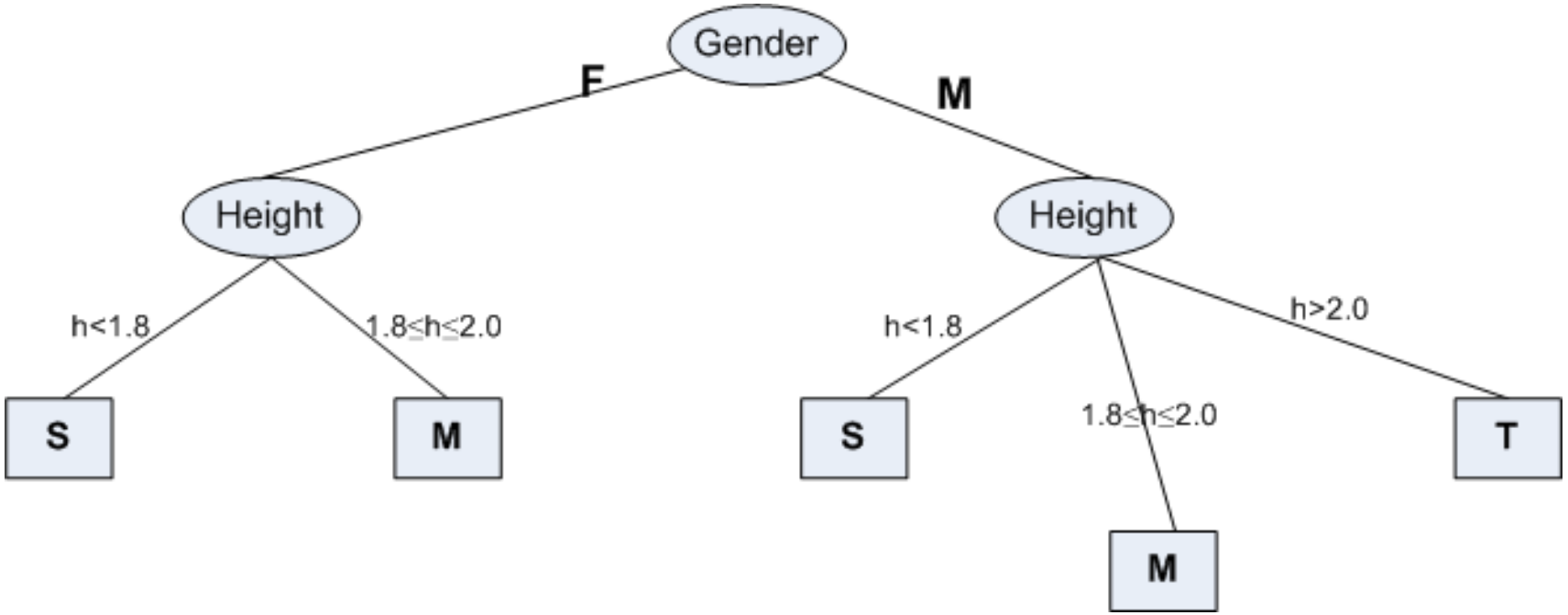


Basic Concept

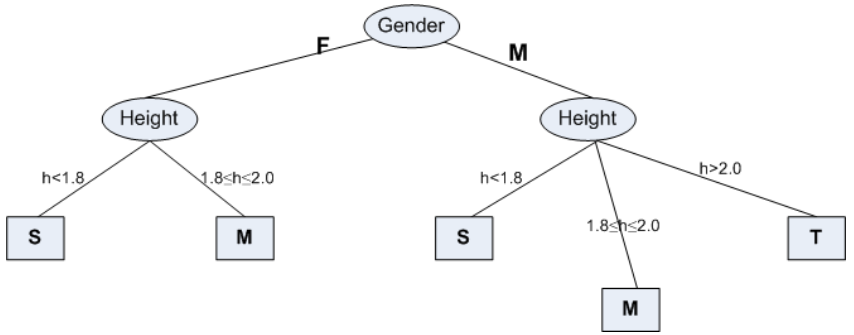
- Decision tree is also possible where attributes are of continuous data type

Decision Tree with numeric data




Gender
F
M
Height
Height
h<1.8
1.8≤h≤2.0
h<1.8
h>2.0
1.8≤h≤2.0
S
M
S
T
M

Some Characteristics

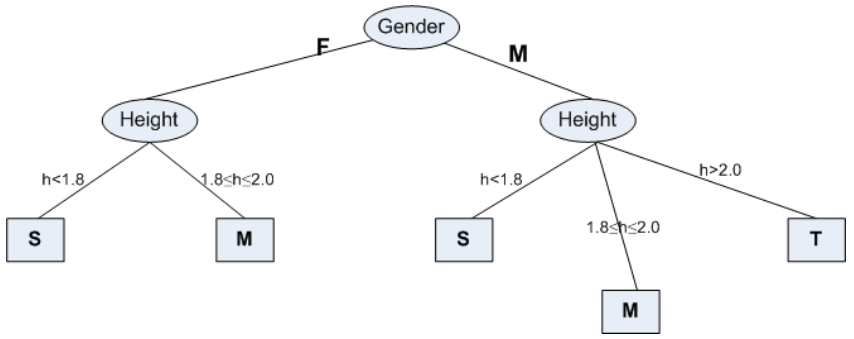


- Decision tree may be *n*-ary, $n \geq 2$.
- There is a special node called root node.
- All nodes drawn with circle (ellipse) are called internal nodes.
- All nodes drawn with rectangle boxes are called terminal nodes or leaf nodes.
- Edges of a node represent the outcome for a value of the node.
- In a path, a node with same label is never repeated.
- Decision tree is not unique, as different ordering of internal nodes can give different decision tree.

Decision Tree and Classification Task

- Decision tree helps us to classify data.
  - Internal nodes are some attribute
  - Edges are the values of attributes
  - External nodes are the outcome of classification

- Such a classification is, in fact, made by posing questions starting from the root node to each terminal node.



Decision Tree and Classification Task

Vertebrate Classification

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	**Mammal**
Python	Cold	scales	no	no	no	no	yes	**Reptile**
Salmon	Cold	scales	no	yes	no	no	no	**Fish**
Whale	Warm	hair	yes	yes	no	no	no	**Mammal**
Frog	Cold	none	no	semi	no	yes	yes	**Amphibian**
Komodo	Cold	scales	no	no	no	yes	no	**Reptile**
Bat	Warm	hair	yes	no	yes	yes	yes	**Mammal**
Pigeon	Warm	feathers	no	no	yes	yes	no	**Bird**
Cat	Warm	fur	yes	no	no	yes	no	**Mammal**
Leopard	Cold	scales	yes	yes	no	no	no	**Fish**
Turtle	Cold	scales	no	semi	no	yes	no	**Reptile**
Penguin	Warm	feathers	no	semi	no	yes	no	**Bird**
Porcupine	Warm	quills	yes	no	no	yes	yes	**Mammal**
Eel	Cold	scales	no	yes	no	no	no	**Fish**
Salamander	Cold	none	no	semi	no	yes	yes	**Amphibian**

What are the class label of Dragon and Shark?

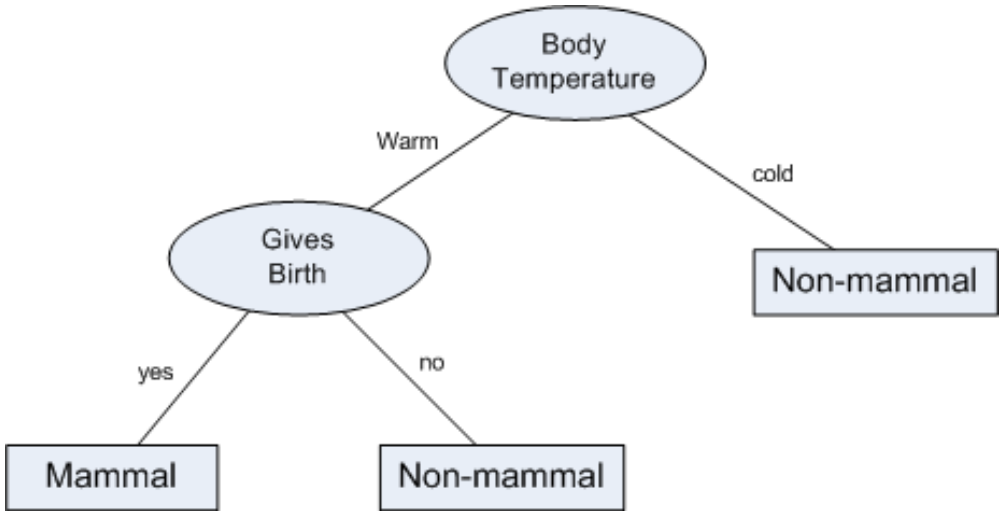
Decision Tree and Classification Task

Vertebrate Classification

- Suppose, a new species is discovered as follows.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Gila Monster	cold	scale	no	no	no	yes	yes	?

- Decision Tree that can be inducted based on the data is as follows.



Decision Tree and Classification Task

- Previous example illustrates how we can solve a classification problem by asking a series of question about the attributes.
  - Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class-label of the test.
- The series of questions and their answers can be organized in the form of a decision tree
  - As a hierarchical structure consisting of nodes and edges
- Once a decision tree is built, it is applied to any test to classify it.

Definition of Decision Tree

Definition : **Decision Tree**

Given a database $D = \{t_1, t_2, \ldots\ldots, t_n\}$, where t_i denotes a tuple, which is defined by a set of attribute $A = \{A_1, A_2, \ldots\ldots, A_m\}$. Also, given a set of classes $C = \{c_1, c_2, \ldots\ldots, c_k\}$.

A decision tree ***T*** is a tree associated with *D* that has the following properties:

- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

Building Decision Tree

- In principle, there are exponentially many decision tree that can be constructed from a given database (also called training data).
  - Some of the tree may not be optimum
  - **Some algorithms for building decision tree:**
    - ID3
    - CART, etc.

Built Decision Tree Algorithm

- **Algorithm BuiltDT**
- Input: D : Training data set Output: T : Decision tree

Steps

1. If all tuples in D belongs to the same class C_j
   Add a leaf node labeled as C_j
   Return *// Termination condition*

2. **Select** an attribute A_i (so that it is not selected twice in the same branch)

3. **Partition** $D = \{ D_1, D_2, \ldots, D_p \}$ based on p different values of A_i in D

4. For each $D_k \in D$
   Create a node and add an edge between D and D_k with label as the A_i's attribute value in D_k

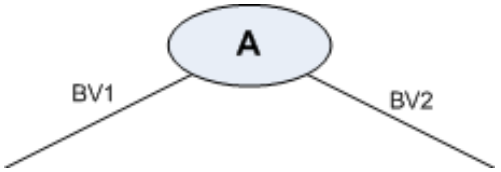
5. For each $D_k \in D$
   BuildTD(D_k) *// Recursive call*

6. Stop

Node Splitting in **BuildDT** Algorithm

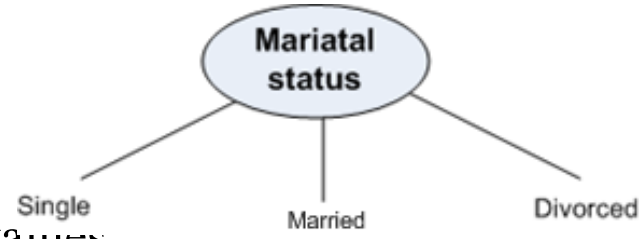
- BuildDT algorithm must provides a method for expressing an attribute test condition and corresponding outcome for different attribute type

- **Case: Binary attribute**
  - This is the simplest case of node splitting
  - The test condition for a binary attribute generates only two outcomes



Node Splitting in **BuildDT** Algorithm

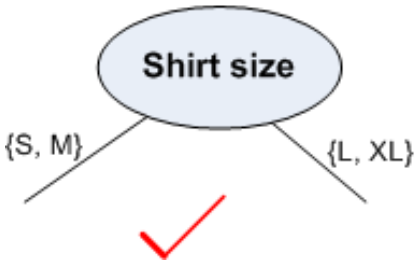
- **Case: Nominal attribute**
  - Since a nominal attribute can have many values, its test condition can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: Outcome depends on the number of distinct values for the corresponding attribute

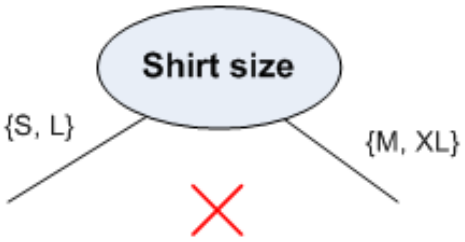


  - Binary splitting by grouping attribute values

Node Splitting in **BuildDT** Algorithm

- **Case: Ordinal attribute**
  - It also can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: It is same as in the case of nominal attribute
  - Binary splitting attribute values should be grouped maintaining the order property of the attribute values





Node Splitting in **BuildDT** Algorithm

- **Case: Numerical attribute**
  - For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
    - Binary outcome: $A > v$ or $A \leq v$
      - In this case, decision tree induction must consider all possible split positions
    - Range query : $v_i \leq A < v_{i+1}$ for $i = 1, 2, \ldots, q$ (if q number of ranges are chosen)
      - Here, q should be decided a priori

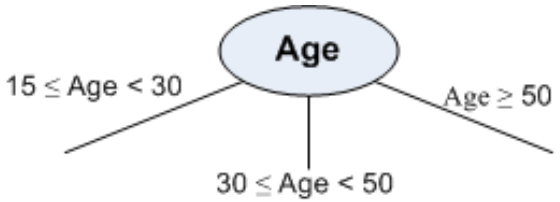


Illustration : **BuildDT** Algorithm

Illustration of BuildDT Algorithm

- Consider a training data set as shown.

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Attributes:

Gender = {Male(M), Female (F)} // Binary attribute
Height = {1.5, …, 2.5} // Continuous attribute

Class = {Short (S), Medium (M), Tall (T)}

Given a person, we are to test in which class s/he belongs

Illustration : **BuildDT** Algorithm

- To built a decision tree, we can select an attribute in two different orderings: <Gender, Height> or <Height, Gender>
- Further, for each ordering, we can choose different ways of splitting
- Different instances are shown in the following.
- **Approach 1 : <Gender, Height>**

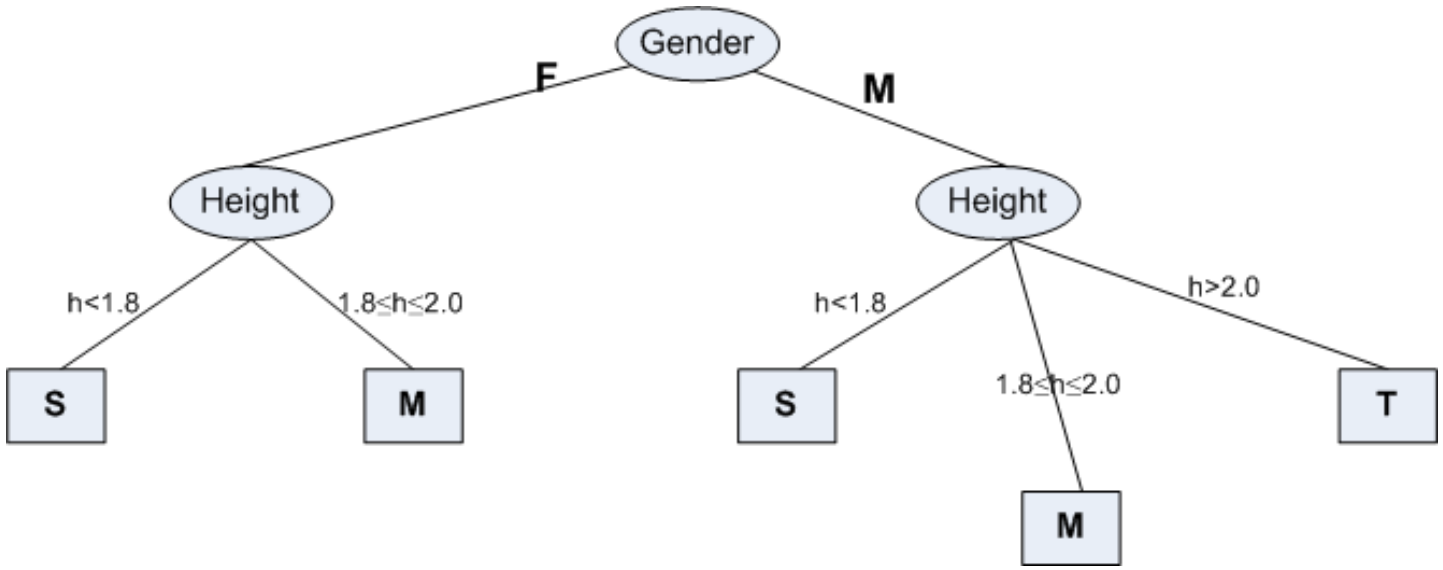


Illustration : **BuildDT** Algorithm

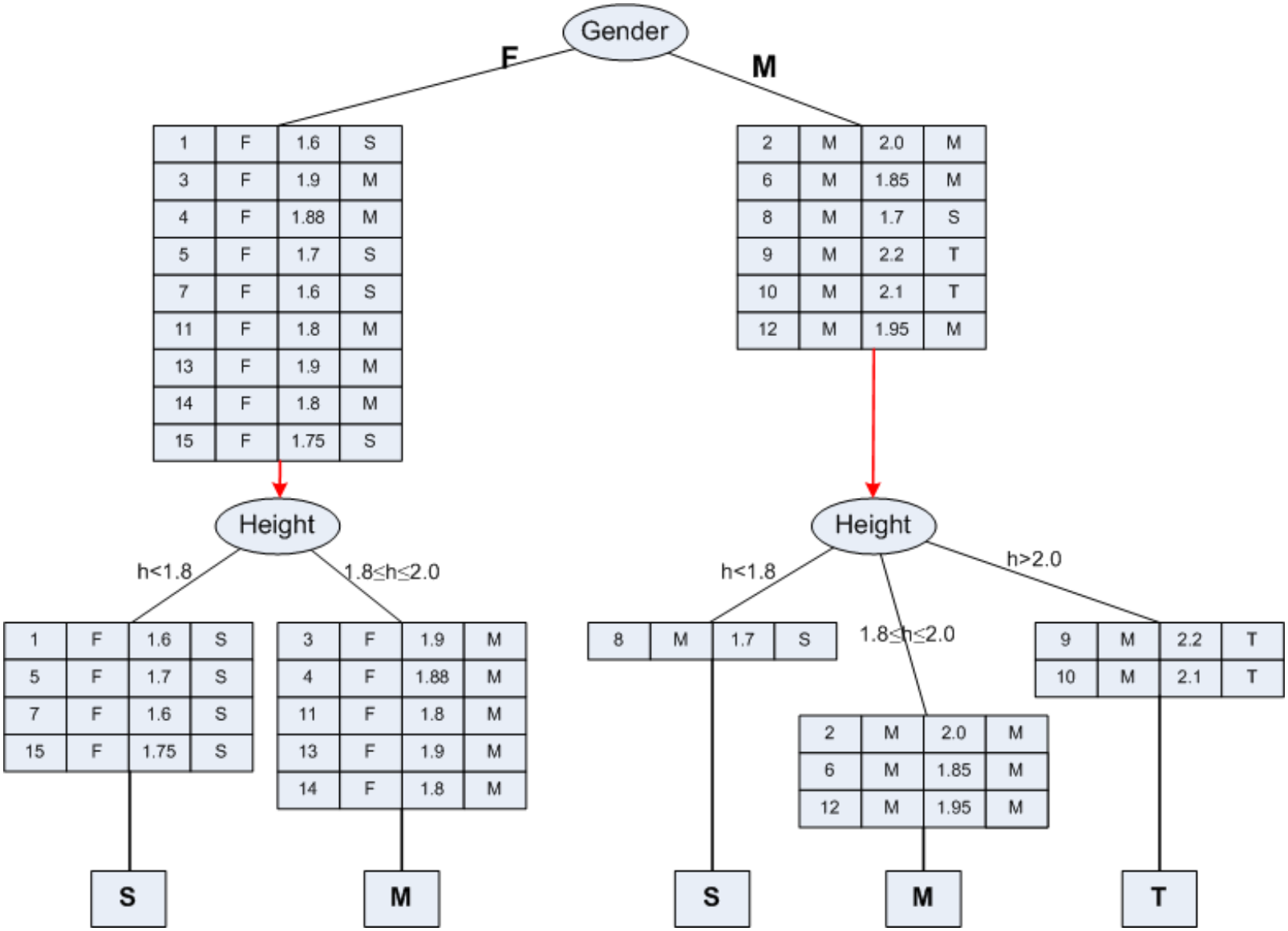
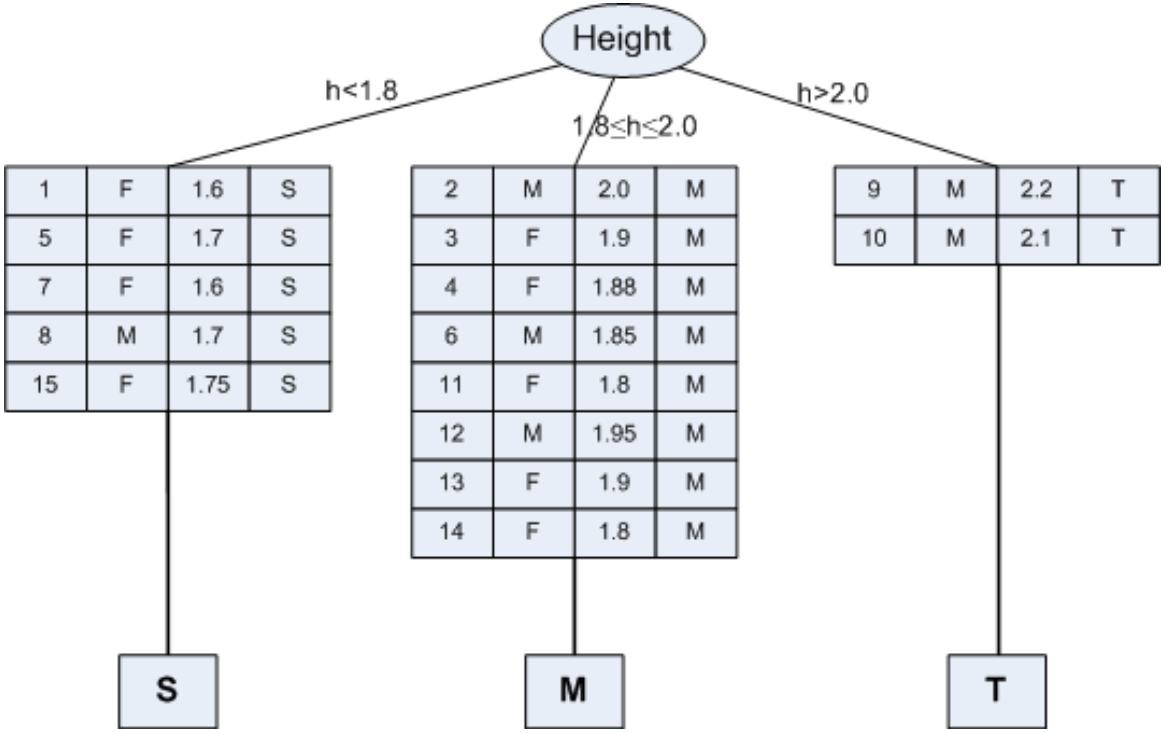


Illustration : **BuildDT** Algorithm

- **Approach 2 : <Height, Gender>**



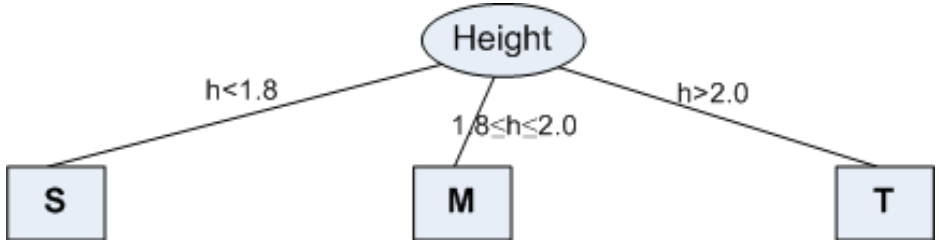


Illustration : **BuildDT** Algorithm

Illustration of BuildDT Algorithm

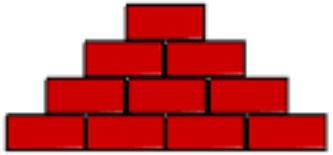
- Consider an anonymous database as shown.

A1	A2	A3	A4	Class
a11	a21	a31	a41	C1
a12	a21	a31	a42	C1
a11	a21	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a31	a42	C1
a11	a21	a32	a42	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a12	a22	a31	a42	C1

- Is there any "clue" that enables to select the "best" attribute first?
- Suppose, following are two attempts:
  - A1→A2→A3→A4 [naïve]
  - A3→A2→A4→A1 [Random]
- Draw the decision trees in the above-mentioned two cases.
- Are the trees different to classify any test data?
- If any other sample data is added into the database, is that likely to alter the decision tree already obtained?

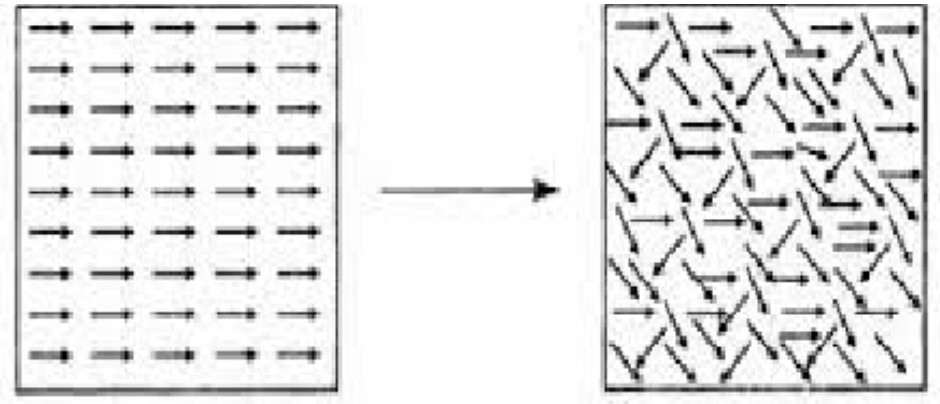
Concept of Entropy

Concept of Entropy



More **ordered**
less **entropy**

Less ordered
higher entropy



More organized or
ordered (less **probable**)

Less organized or
disordered (**more** probable)

Entropy of a Training Set

- If there are k classes $c_1, c_2 \ldots\ldots, c_k$ and p_i for $i = 1\ to\ k$ denotes the number of occurrences of classes c_i divided by the total number of instances (i.e., the frequency of occurrence of c_i) in the training set, then entropy of the training set is denoted by

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

Here, E is measured in "bits" of information.

Note:

- The above formula should be summed over the non-empty classes only, that is, classes for which $p_i \neq 0$
- E is always a positive quantity
- E takes it's minimum value (zero) if and only if all the instances have the same class (i.e., the training set with only **one** non-empty class, for which the probability 1).
- Entropy takes its maximum value when the instances are equally distributed among k possible classes. In this case, the maximum value of E is $log_2\, k$.

Entropy of a Training Set

OPTH dataset

Consider the OTPH data shown in the following table with total 24 instances in it.

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	1
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	2
3	2	1	1	3
3	2	1	2	1
3	2	2	1	3
3	2	2	2	3

Entropy of a training set

Specification of the attributes are as follows.

Age	Eye Sight	Astigmatic	Use Type
1: Young	1: Myopia	1: No	1: Frequent
2: Middle-aged	2: Hypermetropia	2: Yes	2: Less
3: Old			

Class: 1: Contact Lens 2:Normal glass 3: Nothing

In the OPTH database, there are 3 classes and 4 instances with class 1, 5 instances with class 2 and 15 instances with class 3. Hence, entropy E of the database is:

$$E = -\frac{4}{24}\log_2\frac{4}{24} - \frac{5}{24}\log_2\frac{5}{24} - \frac{15}{24}\log_2\frac{15}{24} = 1.3261$$

Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.
- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.
- Different algorithms have been proposed to take a good control over
    1. Choosing the best attribute to be splitted, and
    2. Splitting criteria
- Several algorithms have been proposed for the above tasks. In this lecture, we shall limit our discussions into two important of them
    - **ID3**
    - **CART**

Algorithm ID3

ID3: Decision Tree Induction Algorithms

- Quinlan [1986] introduced the ID3, a popular short form of **I**terative **D**ichotomizer 3 for decision trees from a set of training data.
- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.
- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

Algorithm ID3

- In ID3, entropy is used to measure how informative a node is.
  - It is observed that splitting on any attribute has the property that average entropy of the resulting training subsets will be less than or equal to that of the previous training set.
- ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
  - The attribute with the largest value of information gain is chosen as the splitting attribute and
  - it partitions into a number of smaller training sets based on the distinct values of attribute under split.

Defining Information Gain

- We consider the following symbols and terminologies to define information gain, which is denoted as α.
- $D \equiv$ denotes the training set at any instant
- $|D| \equiv$ denotes the size of the training set D
- $E(D) \equiv$ denotes the entropy of the training set D
- The entropy of the training set D

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

  - where the training set D has $c_1, c_2, \ldots, c_k$, the k number of distinct classes and
  - p_i, $0 < p_i \leq 1$ is the probability that an arbitrary tuple in D belongs to class c_i ($i = 1, 2, \ldots, k$).

Defining Information Gain

- p_i can be calculated as

$$p_i = \frac{|C_{i,D}|}{|D|}$$

- where $C_{i,D}$ is the set of tuples of class c_i in D.

- Suppose, we want to partition D on some attribute A having m distinct values $\{a_1, a_2, \ldots, a_m\}$.

- Attribute A can be considered to split D into m partitions $\{D_1, D_2, \ldots, D_m\}$, where D_j $(j = 1,2, \ldots, m)$ contains those tuples in D that have outcome a_j of A

Defining Information Gain

Definition 9.4: **Weighted Entropy**

The weighted entropy denoted as $E_A(D)$ for all partitions of D with respect to A is given by:

$$E_A(D) = \sum_{j=1}^{m} \frac{|D_j|}{|D|} E(D_j)$$

Here, the term $\frac{|D_j|}{|D|}$ denotes the weight of the j-th training set.

More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from D based on the splitting of A.

Defining Information Gain

- Our objective is to take A on splitting to produce an exact classification (also called pure), that is, all tuples belong to one class.
- However, it is quite likely that the partitions is impure, that is, they contain tuples from two or more classes.
- In that sense, $E_A(D)$ is a measure of impurities (or purity). A lesser value of $E_A(D)$ implying more power the partitions are.

Definition 9.5: Information Gain

Information gain, $\alpha(A, D)$ of the training set D splitting on the attribute A is given by

$$\alpha(A, D) = E(D) - E_A(D)$$

In other words, $\alpha(A, D)$ gives us an estimation how much would be gained by splitting on A. The attribute A with the highest value of α should be chosen as the splitting attribute for D.

Information Gain Calculation

Information gain on splitting OPTH

- Let us refer to the OPTH database discussed in Slide #25.
- Splitting on **Age** at the root level, it would give three subsets D_1, D_2 and D_3 as shown in the tables in the following three slides.
- The entropy $E(D_1), E(D_2)$ and $E(D_3)$ of training sets D_1, D_2 and D_3 and corresponding weighted entropy $E_{Age}(D_1)$, $E_{Age}(D_2)$ and $E_{Age}(D_3)$ are also shown alongside.
- The Information gain $\alpha\ (Age, OPTH)$ is then can be calculated as **0.0394**.
- Recall that entropy of OPTH data set, we have calculated as *E(OPTH)* = **1.3261**
  (see Slide #25)

Information Gain Calculation

Information gain on splitting OPTH

Training set: $D_1(\text{Age} = 1)$

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

$$E(D_1) = -\frac{2}{8}log_2(\frac{2}{8}) - \frac{2}{8}log_2(\frac{2}{8}) - \frac{4}{8}log_2(\frac{4}{8}) = \mathbf{1.5}$$

$$E_{Age}(D_1) = \frac{8}{24} \times 1.5 = \mathbf{0.5000}$$

Calculating Information Gain

Training set: $D_2(\text{Age} = 2)$

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

$$E(D_2) = -\frac{1}{8}log_2(\frac{1}{8}) - \frac{2}{8}log_2(\frac{2}{8}) - \frac{5}{8}log_2(\frac{5}{8})$$
$$= 1.2988$$

$$E_{Age}(D_2) = \frac{8}{24} \times 1.2988 = 0.4329$$

Calculating Information Gain

Training set: $D_3(\text{Age} = 3)$

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

$$E(D_3) = -\frac{1}{8}log_2(\frac{1}{8}) - \frac{1}{8}log_2(\frac{1}{8}) - \frac{6}{8}log_2(\frac{6}{8}) = \mathbf{1.0613}$$

$$E_{Age}(D_3) = \frac{8}{24} \times 1.0613 = \mathbf{0.3504}$$

$$\alpha\,(Age, D) = 1.3261 - (0.5000 + 0.4329 + 0.3504) = \mathbf{0.0394}$$

Information Gains for Different Attributes

- In the same way, we can calculate the information gains, when splitting the OPTH database on Eye-sight, Astigmatic and Use Type. The results are summarized below.
- Splitting attribute: Age

$$\alpha(Age, OPTH) = 0.0394$$

- Splitting attribute: Eye-sight

$$\alpha(Eye - sight, OPTH) = 0.0395$$

- Splitting attribute: Astigmatic

$$\alpha(\text{Astigmatic}, OPTH) = 0.3770$$

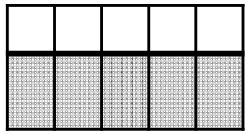
- Splitting attribute: Use Type

$$\alpha(\text{Use Type}, OPTH) = 0.5488$$

Decision Tree Induction : ID3 Way

- The ID3 strategy of attribute selection is to choose to split on the attribute that gives the greatest reduction in the weighted average entropy
  - The one that maximizes the value of information gain
- In the example with OPTH database, the larger values of information gain is $\alpha(\text{Use Type}, OPTH) = 0.5488$
  - Hence, the attribute should be chosen for splitting is "Use Type".
- The process of splitting on nodes is repeated for each branch of the evolving decision tree, and the final tree, which would look like is shown in the following slide and calculation is left for practice.

Decision Tree Induction : ID3 Way



$E(OPTH) = 1.3261$

$OPTH$

Age ✗ $\alpha = 0.0394$

Eye-sight ✗ $\alpha = 0.395$

Use Type ✓ $\alpha = 0.5488$

Astigmatic ✗ $\alpha = 0.3770$

$D1$ $Frequent(1)$

$E(D_1) = ?$

Age	Eye	Ast	Use	Class
1	1	1	1	3
1	1	2	1	3
1	2	1	1	3
1	2	2	1	3
2	1	1	1	3
2	2	1	1	3
2	2	2	1	3
3	1	1	1	3
3	1	2	1	3
3	2	1	1	3
3	2	2	1	3

? Age $\alpha = ?$

? Eye-sight $\alpha = ?$

? Astigmatic $\alpha = ?$

$D2$ $Less(2)$

$E(D_2) = ?$

Age	Eye	Ast	Use	Class
1	1	1	2	2
1	1	2	2	1
1	2	1	2	2
1	2	2	2	1
2	1	1	2	2
2	1	2	2	1
2	2	1	2	2
3	1	1	2	3
3	1	2	2	3
3	2	1	2	2
3	2	2	2	3

? Age $\alpha = ?$

? Eye-sight $\alpha = ?$

? Astigmatic $\alpha = ?$

Limiting Values of Information Gain

- The Information gain metric used in ID3 always should be positive or zero.
- It is always positive value because information is always gained (i.e., purity is improved) by splitting on an attribute.
- On the other hand, when a training set is such that if there are k classes, and the entropy of training set takes the largest value i.e., $\log_2 k$ (this occurs when the classes are balanced), then the information gain will be zero.

Splitting of Continuous Attribute Values

- In the foregoing discussion, we assumed that an attribute to be splitted is with a finite number of discrete values. Now, there is a great deal if the attribute is not so, rather it is a continuous-valued attribute.
- There are two approaches mainly to deal with such a case.

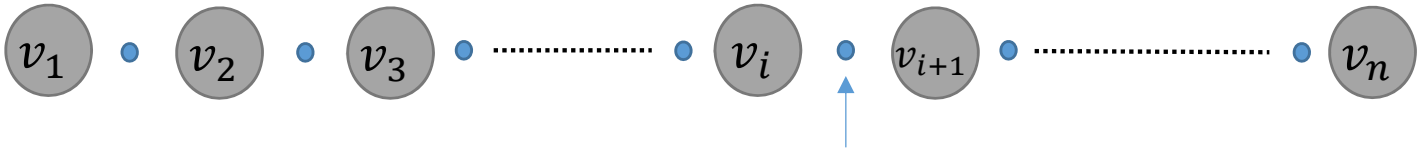
1. **Data Discretization**: All values of the attribute can be discretized into a finite number of group values and then split point can be decided at each boundary point of the groups.

So, if there are $n - groups$ of discrete values, then we have $(n + 1)$ split points.

Splitting of Continuous attribute values

2. Mid-point splitting: Another approach to avoid the data discretization.

- It sorts the values of the attribute and take the distinct values only in it.
- Then, the mid-point between each pair of adjacent values is considered as a split-point.



- Here, if n-distinct values are there for the attribute A, then we choose $n-1$ split points as shown above.
- For example, there is a split point $s = \frac{v_i + v_{(i+1)}}{2}$ in between v_i and $v_{(i+1)}$
- For each split-point, we have two partitions: $A \leq s$ *and* $A > s$, and finally the point with maximum information gain is the desired split point for that attribute.

Algorithm CART

CART Algorithm

- It is observed that information gain measure used in ID3 is biased towards test with many outcomes, that is, it prefers to select attributes having a large number of values.

- L. Breiman, J. Friedman, R. Olshen and C. Stone in 1984 proposed an algorithm to build a binary decision tree also called CART decision tree.
  - CART stands for **Classification and Regression Tree**
  - In fact, invented independently at the same time as ID3 (1984).
  - ID3 and CART are two cornerstone algorithms spawned a flurry of work on decision tree induction.

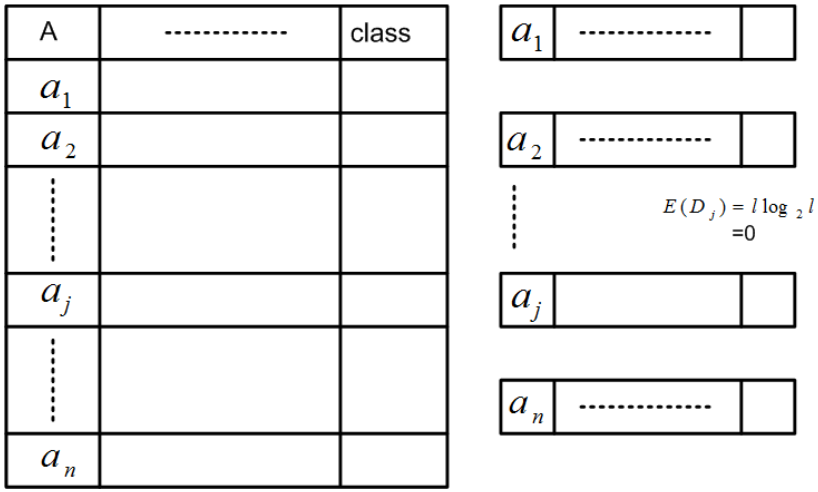
- CART is a technique that generates a **binary decision tree;** That is, unlike ID3, in CART, for each node only two children is created.

- ID3 uses Information gain as a measure to select the best attribute to be splitted, whereas CART do the same but using another measurement called **Gini index** . It is also known as **Gini Index of Diversity** and is denote as γ.

Algorithm C4.5 : Introduction

Example : Limitation of ID3

In the following, each tuple belongs to a unique class. The splitting on A is shown.



$$E_A(D) = \sum_{j=1}^{n} \frac{|D_j|}{|D|} . E(D_j) = \sum_{j=1}^{n} \frac{1}{|D|} . 0 = 0$$

Thus, $\alpha(A, D) = E(D) - E_A(D)$ is maximum in such a situation.

Gini Index of Diversity

Definition : **Gini Index**

Suppose, D is a training set with size $|D|$ and $C = \{c_1, c_2, \dots, c_k\}$ be the set of k classifications and $A = \{a_1, a_2, \dots, a_m\}$ be any attribute with m different values of it. Like entropy measure in ID3, CART proposes Gini Index (denoted by G) as the measure of impurity of D. It can be defined as follows.

$$G(D) = 1 - \sum_{i=1}^{k} p_i^2$$

where p_i is the probability that a tuple in D belongs to class c_i and p_i can be estimated as

$$p_i = \frac{|C_{i,D}|}{D}$$

where $|C_{i,D}|$ denotes the number of tuples in D with class c_i.

Gini Index of Diversity

Note

- $G(D)$ measures the "impurity" of data set D.
- The smallest value of $G(D)$ is zero
  - which it takes when all the classifications are same.
- It takes its largest value $= 1 - \frac{1}{k}$
  - when the classes are evenly distributed between the tuples, that is the frequency of each class is $\frac{1}{k}$.

Gini Index of Diversity

Definition : **Gini Index of Diversity**

Suppose, a binary partition on A splits D into D_1 and D_2, then the weighted average Gini Index of splitting denoted by $G_A(D)$ is given by

$$G_A(D) = \frac{|D_1|}{D}.G(D_1) + \frac{|D_2|}{D}.G(D_2)$$

This binary partition of D reduces the impurity and the reduction in impurity is measured by

$$\gamma(A, D) = G(D) - G_A(D)$$

Gini Index of Diversity and CART

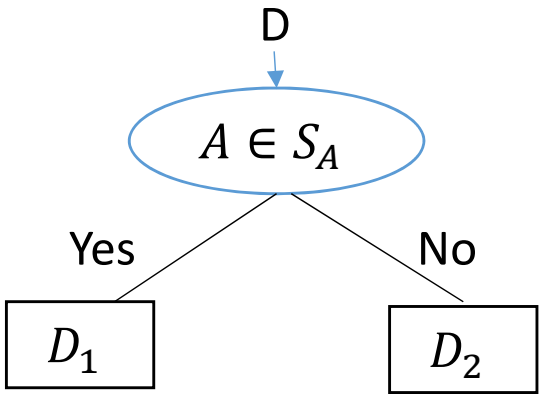
- This $\gamma(A, D)$ is called the Gini Index of diversity.
- It is also called as "impurity reduction".
- The attribute that maximizes the reduction in impurity (or equivalently, has the minimum value of $G_A(D)$) is selected for the attribute to be splitted.

n-ary Attribute Values to Binary Splitting

- The CART algorithm considers a binary split for each attribute.
- We shall discuss how the same is possible for attribute with more than two values.
- **Case 1: Discrete valued attributes**
- Let us consider the case where A is a discrete-valued attribute having m discrete values $a_1, a_2, \ldots, a_m$.
- To determine the best binary split on A, we examine all of the possible subsets say 2^A of A that can be formed using the values of A.
- Each subset $S_A \in 2^A$ can be considered as a binary test for attribute A of the form "$A \in S_A$?".

n-ary Attribute Values to Binary Splitting

- Thus, given a data set *D*, we have to perform a test for an attribute value *A* like



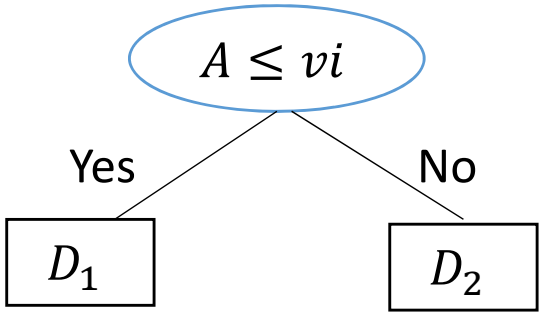
- This test is satisfied if the value of *A* for the tuples is among the values listed in S_A.
- If *A* has *m* distinct values in *D*, then there are 2^m possible subsets, out of which the empty subset { } and the power set $\{a_1, a_2, \ldots, a_n\}$ should be excluded (as they really do not represent a split).
- Thus, there are $2^{m-1} - 1$ possible ways to form two partitions of the dataset *D*, based on the binary split of *A*.

n-ary Attribute Values to Binary Splitting

Case2: Continuous valued attributes

- For a continuous-valued attribute, each possible split point must be taken into account.
- The strategy is similar to that followed in ID3 to calculate information gain for the continuous –valued attributes.
- According to that strategy, the mid-point between a_i and a_{i+1} , let it be v_i, then

$$v_i = \frac{a_i + a_{i+1}}{2}$$



n-ary Attribute Values to Binary Splitting

- Each pair of (sorted) adjacent values is taken as a possible split-point say v_i.
- D_1 is the set of tuples in D satisfying $A \leq v_i$ and D_2 in the set of tuples in D satisfying $A > v_i$.
- The point giving the minimum Gini Index $G_A(D)$ is taken as the split-point of the attribute A.

Note

- The attribute A and either its splitting subset S_A (for discrete-valued splitting attribute) or split-point v_i (for continuous valued splitting attribute) together form the splitting criteria.

CART Algorithm : Illustration

CART Algorithm

Suppose we want to build decision tree for the data set EMP as given in the table below.

Age
Y : young
M : middle-aged
O : old

Salary
L : low
M : medium
H : high

Job
G : government
P : private

Performance
A : Average
E : Excellent

Class : Select
Y : yes
N : no

Tuple#	Age	Salary	Job	Performance	Select
1	Y	H	P	A	N
2	Y	H	P	E	N
3	M	H	P	A	Y
4	O	M	P	A	Y
5	O	L	G	A	Y
6	O	L	G	E	N
7	M	L	G	E	Y
8	Y	M	P	A	N
9	Y	L	G	A	Y
10	O	M	G	A	Y
11	Y	M	G	E	Y
12	M	M	P	E	Y
13	M	H	G	A	Y
14	O	M	P	E	N

CART Algorithm : Illustration

For the EMP data set,

$$G(EMP) = 1 - \sum_{i=1}^{2} p_i^2$$

$$= 1 - \left[\left(\frac{9}{14}\right)^2 + \left(\frac{5}{14}\right)^2\right]$$

$$= \mathbf{0.4592}$$

Now let us consider the calculation of $G_A(EMP)$ for **Age**, **Salary**, **Job** and **Performance**.

CART Algorithm : Illustration

Attribute of splitting: Age

The attribute age has three values, namely Y, M and O. So there are 6 subsets, that should be considered for splitting as:

$$\underset{age_1}{\{Y\}}, \underset{age_2}{\{M\}}, \underset{age_3}{\{O\}}, \underset{age_4}{\{Y,M\}}, \underset{age_5}{\{Y,O\}}, \underset{age_6}{\{M,O\}}$$

$$G_{age_1}(D) = \frac{5}{14} * \left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) + \frac{9}{14}\left(1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2\right) = \mathbf{0.3937}$$

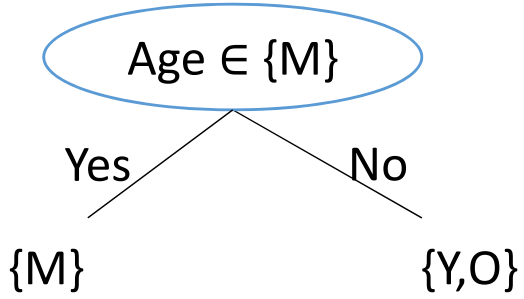
$G_{age_2}(D) = \mathbf{0.3571}$

$G_{age_3}(D) = \mathbf{0.4571}$

$G_{age_4}(D) = G_{age_3}(D)$

$G_{age_5}(D) = G_{age_2}(D)$

$G_{age_6}(D) = G_{age_1}(D)$



The best value of Gini Index while splitting attribute Age is $\gamma(Age_2, D) = \mathbf{0.1021}$

CART Algorithm : Illustration

Attribute of Splitting: Salary

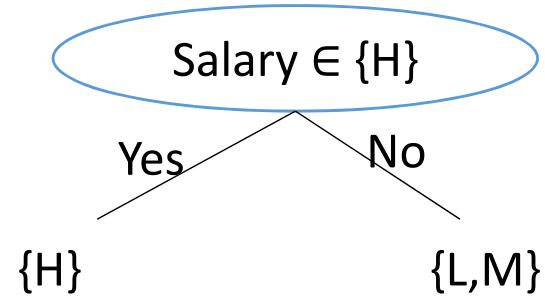
The attribute salary has three values namely L, M and H. So, there are 6 subsets, that should be considered for splitting as:

$$\underset{sal_1}{\{L\}}, \quad \underset{sal_2}{\{M,H\}}, \quad \underset{sal_3}{\{M\}}, \quad \underset{sal_4}{\{L,H\}}, \quad \underset{sal_5}{\{H\}}, \quad \underset{sal_6}{\{L,M\}}$$

$G_{sal_1}(D) = G_{sal_2}(D) = 0.45$

$G_{sal_3}(D) = G_{sal_4}(D) = 0.4583$

$G_{sal_5}(D) = G_{sal_6}(D) = 0.4429$



$$\gamma(salary_{(5,6),} D) = 0.4592 - 0.4429 = 0.0163$$

CART Algorithm : Illustration

Attribute of Splitting: job

Job being the binary attribute , we have

$$G_{job}(D) = \frac{7}{14} G(D_1) + \frac{7}{14} G(D_2)$$

$$= \frac{7}{14}\left[1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2\right] + \frac{7}{14}\left[1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2\right] = \mathbf{0.3673}$$

$$\gamma(job, D) = \mathbf{0.0919}$$

CART Algorithm : Illustration

Attribute of Splitting: Performance

Job being the binary attribute , we have

$G_{Performance}(D) =$ **0.4286**

$\gamma(performance, D) =$ **0.0306**

Out of these $\gamma(Age_2, D)$ gives the maximum value and hence, the attribute Age_2 would be chosen for splitting subset $\{M\}$ or $\{Y, O\}$.

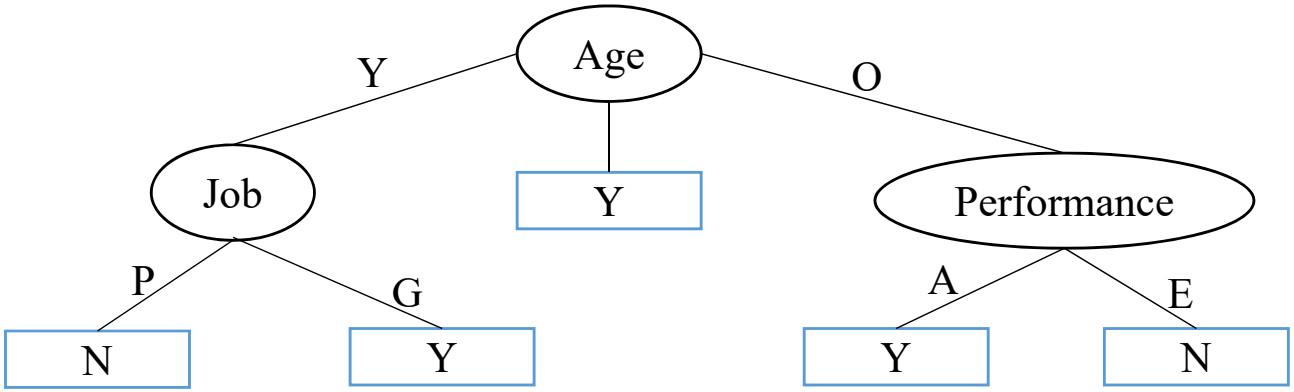
Note:

It can be noted that the procedure following "information gain" calculation (i.e. $\propto (A, D)$) and that of "impurity reduction" calculation (i.e. $\gamma(A, D)$) are near about.

Decision Trees with ID3 and CART Algorithms

Comparing Decision Trees of EMP Data set

Compare two decision trees obtained using ID3 and CART for the EMP dataset. The decision tree according to ID3 is given for your ready reference (subject to the verification)



Decision Tree using ID3

?

Decision Tree using CART

Table 11.6

Algorithm	Splitting Criteria	Remark
ID3	**Information Gain** $\alpha(A, D) = E(D) - E_A(\mathrm{D})$ Where $E(D)$ = Entropy of D (a measure of uncertainty) = $-\sum_{i=1}^{k} p_i \log_2 p_i$ where D is with set of k classes C_1, C_2, … , C_k and $p_i = \frac{\lvert C_{i,D} \rvert}{\lvert D \rvert}$; Here, $C_{i,D}$ is the set of tuples with class C_i in D. E_A (D) = Weighted average entropy when D is partitioned on the values of attribute A = $\sum_{j=1}^{m} \frac{\lvert D_j \rvert}{\lvert D \rvert} E(D_j)$ Here, m denotes the distinct values of attribute A.	• The algorithm calculates $\alpha(A_i, D)$ for all A_i *in* D and choose that attribute which has **maximum** $\alpha(A_i, D)$. • The algorithm can handle both categorical and numerical attributes. • It favors splitting those attributes, which has a large number of distinct values.

Algorithm	Splitting Criteria	Remark
CART	**Gini Index** $\gamma(A, D) = G(D) - G_A(\mathrm{D})$ where $G(D)$ = Gini index (a measure of impurity) $= 1 - \sum_{i=1}^{k} {p_i}^2$ Here, $p_i = \frac{\lvert C_{i,D} \rvert}{\lvert D \rvert}$ and D is with k number of classes and $G_A(D) = \frac{\lvert D_1 \rvert}{\lvert D \rvert} G(D_1) + \frac{\lvert D_2 \rvert}{\lvert D \rvert} G(D_2)$, when D is partitioned into two data sets D_1 and D_2 based on some values of attribute A.	• The algorithm calculates all binary partitions for all possible values of attribute A and choose that binary partition which has the **maximum** $\gamma(A, D)$. • The algorithm is computationally very expensive when the attribute A has a large number of values.

Notes on Decision Tree Induction algorithms

1. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.

2. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.

3. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of O(d), where d is the maximum depth of the tree.

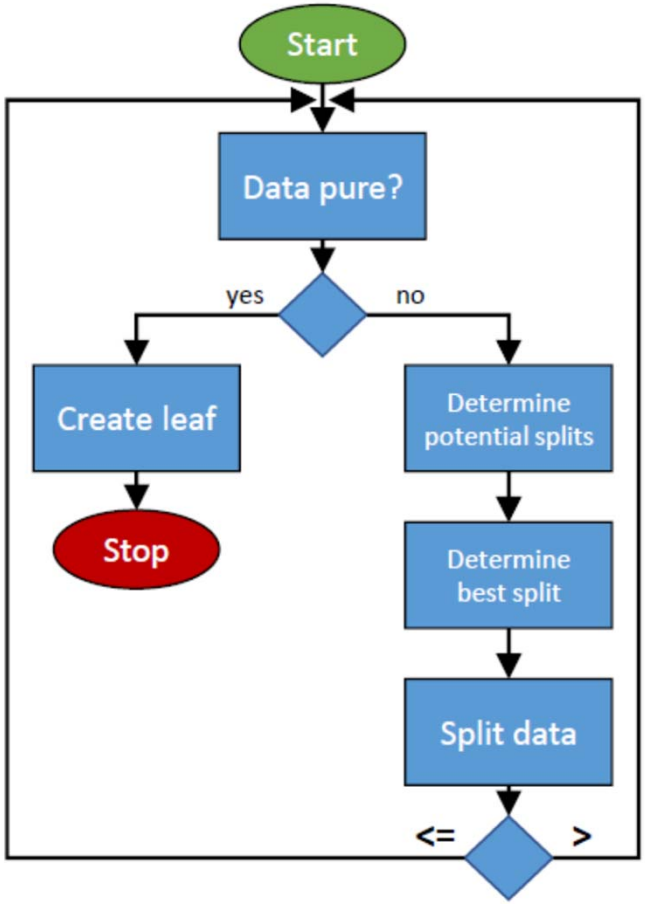
Notes on Decision Tree Induction algorithms

4. Data Fragmentation Problem: Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.

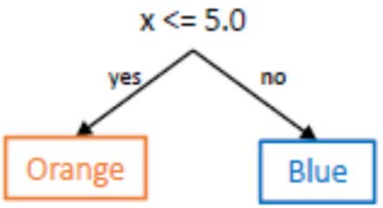
Tree Pruning

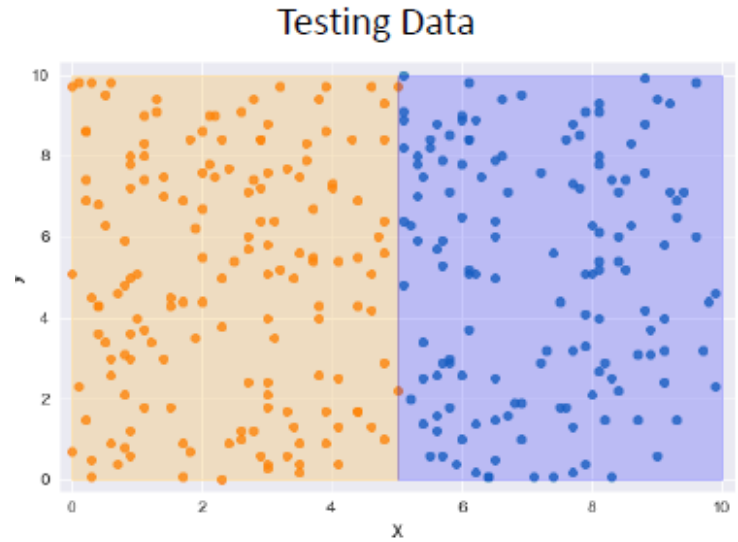
Pre-prunning: procedures prevent a complete induction of the training set by replacing a stop () criterion in the induction algorithm (e.g. max. Tree depth or information gain (Attr)> minGain).

Post-pruning (or just pruning) is the most common way of simplifying trees. Here, nodes and subtrees are replaced with leaves to reduce complexity. Pruning can not only significantly reduce the size but also improve the classification accuracy of unseen objects.

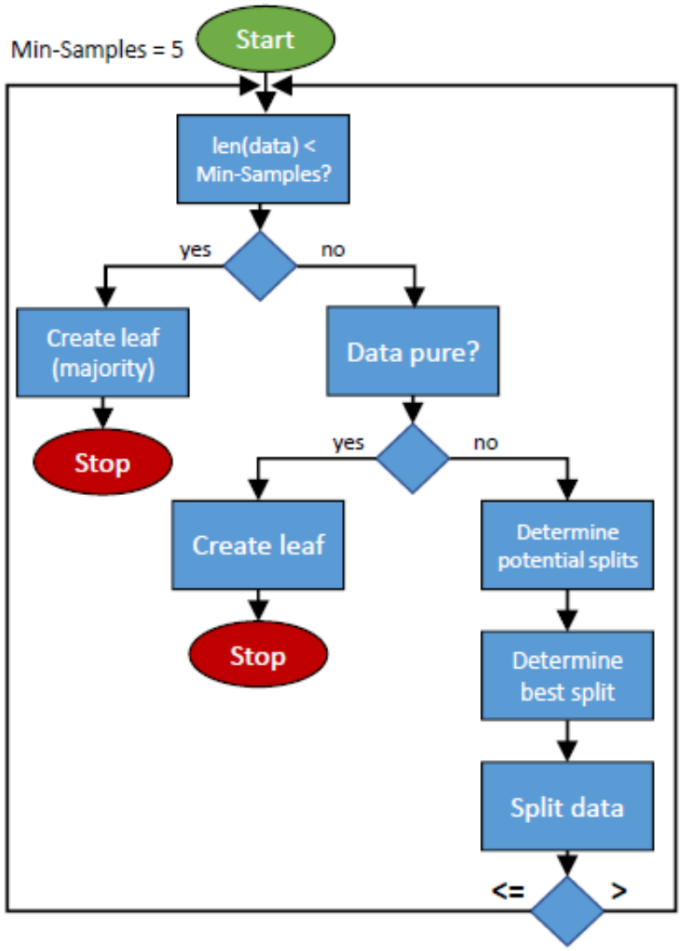

Start
Data pure?
yes
no
Create leaf
Stop
Determine potential splits
Determine best split
Split data
<=
>

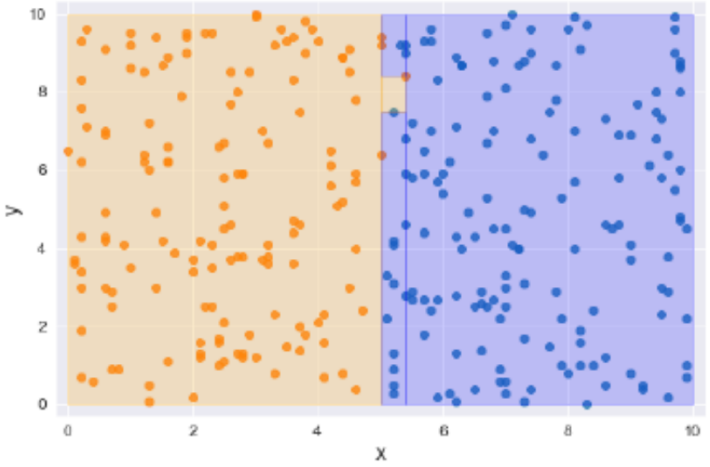

Training Data
y
X
0
2
4
6
8
10

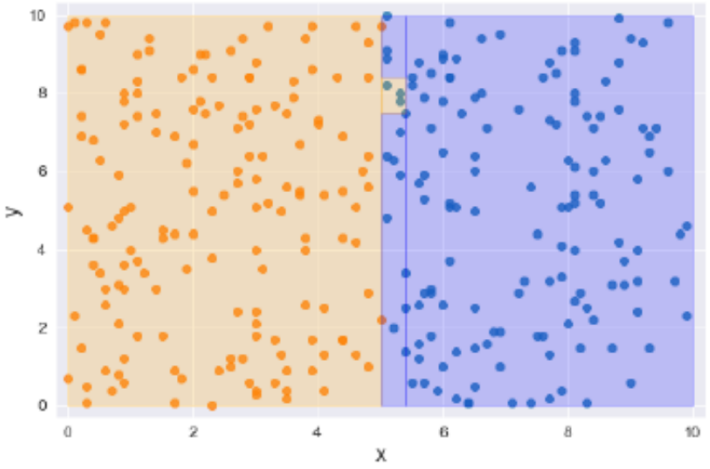

x <= 5.0
yes
no
Orange
Blue

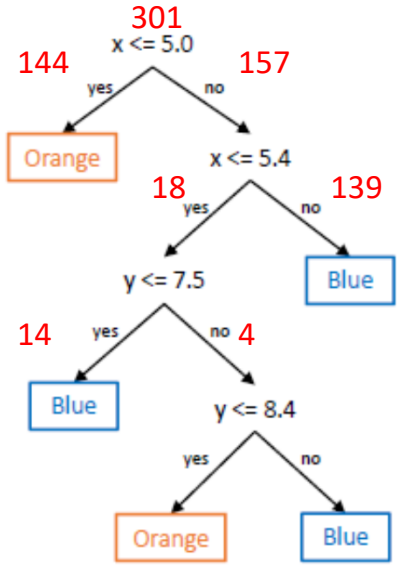

Testing Data
y
X
0
2
4
6
8
10

Pre-Pruning

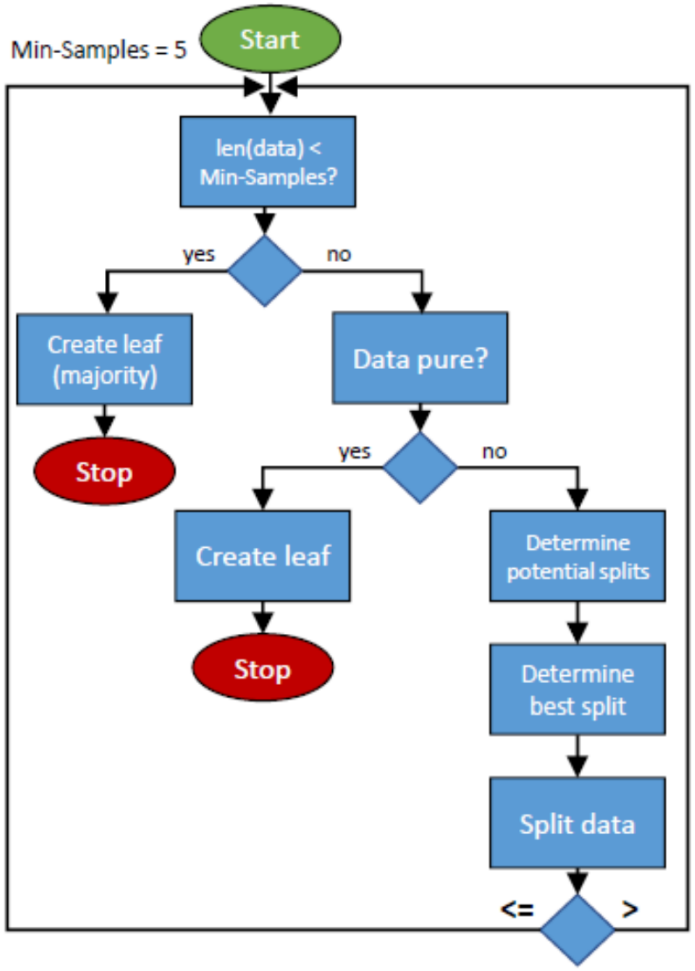

Min-Samples = 5
Start
len(data) < Min-Samples?
yes
no
Create leaf (majority)
Stop
Data pure?
yes
no
Create leaf
Stop
Determine potential splits
Determine best split
Split data
<=
>

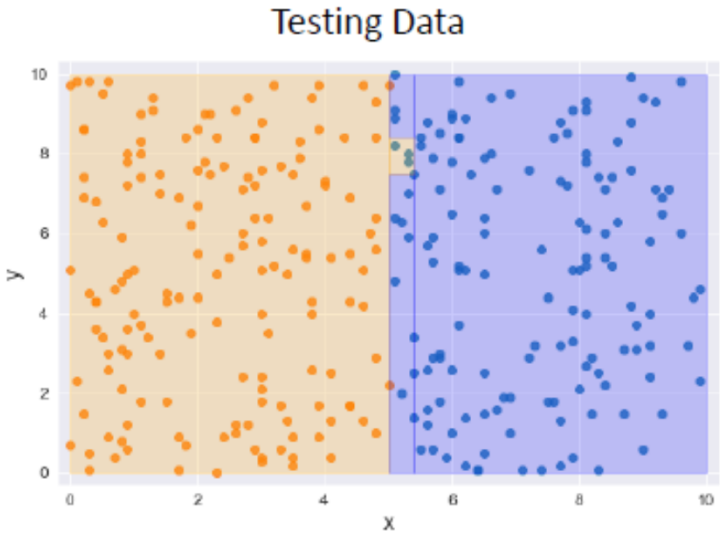

Training Data
y
X

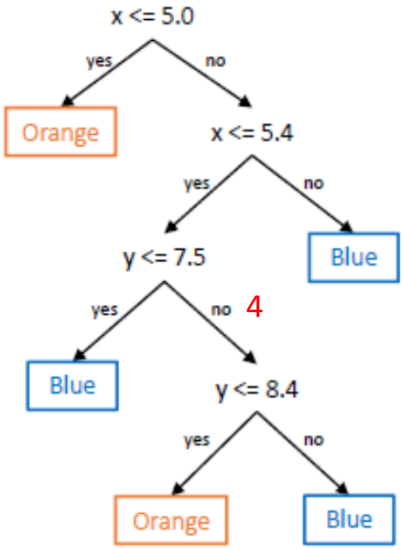

Testing Data
y
X


301
x <= 5.0
144
157
yes
no
Orange
x <= 5.4
18
139
yes
no
y <= 7.5
Blue
14
yes
no
4
Blue
y <= 8.4
yes
no
Orange
Blue

Pre-Pruning

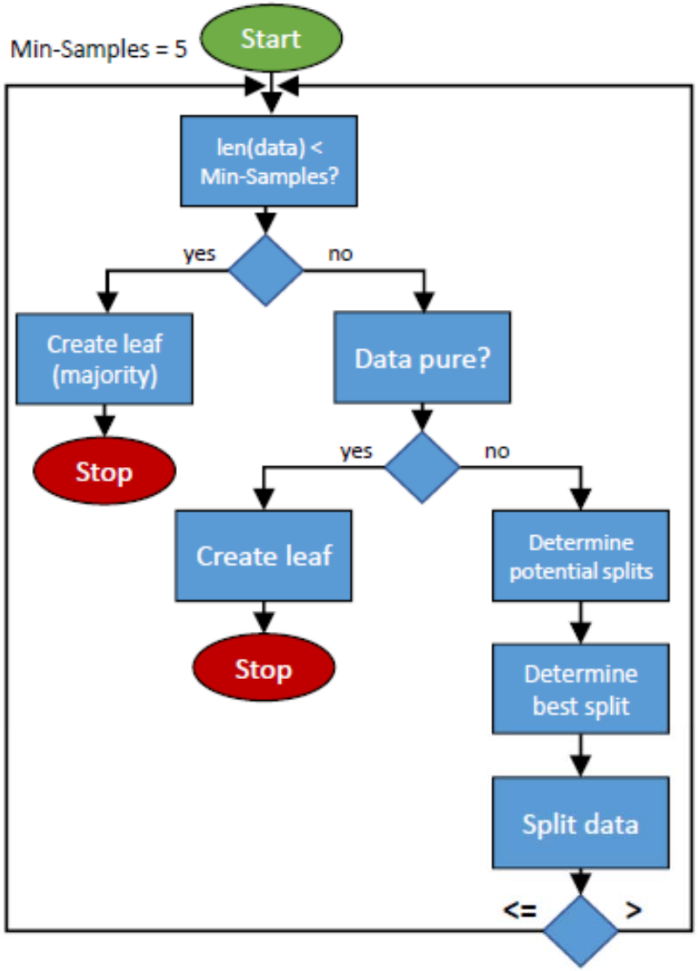


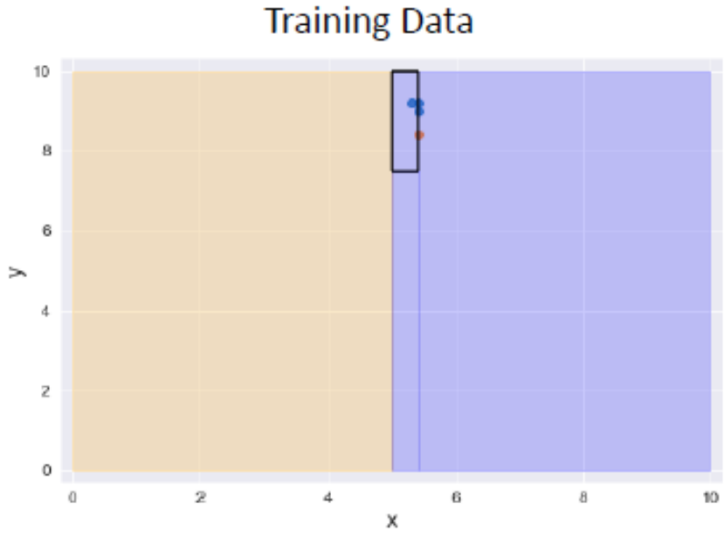


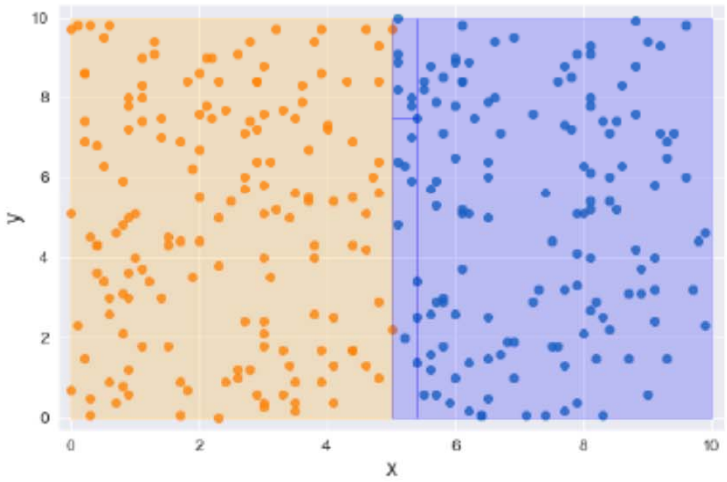


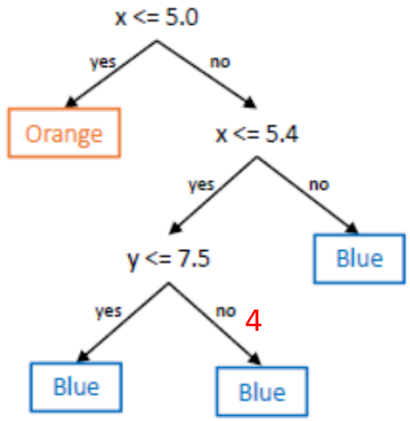
4

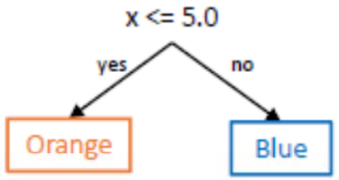
Pre-Pruning


Min-Samples = 5
Start
len(data) < Min-Samples?
yes
no
Create leaf (majority)
Stop
Data pure?
yes
no
Create leaf
Stop
Determine potential splits
Determine best split
Split data
<=
>

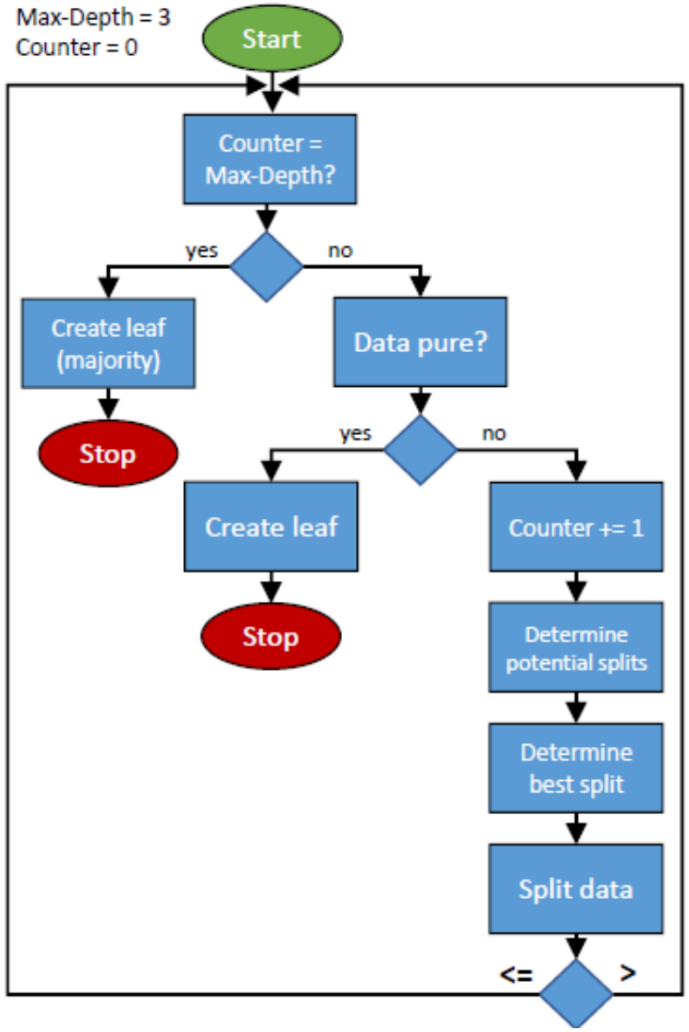

Training Data
y
x
0
2
4
6
8
10


Testing Data
y
x
0
2
4
6
8
10

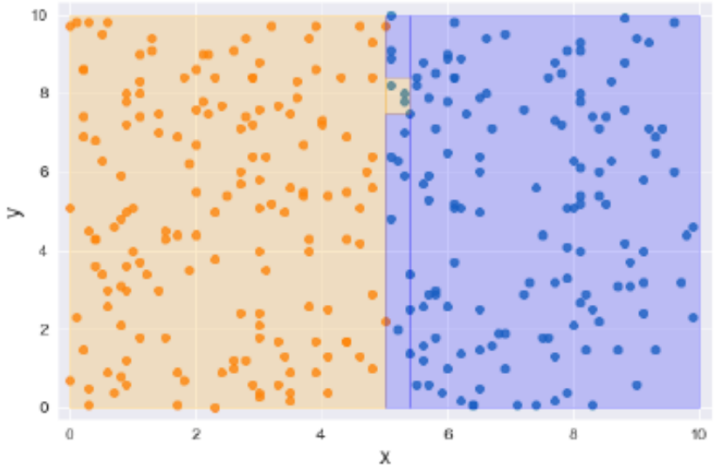

x <= 5.0
yes
no
Orange
x <= 5.4
yes
no
y <= 7.5
Blue
yes
no
Blue
Blue

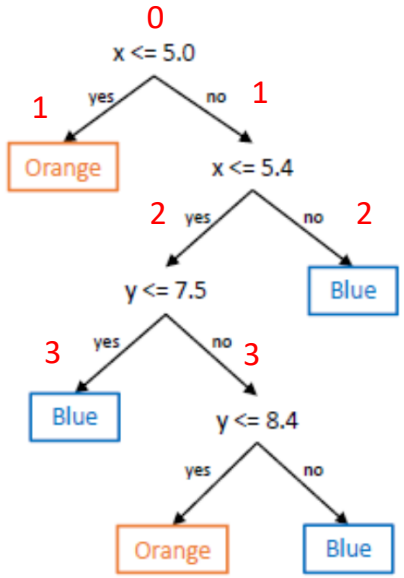

x <= 5.0
yes
no
Orange
Blue

Pre-Pruning

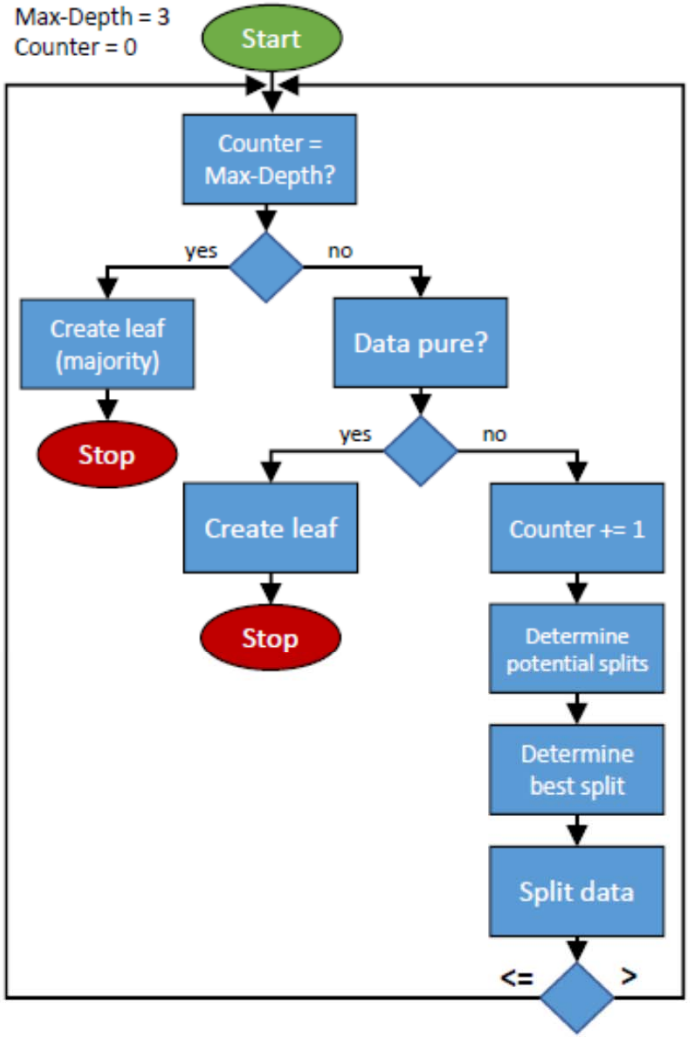

Max-Depth = 3
Counter = 0
Start
Counter = Max-Depth?
yes
no
Create leaf (majority)
Stop
Data pure?
yes
no
Create leaf
Stop
Counter += 1
Determine potential splits
Determine best split
Split data
<=
>

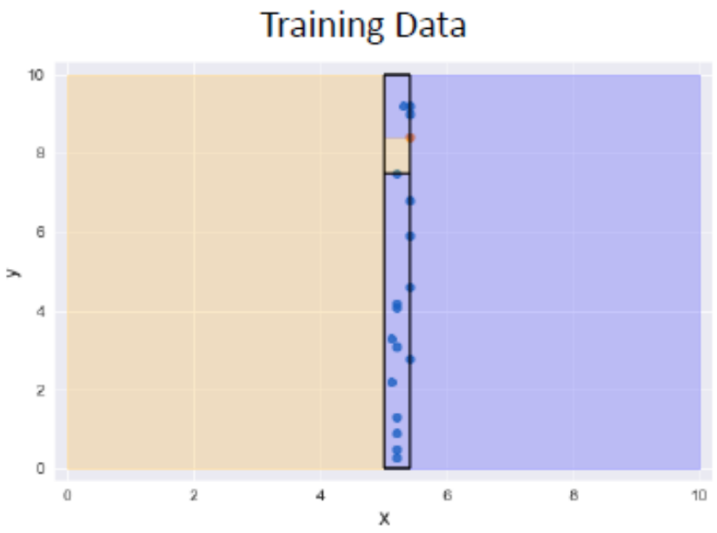

Training Data
y
x

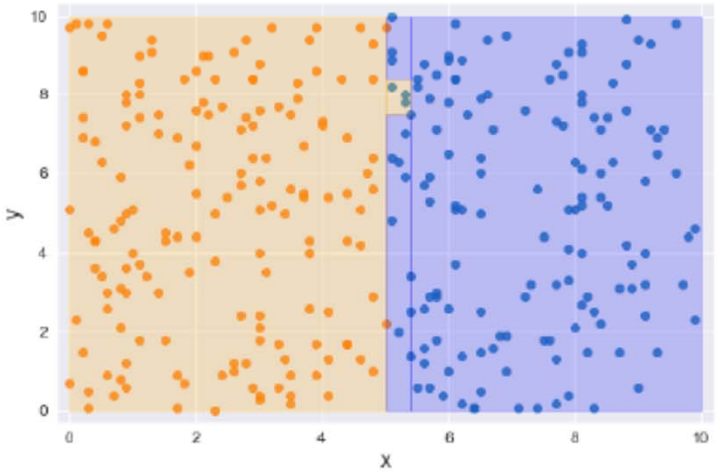

Testing Data
y
x

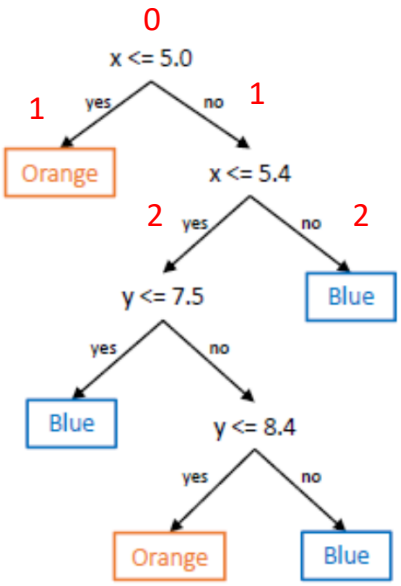

0
x <= 5.0
yes
no
1
1
Orange
x <= 5.4
2
yes
no
2
y <= 7.5
Blue
3
yes
no
3
Blue
y <= 8.4
yes
no
Orange
Blue

Pre-Pruning

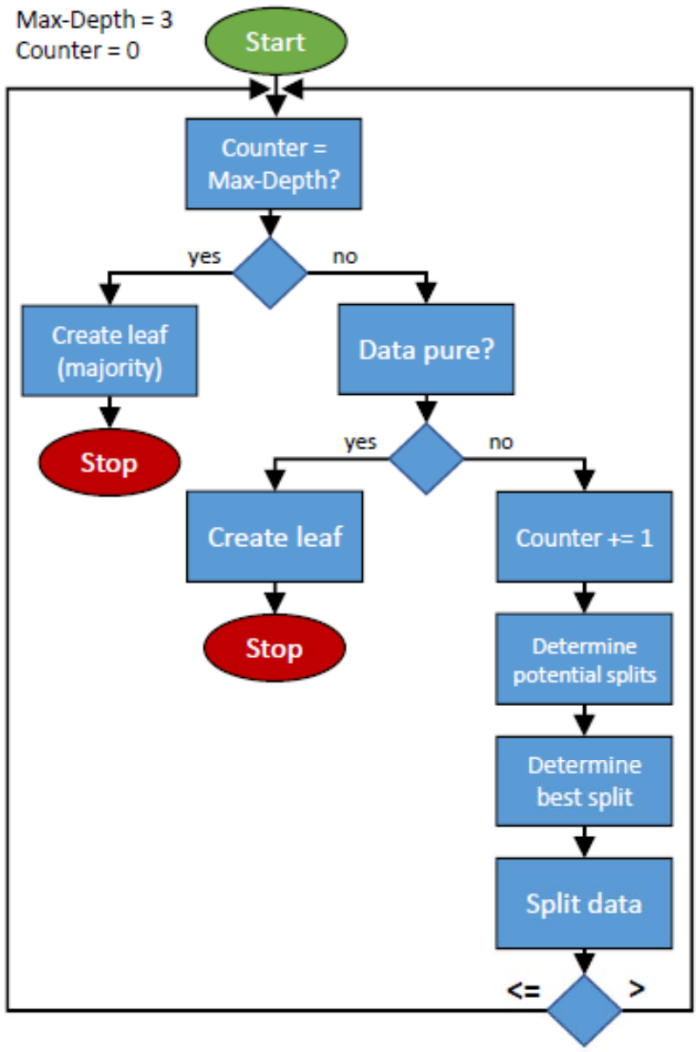

Max-Depth = 3
Counter = 0
Start
Counter = Max-Depth?
yes
no
Create leaf (majority)
Stop
Data pure?
yes
no
Create leaf
Stop
Counter += 1
Determine potential splits
Determine best split
Split data
<=
>

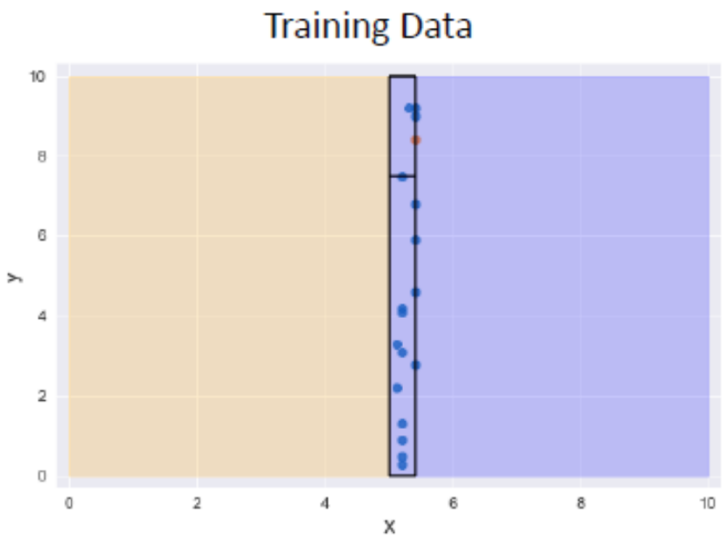

Training Data
y
x
0
2
4
6
8
10

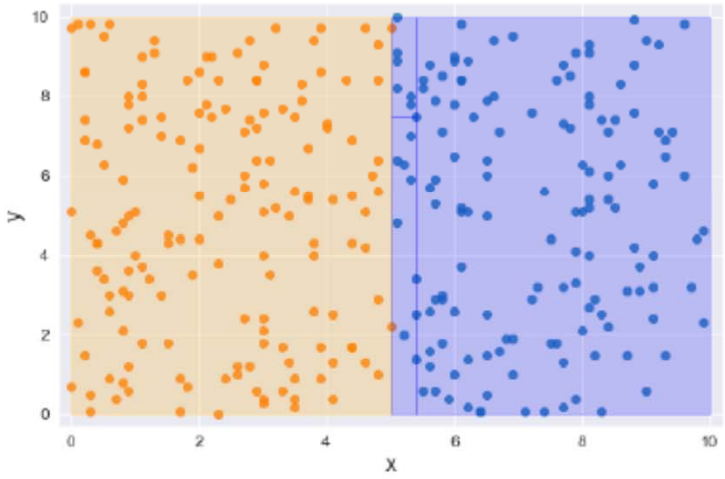

Testing Data
y
x
0
2
4
6
8
10

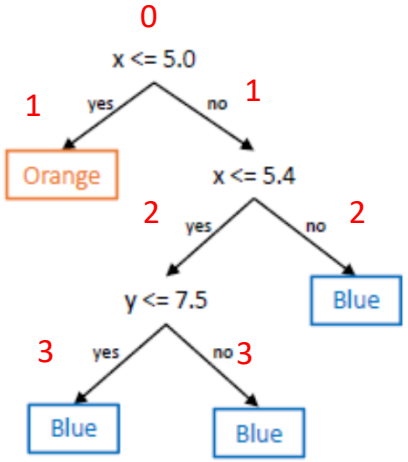

0
x <= 5.0
yes
no
1
1
Orange
x <= 5.4
2
yes
no
2
y <= 7.5
Blue
yes
no
Blue
y <= 8.4
yes
no
Orange
Blue

Pre-Pruning

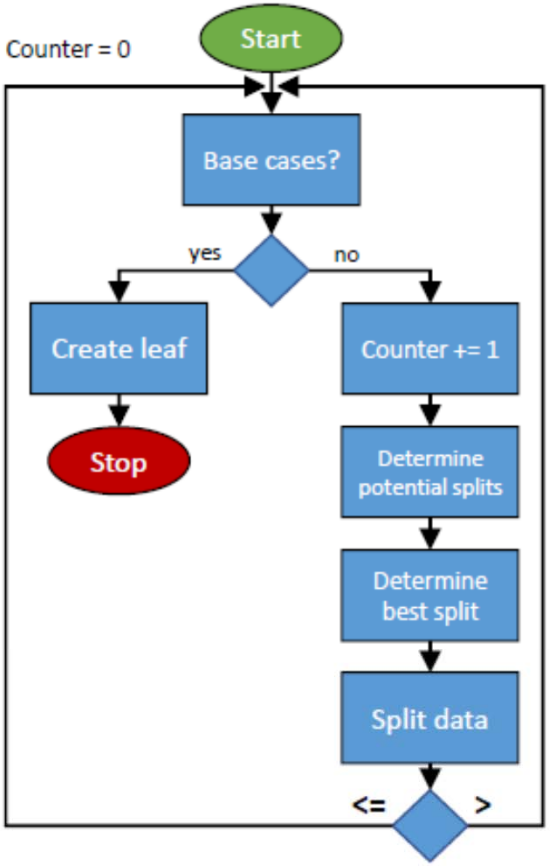

Max-Depth = 3
Counter = 0
Start
Counter = Max-Depth?
yes
no
Create leaf (majority)
Stop
Data pure?
yes
no
Create leaf
Stop
Counter += 1
Determine potential splits
Determine best split
Split data
<=
>

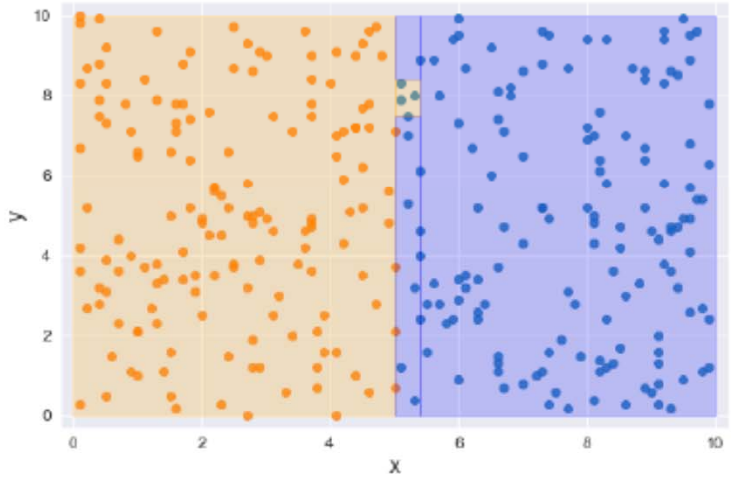

Training Data
y
X

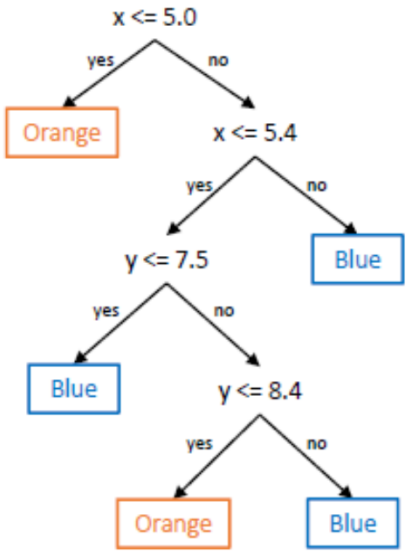

Testing Data
y
X


0
x <= 5.0
1
yes
no
1
Orange
x <= 5.4
2
yes
no
2
y <= 7.5
Blue
3
yes
no
3
Blue
Blue

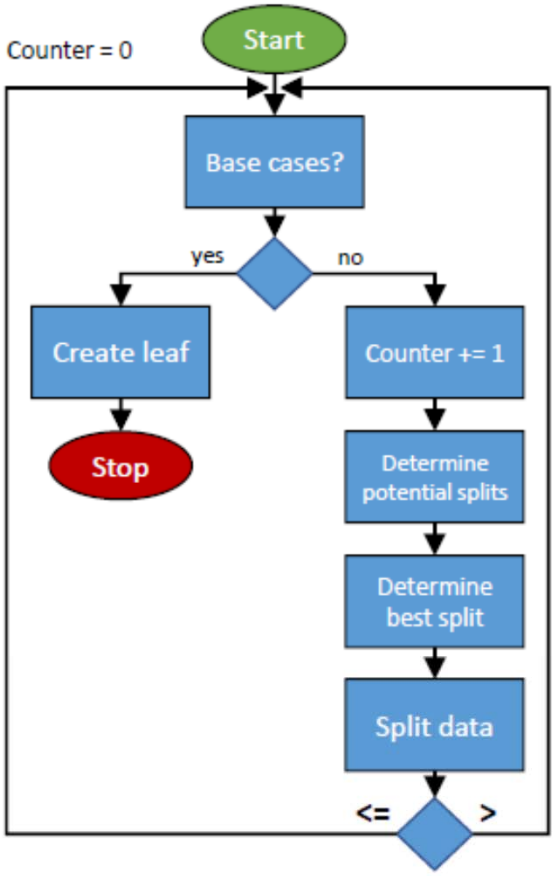
Post-Pruning

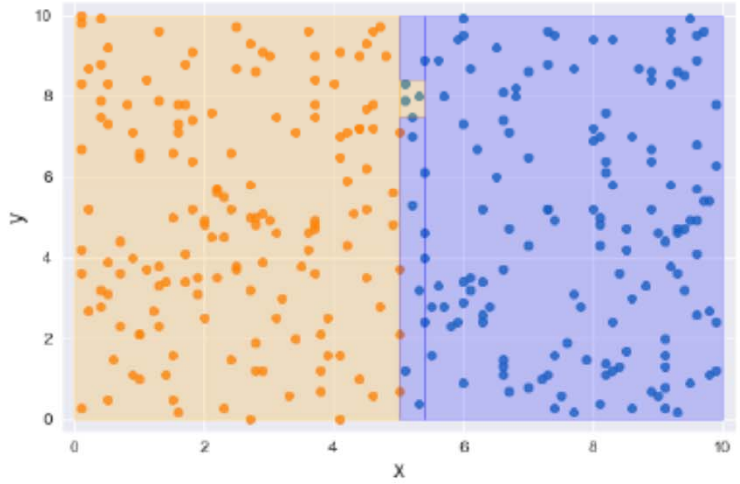


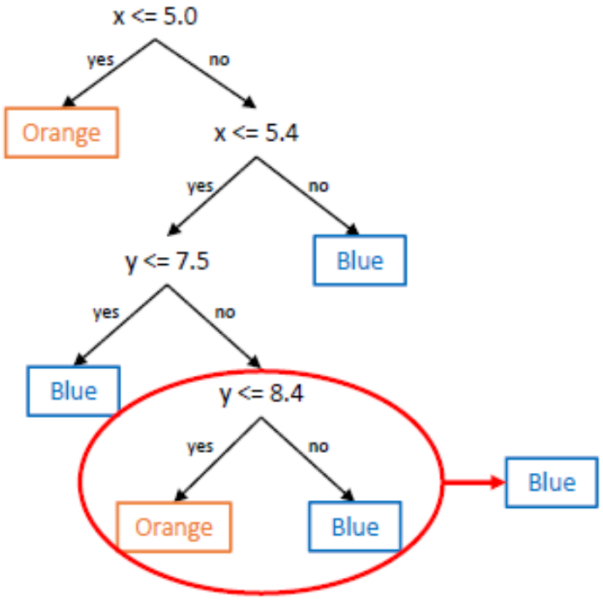




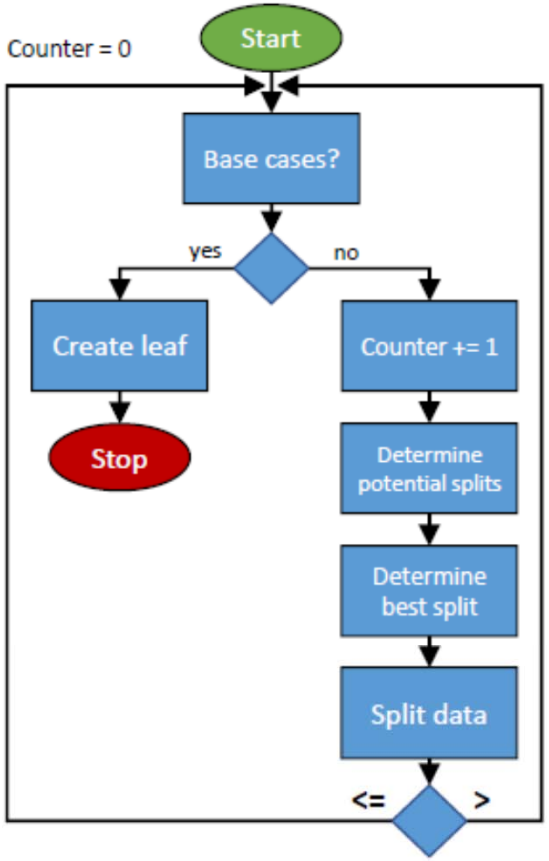
Post-Pruning

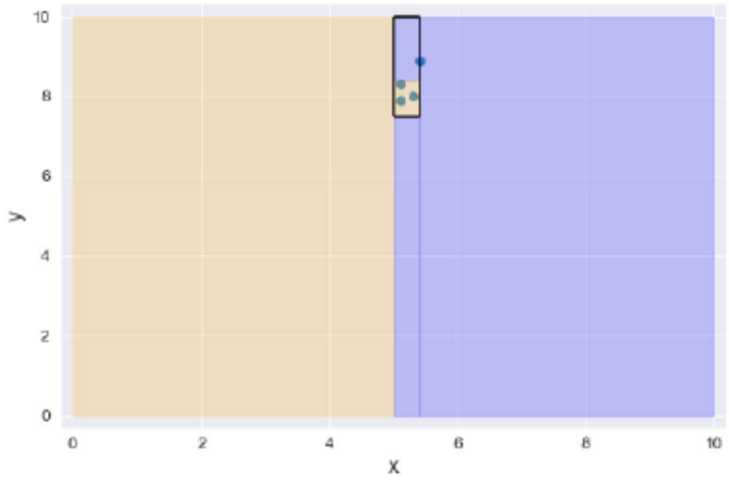


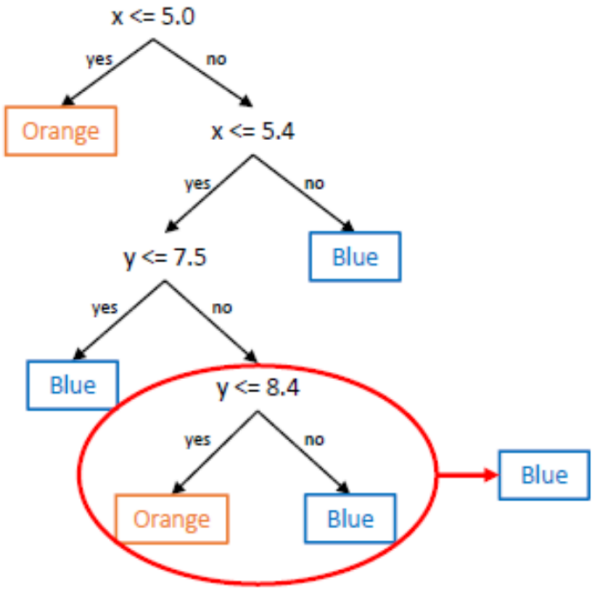




Post-Pruning

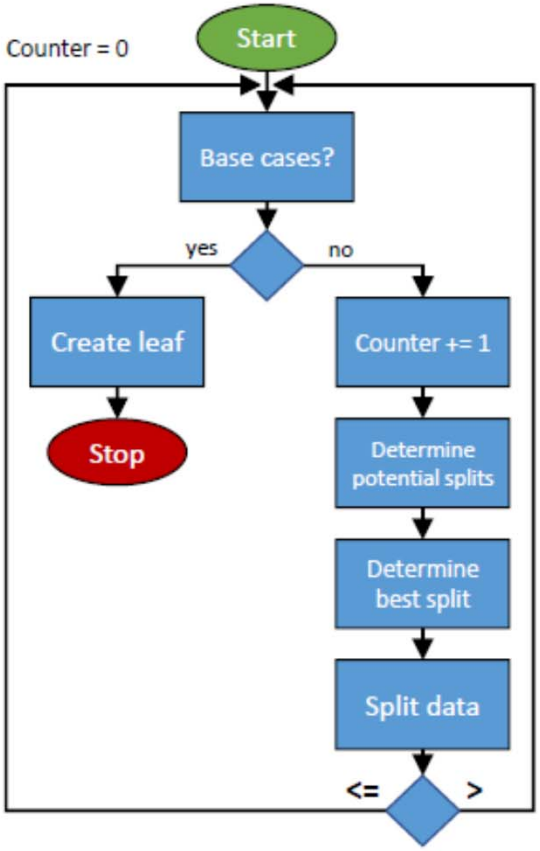


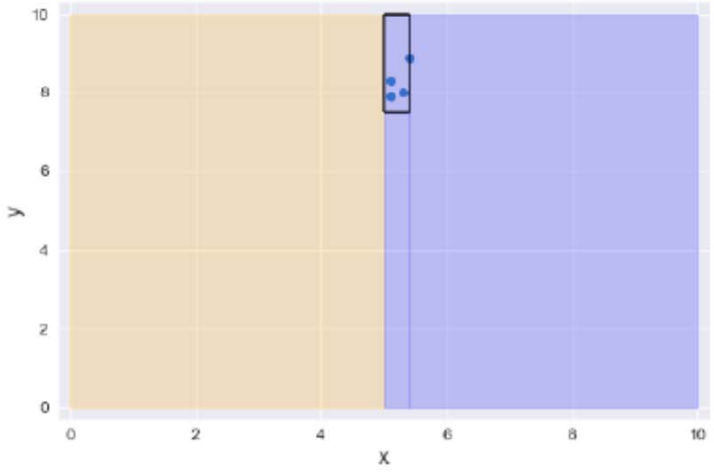


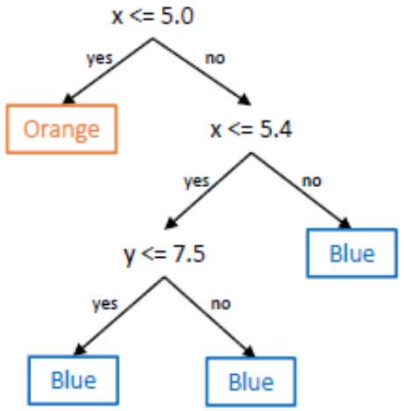


	Errors
Decision Node	3
Leaf	0

Post-Pruning

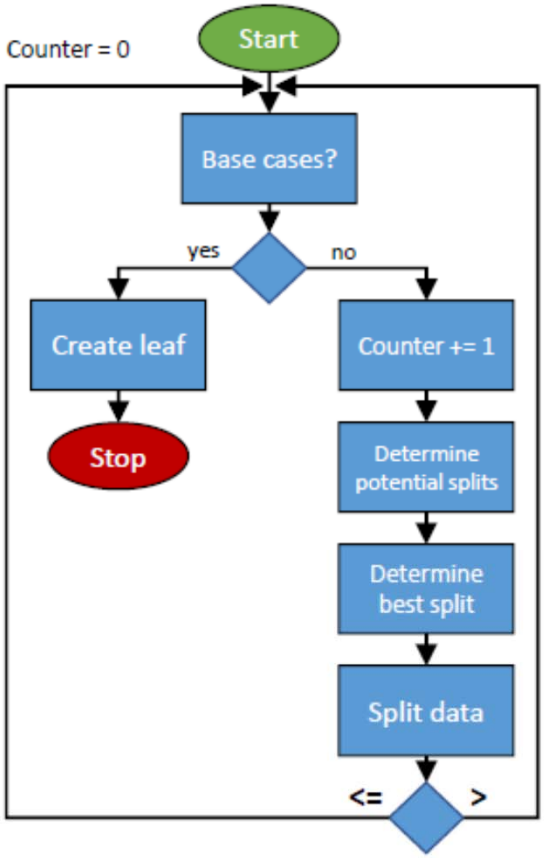


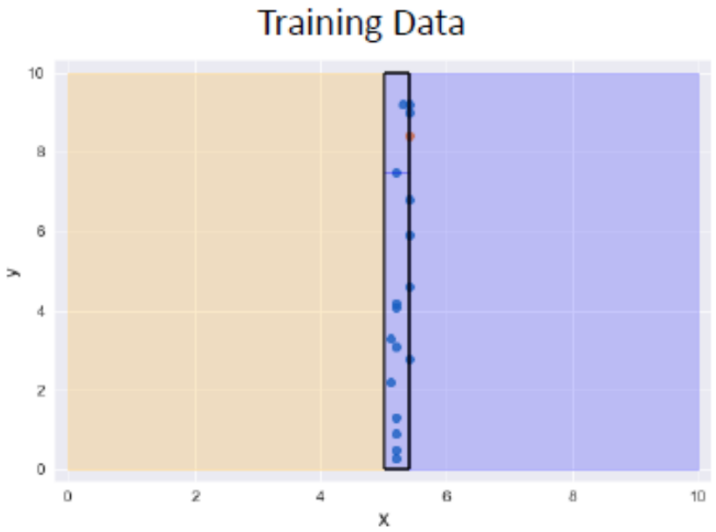


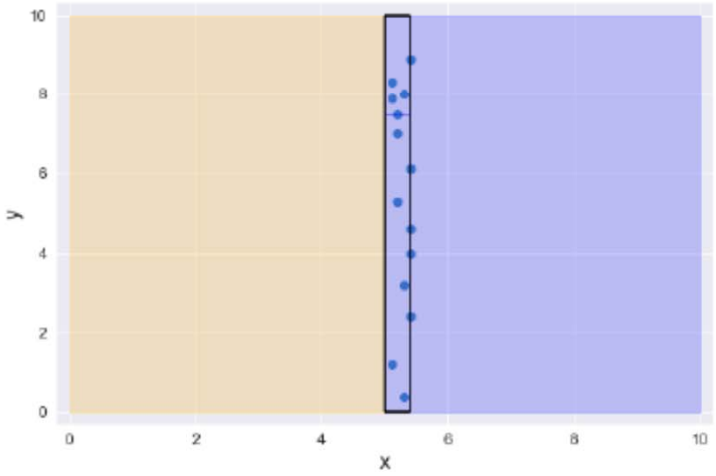


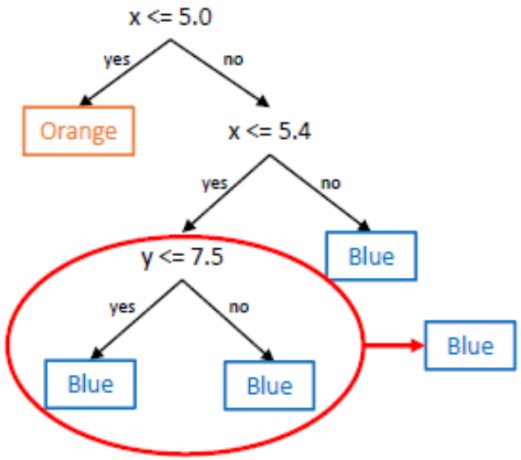
	Errors
Decision Node	3
Leaf	0

Post-Pruning



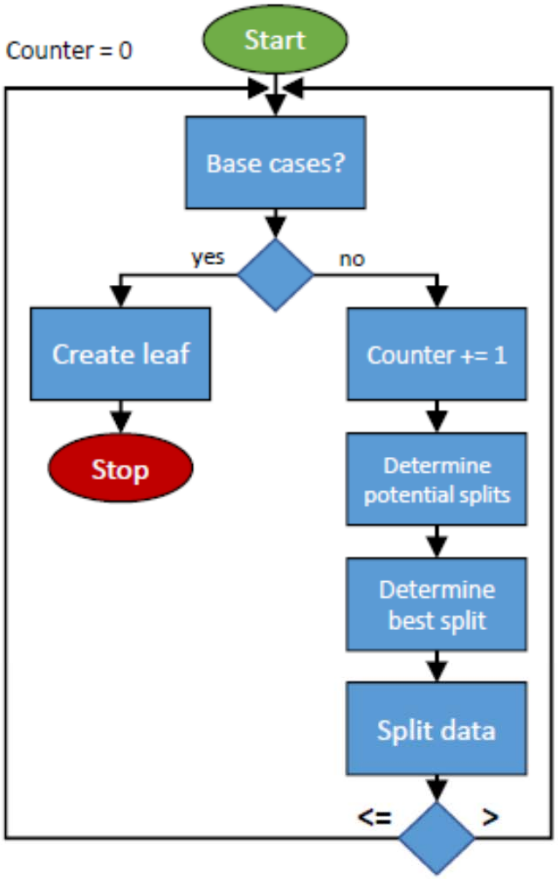


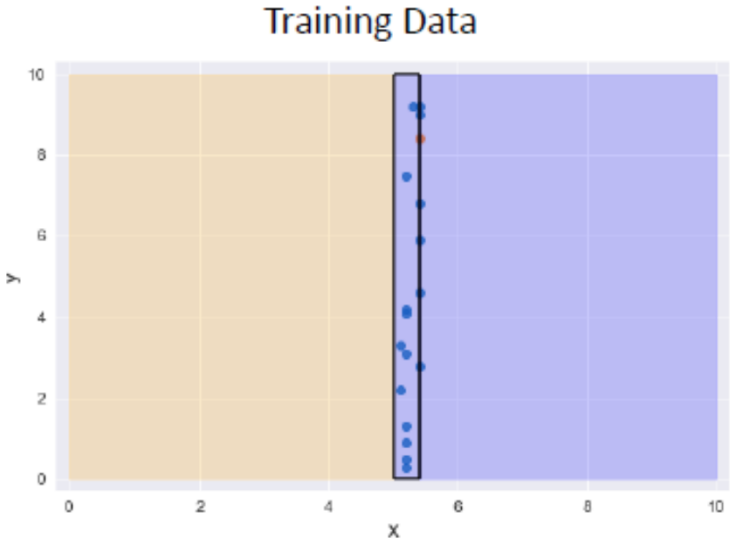


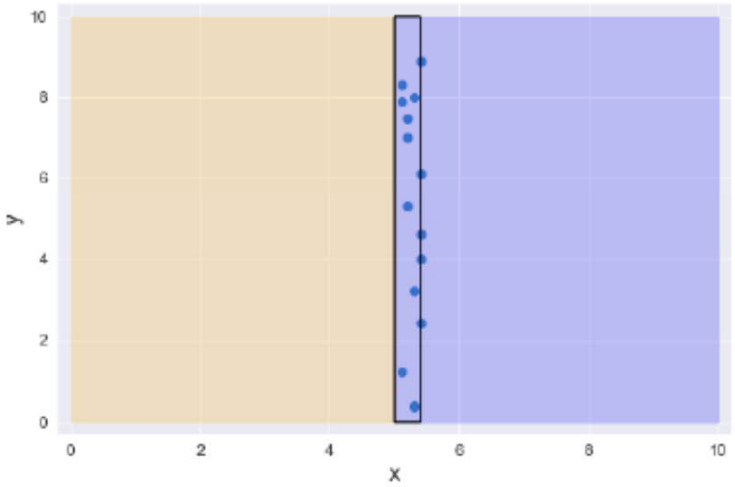


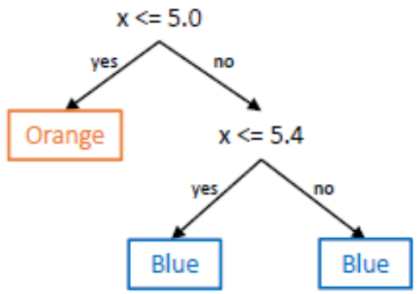
	Errors
Decision Node	0
Leaf	0

Post-Pruning

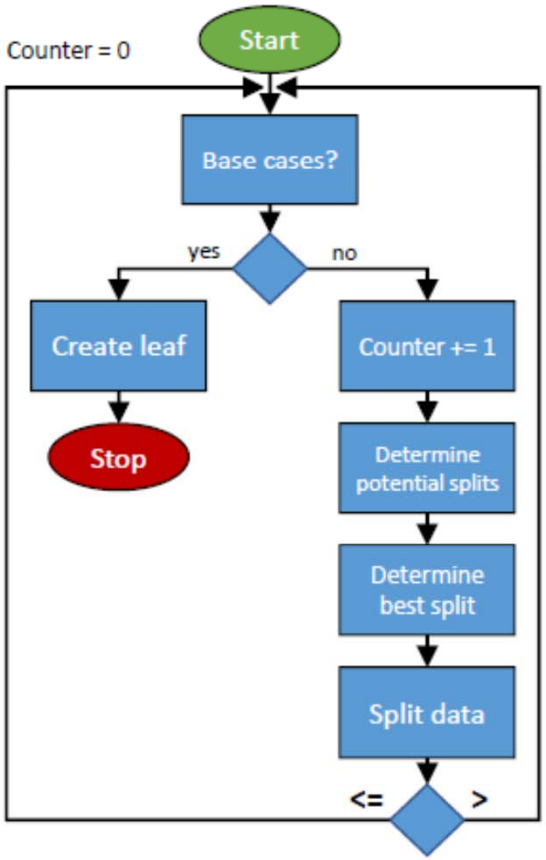


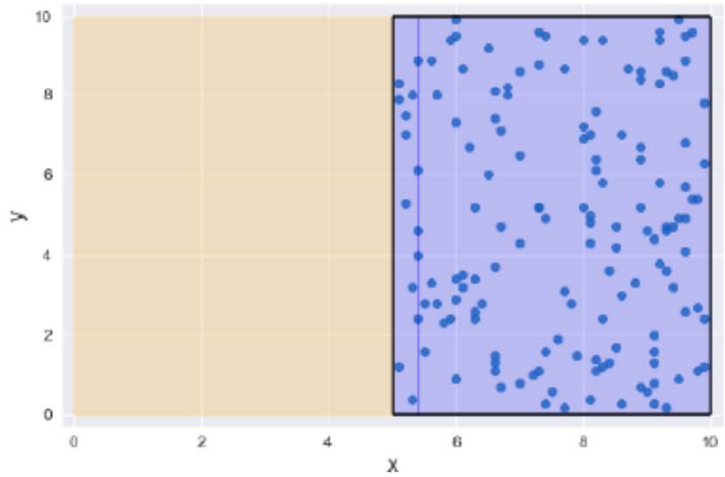


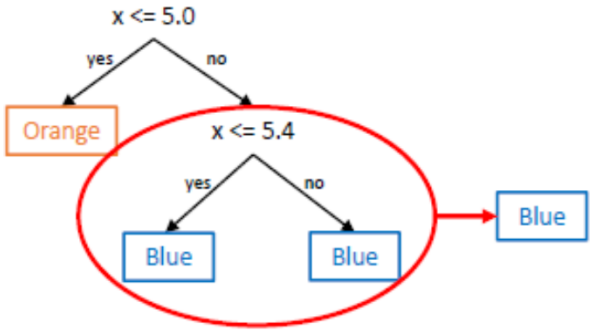




Post-Pruning

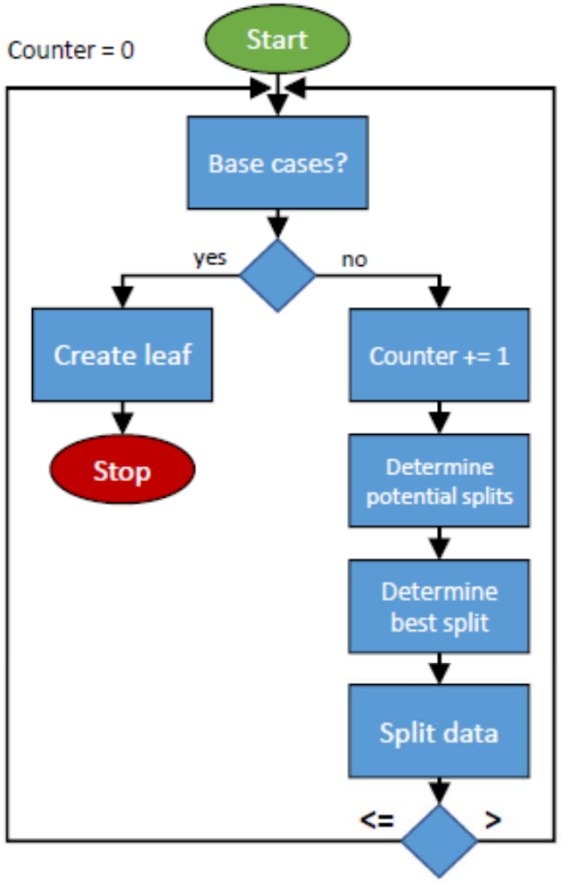




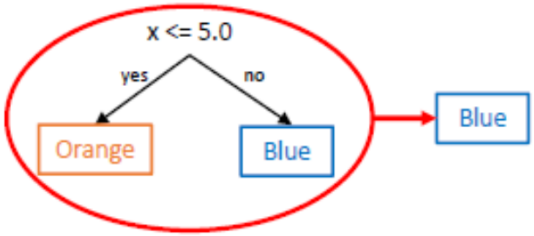


	Errors
Decision Node	0
Leaf	0

Post-Pruning



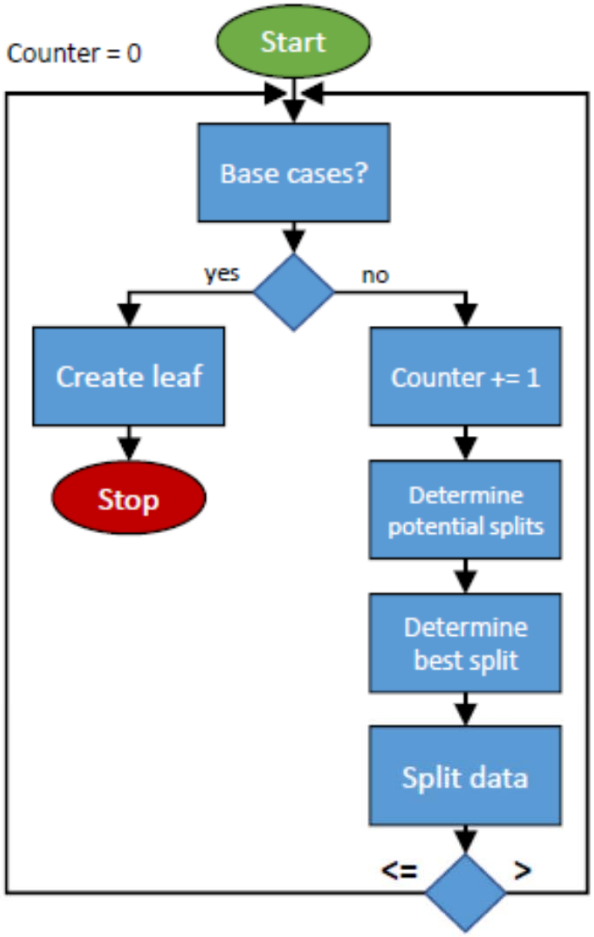
Blue dots: 156
Orange dots: 145

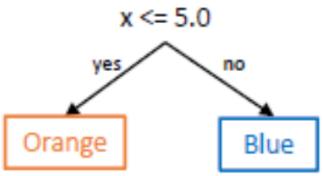


Blue dots: 146
Orange dots: 154

	Errors
Decision Node	0
Leaf	154

Post-Pruning

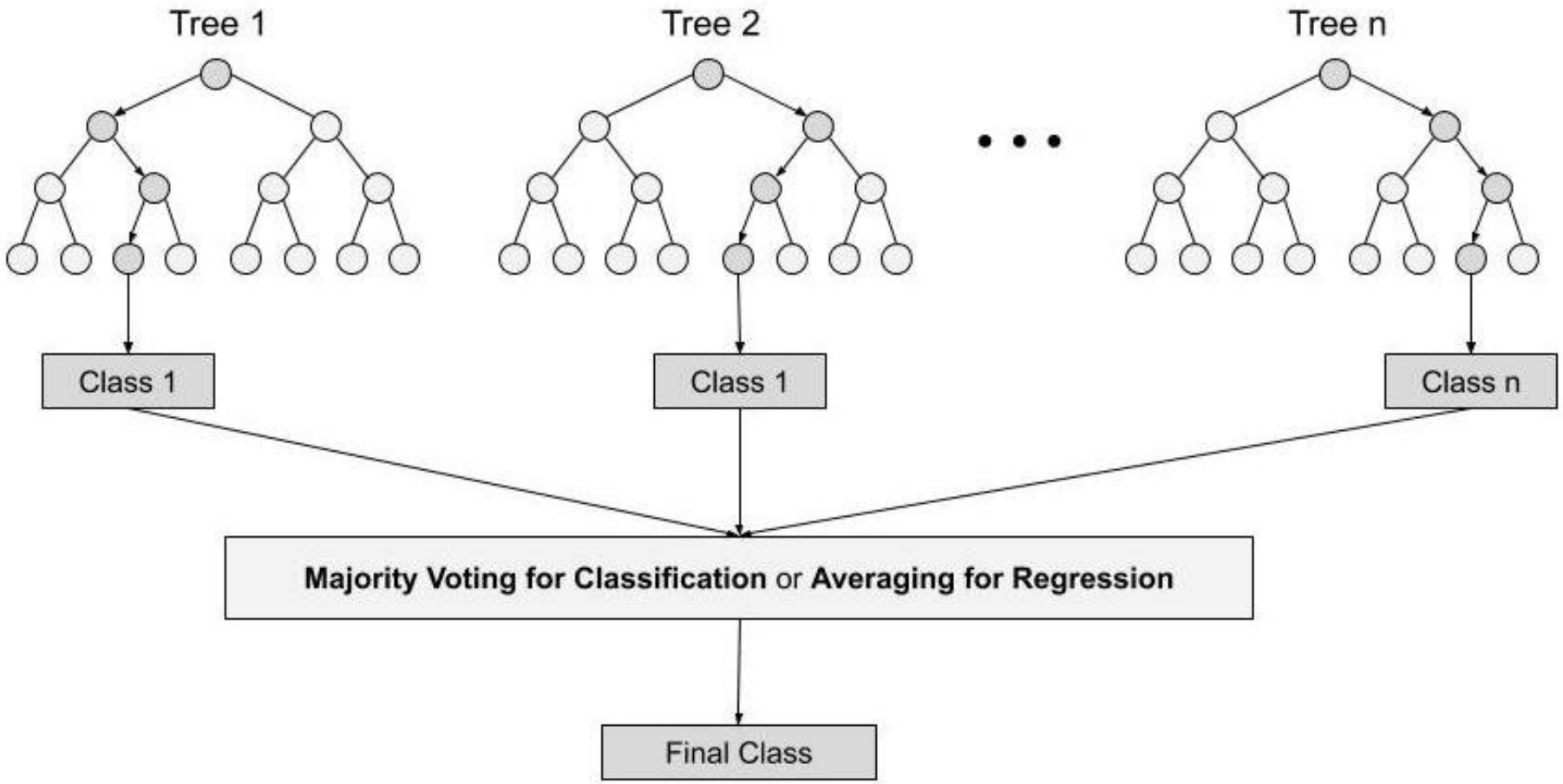




Random forest

Steps involved in random forest algorithm:

- Step 1: In Random forest n number of random records are taken from the data set having k number of records.
- Step 2: Individual decision trees are constructed for each sample.
- Step 3: Each decision tree will generate an output.
- Step 4: Final output is considered based on ***Majority Voting or Averaging*** for Classification and regression respectively.

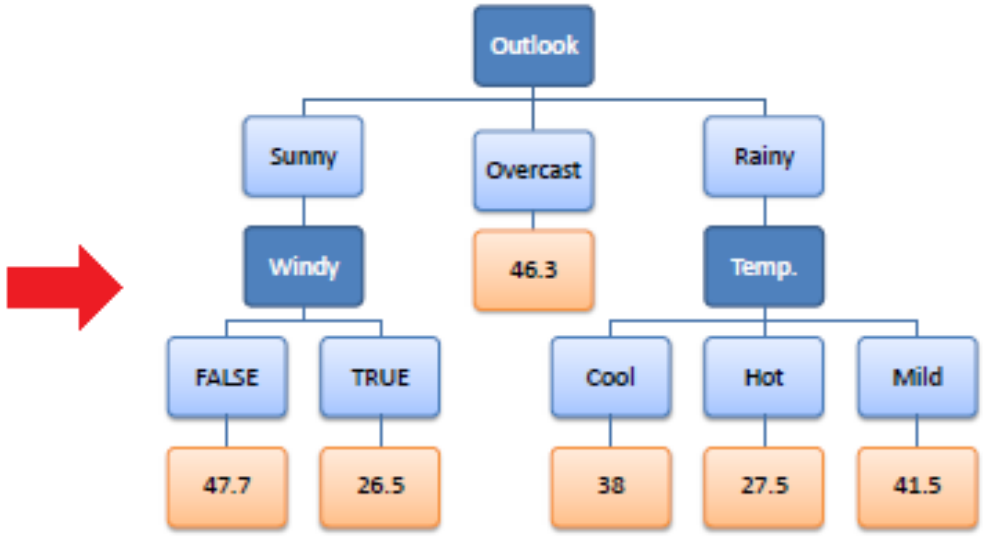

Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

Important Features of Random Forest

- **1. Diversity-** Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **2. Immune to the curse of dimensionality-** Since each tree does not consider all the features, the feature space is reduced.
- **3. Parallelization-**Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **4. Stability-** Stability arises because the result is based on majority voting/ averaging.

Decision Tree for Regression

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



http://www.saedsayad.com/decision_tree_reg.htm

Decision Tree 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. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction.

Standard Deviation

- 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). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

a) Standard deviation for **one** attribute:

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$\textbf{\textit{Count}} = n = 14$$

$$\textbf{\textit{Average}} = \bar{x} = \frac{\sum x}{n} = 39.8$$

$$\textbf{\textit{Standard Deviation}} = S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} = 9.32$$

$$\textbf{\textit{Coeffeicient of Variation}} = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

- Standard Deviation (**S**) is for tree building (branching).
- Coefficient of Deviation (**CV**) is used to decide when to stop branching. We can use Count (**n**) as well.
- Average (**Avg**) is the value in the leaf nodes.

b) Standard deviation for **two** attributes (target and predictor):

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

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = **P**(Sunny)***S**(Sunny) + **P**(Overcast)***S**(Overcast) + **P**(Rainy)***S**(Rainy)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

Standard Deviation Reduction

- The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest standard deviation reduction (i.e., the most homogeneous branches).
- Step 1: The standard deviation of the target is calculated.

Standard deviation (Hours Played) = 9.32

- ***Step 2***: The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

$$SDR(T,X) = S(T) - S(T,X)$$

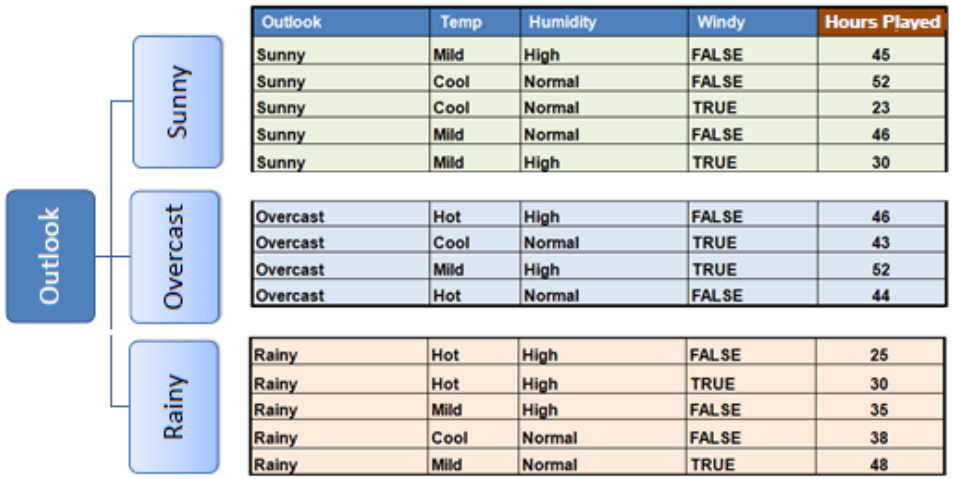
SDR(Hours , Outlook) = **S**(Hours) – **S**(Hours, Outlook)

= 9.32 – 7.66 = 1.66

- ***Step 3***: The attribute with the largest standard deviation reduction is chosen for the decision node.

★		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

- ***Step 4a***: The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.



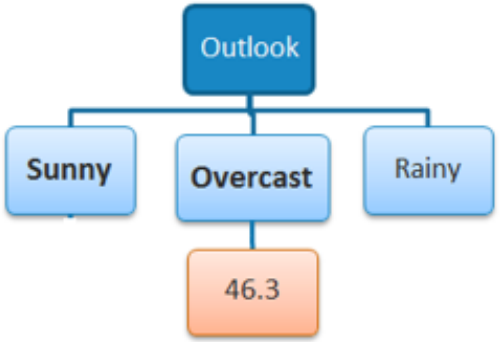
Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

In practice, we need some termination criteria. For example, when coefficient of deviation (**CV**) for a branch becomes smaller than a certain threshold (e.g., 10%) and/or when too few instances (**n**) remain in the branch (e.g., 3).

- ***Step 4b***: "Overcast" subset does not need any further splitting because its CV (8%) is less than the threshold (10%). The related leaf node gets the average of the "Overcast" subset.

Outlook - Overcast

		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5



- ***Step 4c***: However, the "Sunny" branch has an CV (28%) more than the threshold (10%) which needs further splitting. We select "Windy" as the best node after "Outlook" because it has the largest SDR.

Outlook - Sunny

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Cool	Normal	TRUE	23
Mild	Normal	FALSE	46
Mild	High	TRUE	30
			S = 10.87
			AVG = 39.2
			CV = 28%

		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

SDR = 10.87-((2/5)*14.5 + (3/5)*7.32) = **0.678**

		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

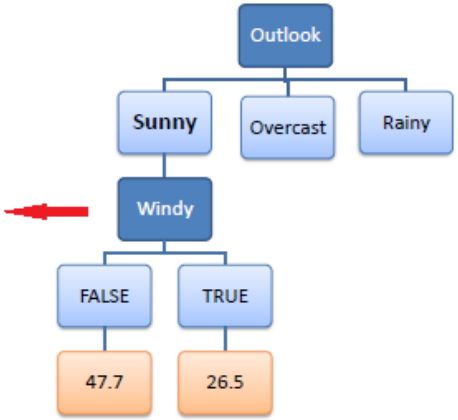
SDR = 10.87-((2/5)*7.5 + (3/5)*12.5) = **0.370**

		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

SDR = 10.87-((3/5)*3.09 + (2/5)*3.5) = **7.62**

- Because the number of data points for both branches (FALSE and TRUE) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Mild	Normal	FALSE	46
Cool	Normal	TRUE	23
Mild	High	TRUE	30



- ***Step 4d***: Moreover, the "rainy" branch has an CV (22%) which is more than the threshold (10%). This branch needs further splitting. We select "Windy" as the best node because it has the largest SDR.

Outlook - Rainy

Temp	Humidity	Windy	Hours Played
Hot	High	FALSE	**25**
Hot	High	TRUE	**30**
Mild	High	FALSE	**35**
Cool	Normal	FALSE	**38**
Mild	Normal	TRUE	**48**
			S = 7.78
			AVG = 35.2
			CV = 22%

		Hours Played (StDev)	Count
Temp	Cool	0	1
	Hot	2.5	2
	Mild	6.5	2

SDR = 7.78 - ((1/5)*0+(2/5)*2.5 + (2/5)*6.5) = **4.18**

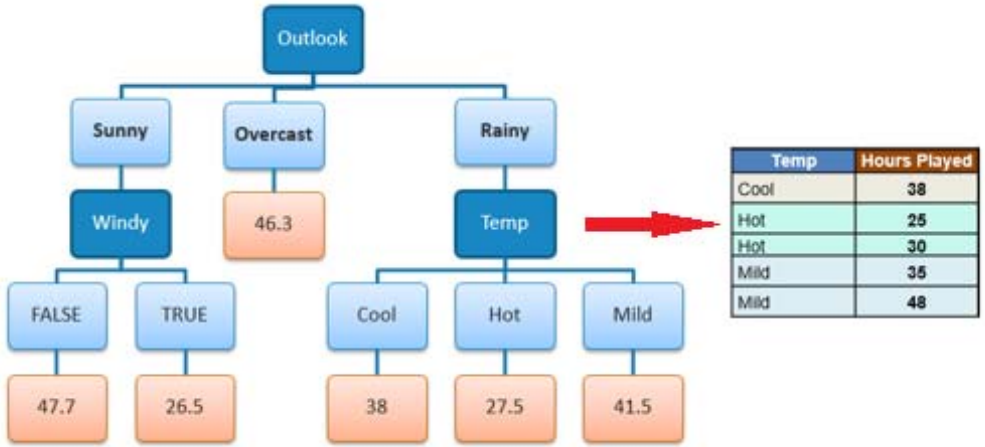
		Hours Played (StDev)	Count
Humidity	High	4.1	3
	Normal	5.0	2

SDR = 7.78 - ((3/5)*4.1 + (2/5)*5.0) = **3.32**

		Hours Played (StDev)	Count
Windy	False	5.6	3
	True	9.0	2

SDR = 7.78 - ((3/5)*5.6 + (2/5)*9.0) = **0.82**

- Because the number of data points for all three branches (Cool, Hot and Mild) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.



Temp	Hours Played
Cool	38
Hot	25
Hot	30
Mild	35
Mild	48

- When the number of instances is more than one at a *leaf node* we calculate the *average* as the final value for the target.

Reference

- The detail material related to this lecture can be found in

Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

https://www.sebastian-mantey.com/theory-blog/decision-tree-algorithm-explained-p4-decision-tree-pruning