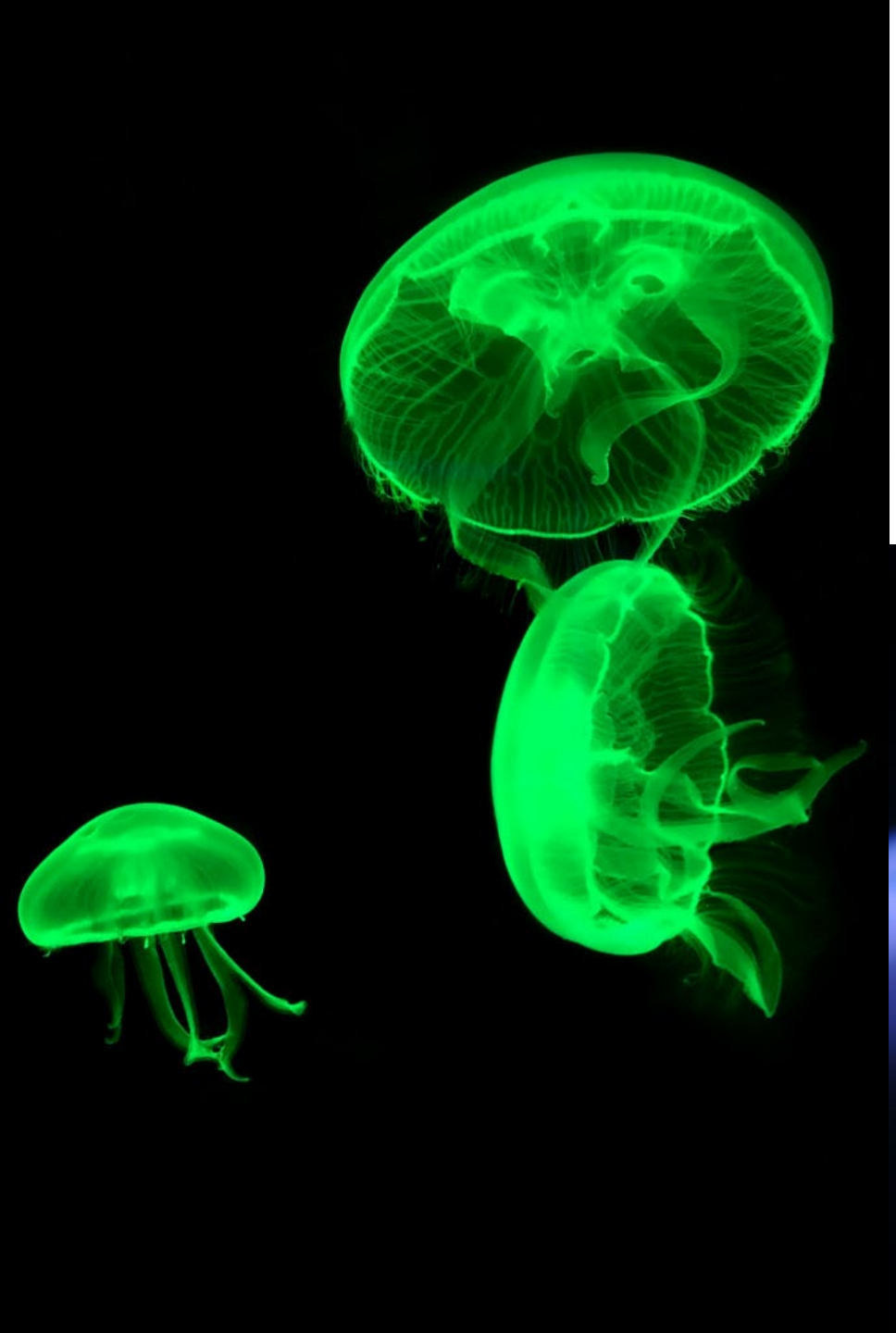
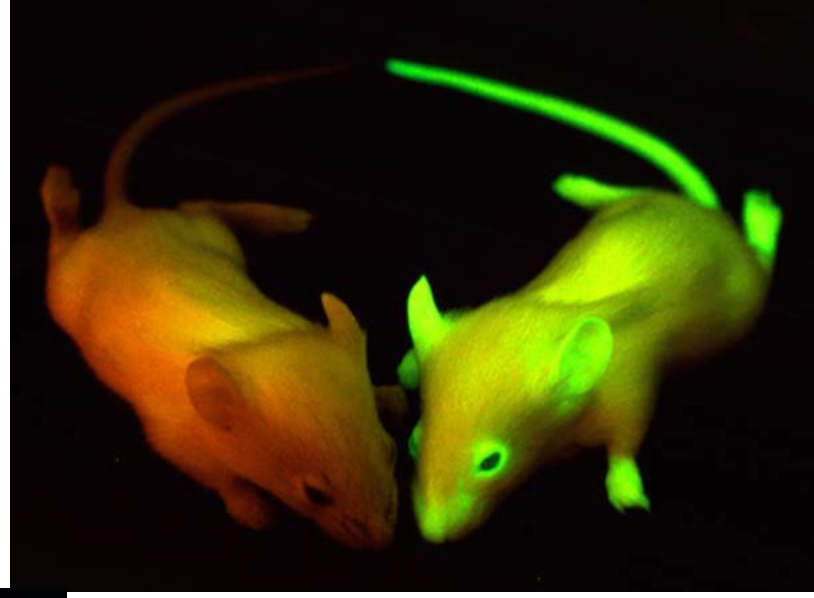
# Decision Trees

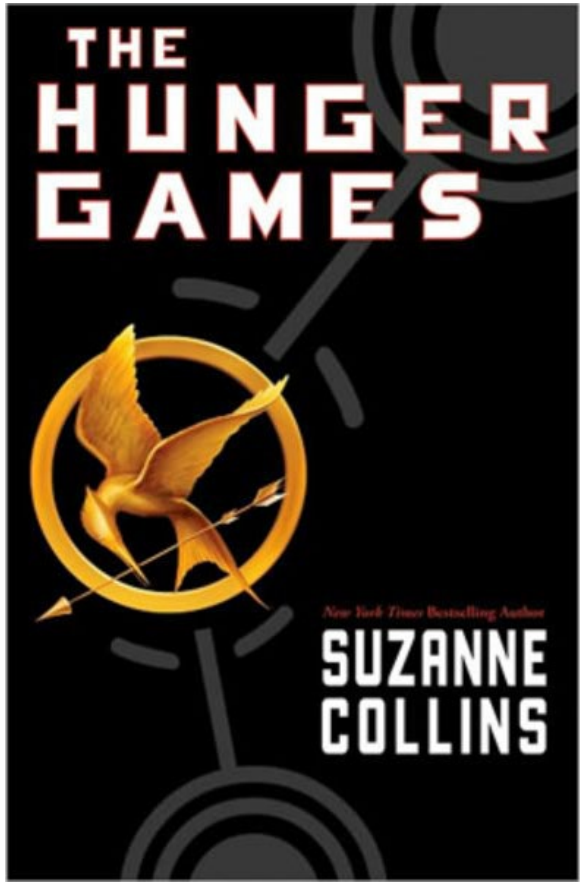




# HW4 is Available

Kaggle competition

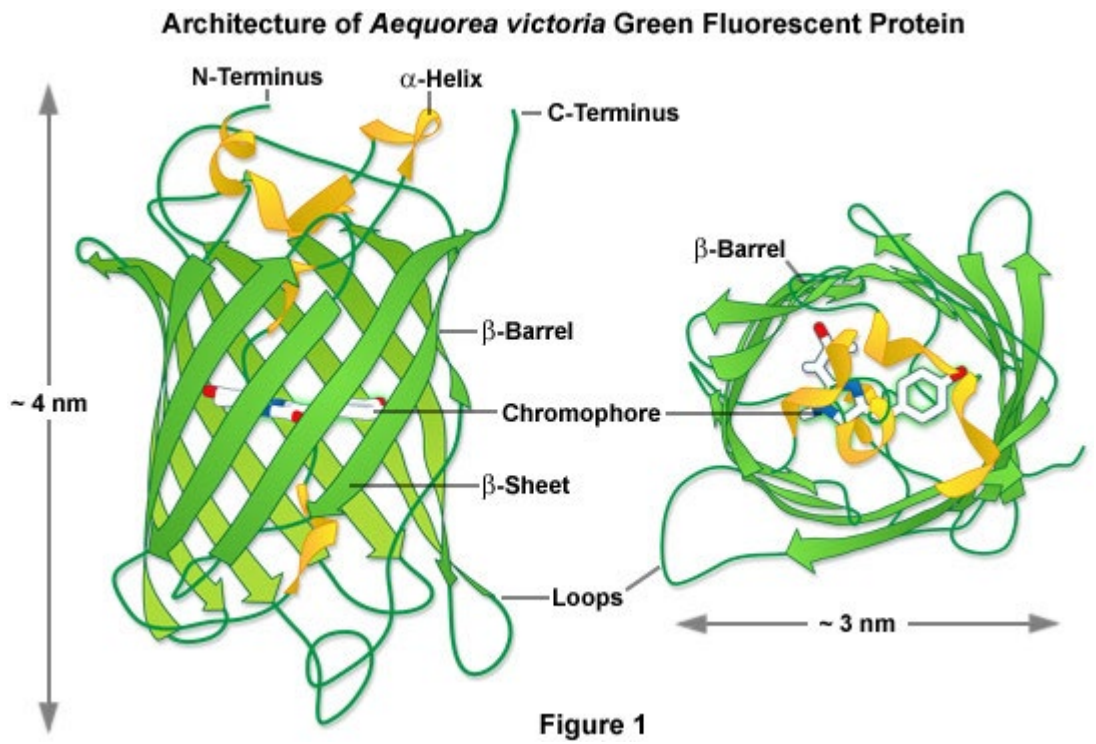
Get data from kaggle.com
Solve the problem
Submit solution for autograding to Kaggle
Submit brief IPYNB file to canvas with win solution



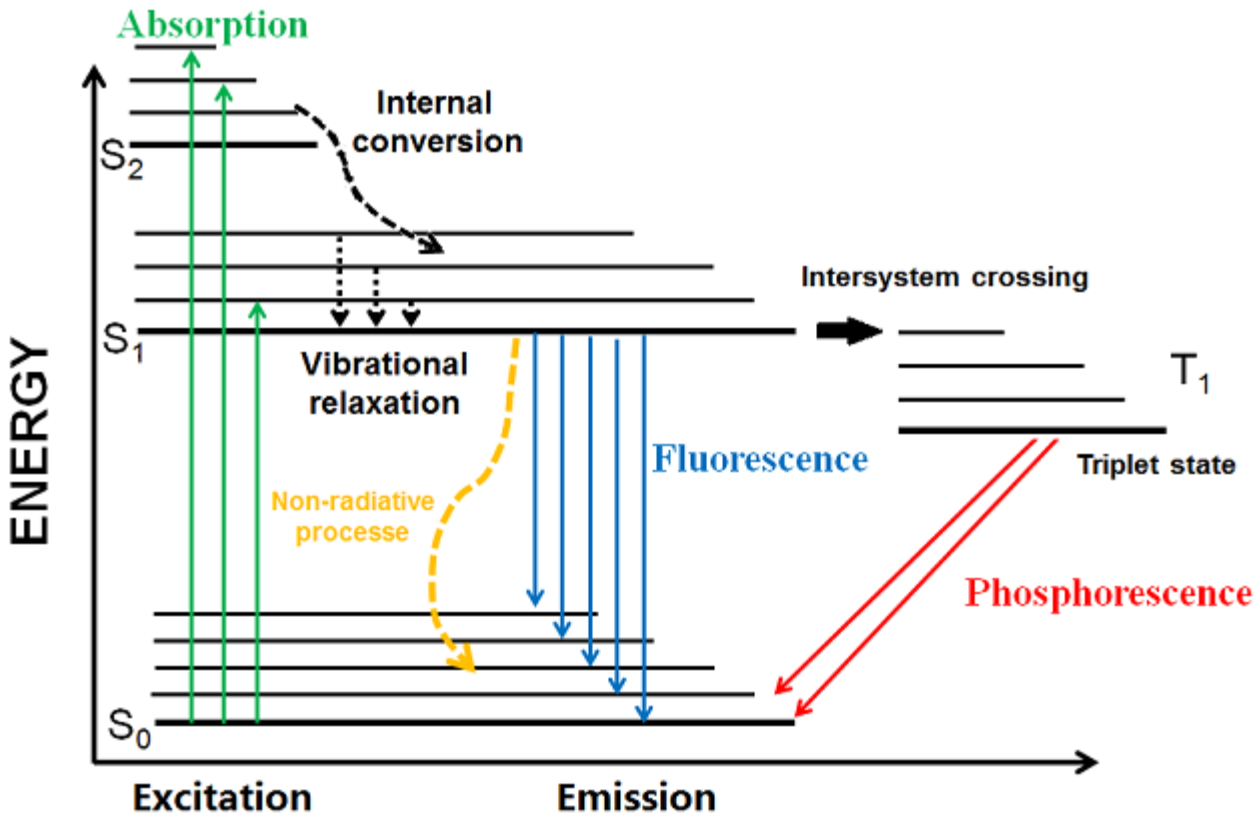

## Sign-up Link:

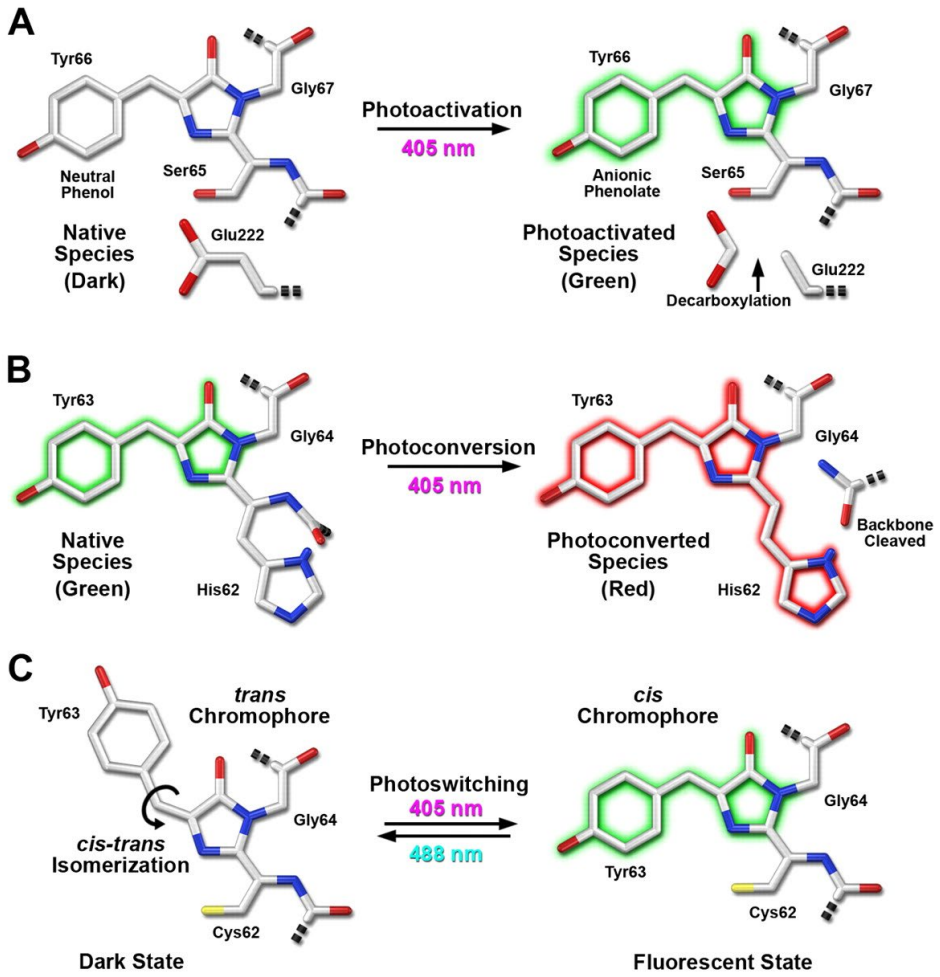
https://www.kaggle.com/t/ea5e608a06aa45d3ab24e2aef5ac6114

# Green fluorescent protein (GFP)

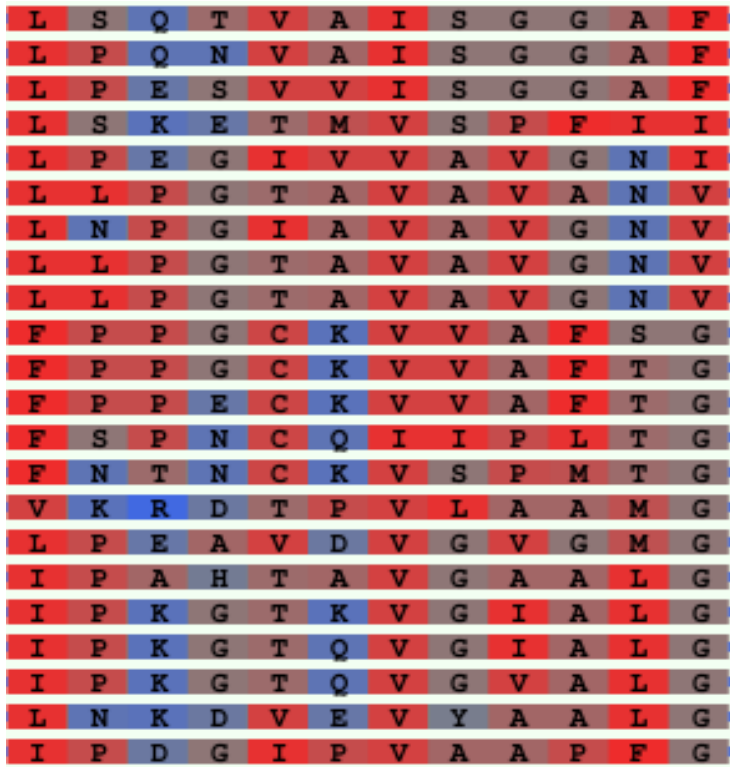


# Green fluorescent protein (GFP)



# Data




...
.
.
.

x

0
1
0
1
0
0
1
1
1
0
0
1
1
.
.
.
.

y

## One Letter Code

| Amino acid | Three letter symbol | One letter symbol* |
|---|---|---|
| Alanine | Ala | A |
| Arginine | Arg | R |
| Asparagine | Asn | N |
| Aspartic acid | Asp | D |
| Cysteine | Cys | C |
| Glutamic acid | Glu | E |
| Glutamine | Gln | Q |
| Glycine | Gly | G |
| Histidine | His | H |
| Isoleucine | Ile | I |
| Leucine | Leu | L |
| Lysine | Lys | K |
| Methionine | Met | M |
| Phenylalanine | Phe | F |
| Proline | Pro | P |
| Serine | Ser | S |
| Threonine | Thr | T |
| Tryptophan | Trp | W |
| Tyrosine | Tyr | Y |
| Valine | Val | V |

## Example of descriptors

| AA | Z1 | Z2 | Z3 |
|---|---|---|---|
| Phe (F) | –4.92 | 1.3 | 0.45 |
| Trp (W) | –4.75 | 3.65 | 0.85 |
| Ile (I) | –4.44 | –1.68 | –1.03 |
| Leu (L) | –4.19 | –1.03 | –0.98 |
| Val (V) | –2.69 | –2.53 | –1.29 |
| Met (M) | –2.49 | –0.27 | –0.41 |
| Tyr (Y) | –1.39 | 2.32 | 0.01 |
| Pro (P) | –1.22 | 0.88 | 2.23 |
| Ala (A) | 0.07 | –1.73 | 0.09 |
| Cys (C) | 0.71 | –0.97 | 4.13 |
| Thr (T) | 0.92 | –2.09 | –1.4 |
| Ser (S) | 1.96 | –1.63 | 0.57 |
| Gln (Q) | 2.19 | 0.53 | –1.14 |
| Gly (G) | 2.23 | –5.36 | 0.3 |
| His (H) | 2.41 | 1.74 | 1.11 |
| Lys (K) | 2.84 | 1.41 | –3.14 |
| Arg (R) | 2.88 | 2.52 | –3.44 |
| Glu (E) | 3.08 | 0.039 | –0.07 |
| Asn (N) | 3.22 | 1.45 | 0.84 |
| Asp (D) | 3.64 | 1.13 | 2.36 |

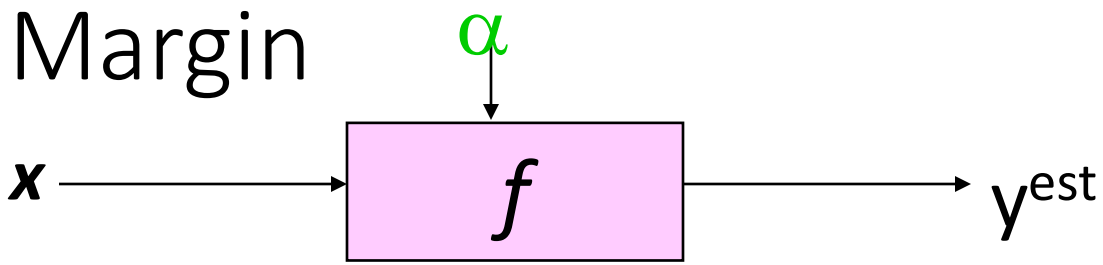
These values are obtained from a principal component analysis of 29 physicochemical properties for the amino acids (see the 'Methods' section). Z1 can be tentatively interpreted as 'hydrophobicity', Z2 as 'bulk of side chain', and Z3 as 'electronic properties', respectively.

# What you need to do:

- Given the training data: Xtrain and Ytrain
- You are allowed to build any type of classification model from Scikit Learn!
- You are allowed to use any type of data processing (feature generation)
- Please explore descriptors/featurizations for amino acid sequences.
- Find the best performing model
- Use your model to score Ytest

- In this work you will predict the brightness level binarized for classification between high brightness (class 1) and low brightness (class 0) for a set of mutants of Green Fluorescent Protein
- Performance will be measured using F1 score metric (sklearn.metrics.f1_score)
- Overfitting is prevented by using public and private leaderboard.

# SMVs recoup: Maximum Margin




$f(x,w,b) = sign(w.\ x - b)$

denotes +1

denotes -1

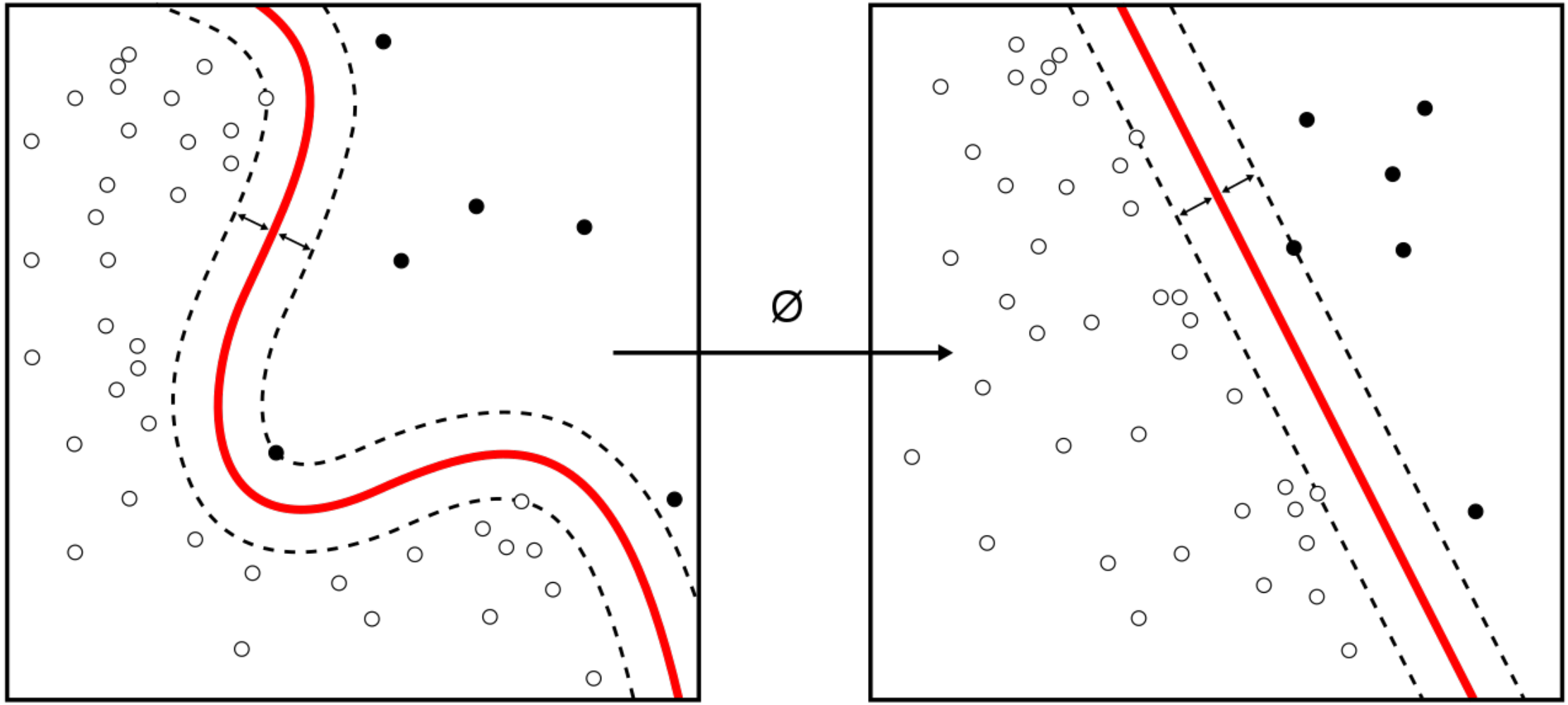
Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

# Kernel Trick



# Kernels and Basis Functions

- In general, kernels make it easy to incorporate basis functions into SVMs:
  - Define $\varphi(\boldsymbol{x})$ any way you like.
  - Define $k(\boldsymbol{x}, \boldsymbol{z}) = \varphi(\boldsymbol{x})^{\boldsymbol{T}} \varphi(\boldsymbol{z})$.

- The kernel function represents a dot product, but in a (typically) higher-dimensional feature space compared to the original space of $\boldsymbol{x}$ and $\boldsymbol{z}$.

# Common SVM basis functions

$\boldsymbol{z}_k$ = ( polynomial terms of $\boldsymbol{x}_k$ of degree 1 to $q$ )

$\boldsymbol{z}_k$ = ( radial basis functions of $\boldsymbol{x}_k$ )

$$\mathbf{z}_k[j] = \varphi_j(\mathbf{x}_k) = \text{KernelFn}\left(\frac{|\mathbf{x}_k - \mathbf{c}_j|}{\text{KW}}\right)$$

$\boldsymbol{z}_k$ = ( sigmoid functions of $\boldsymbol{x}_k$ )

# SVM parameters

SVM has another set of parameters called hyperparameters.

- The soft margin constant *C*.
- Any parameters the kernel function depends on
  - linear kernel – no hyperparameter (except for *C*)
  - polynomial – degree
  - Gaussian – width of Gaussian

# SVMs: Recap

- Advantages:
  - Training finds globally best solution.
    - No need to worry about local optima, or iterations.
  - SVMs can define complex decision boundaries.
- Disadvantages:
  - Training time is cubic to the number of training data. This makes it hard to apply SVMs to large datasets.
  - High-dimensional kernels increase the risk of overfitting.
    - Usually larger training sets help reduce overfitting, but SVMs cannot be applied to large training sets due to cubic time complexity.
  - Some choices must still be made manually.
    - Choice of kernel function.
    - Choice of parameter $C$ in formula $C\left(\sum_{n=1}^{N} \xi_n\right) + \frac{1}{2}\|\boldsymbol{w}\|^{\mathbf{2}}$.

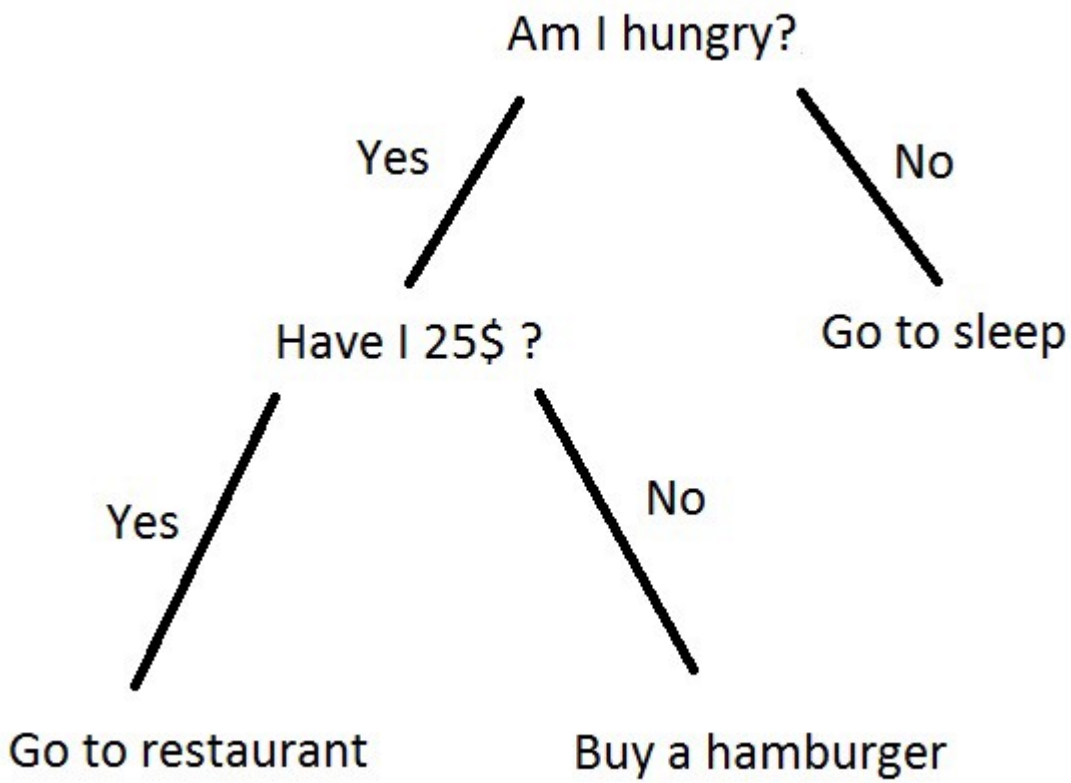
# Lecture 10: Decision Trees

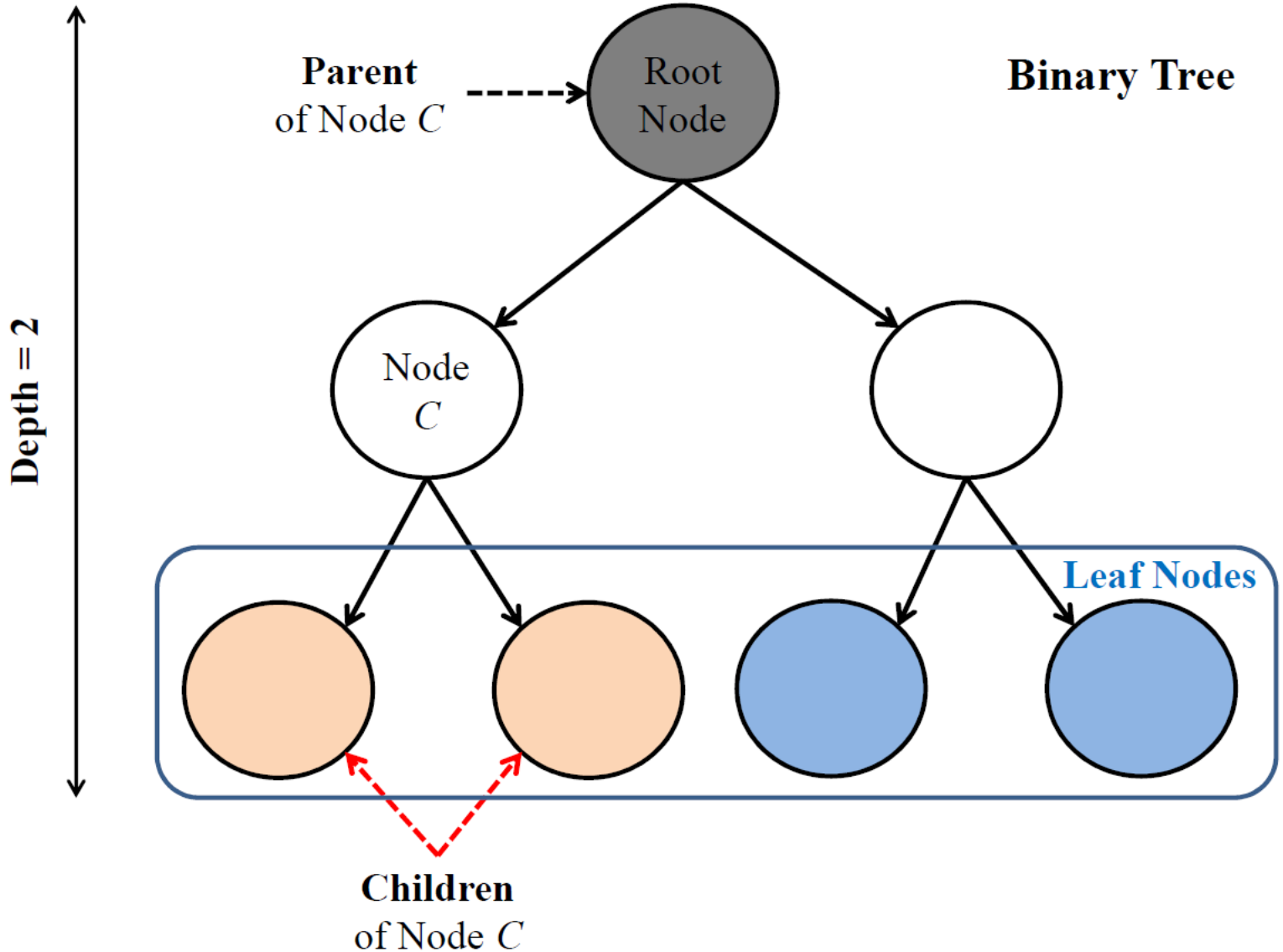
Olexandr Isayev

Department of Chemistry, CMU

olexandr@cmu.edu

# Decision Trees



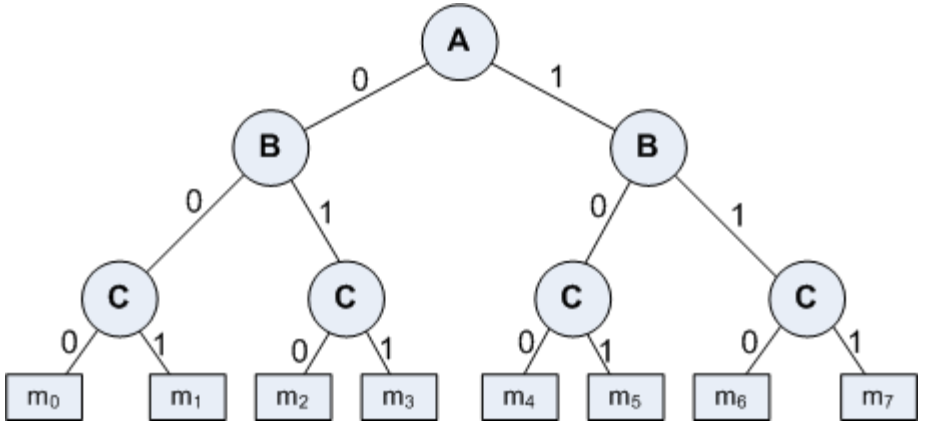

Depth = 2
Parent
of Node C
Root
Node
Binary Tree
Node
C
Leaf Nodes
Children
of Node C

# Basic Concept

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

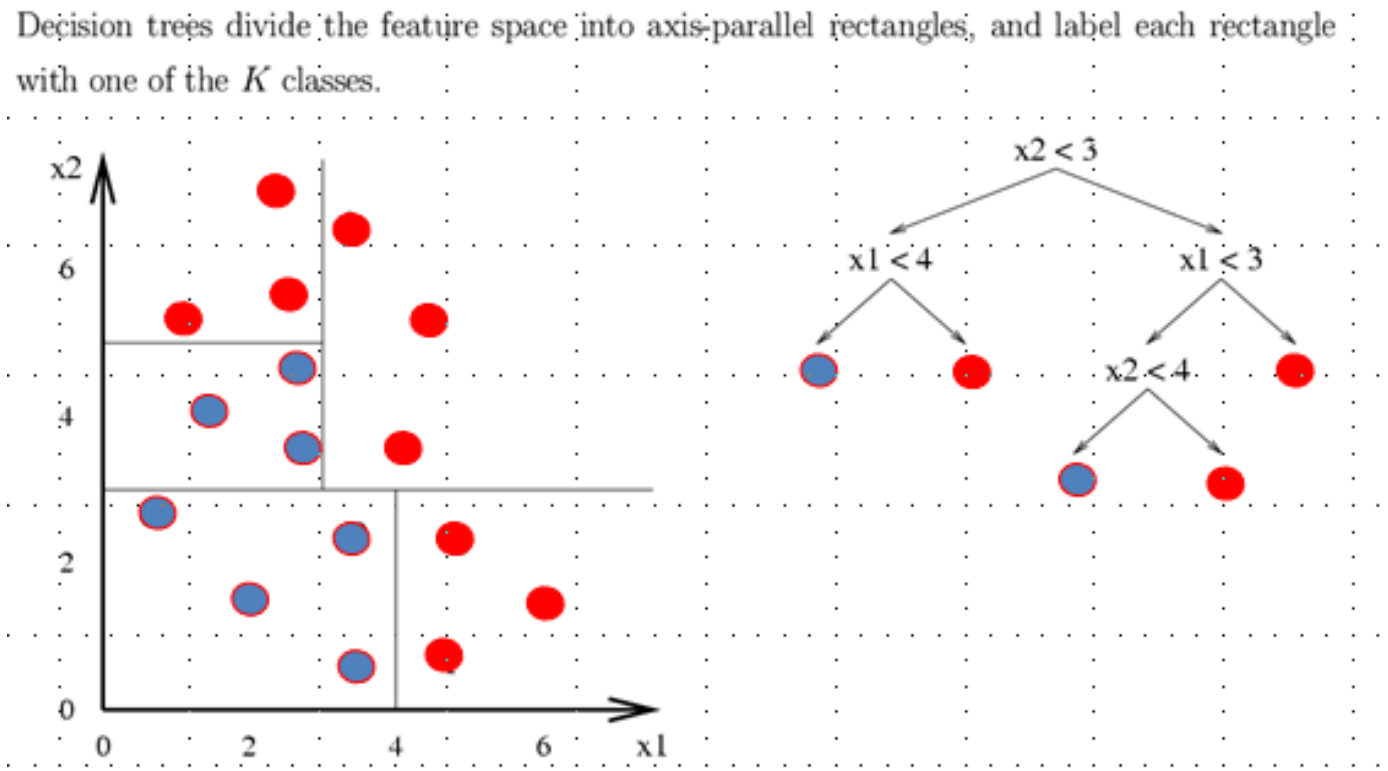
## Binary Decision Tree

| *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 | $m_7$ |



# Decision tree

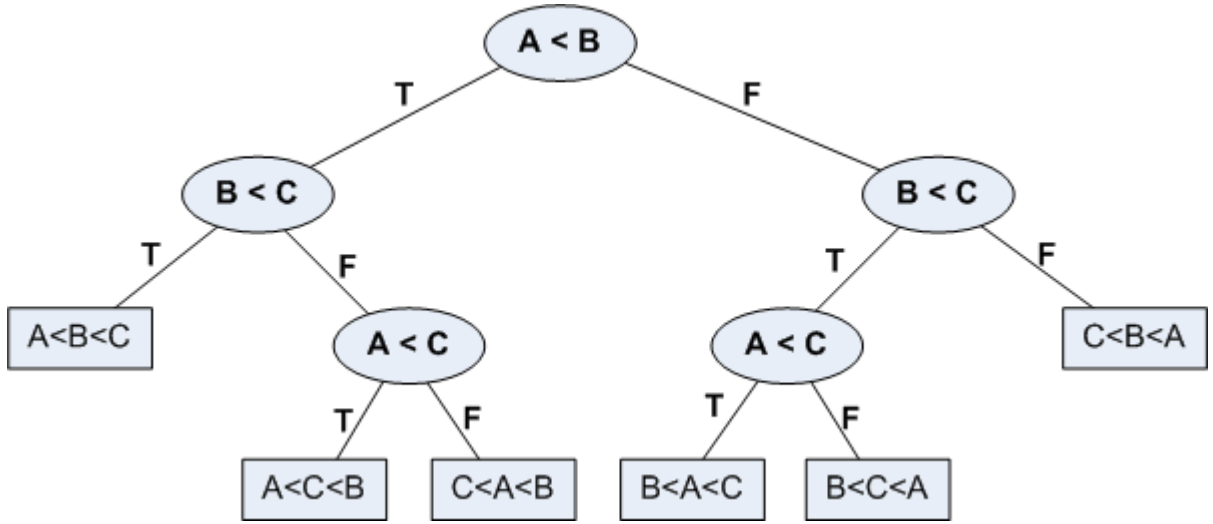
- A flow-chart-like tree structure
- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution



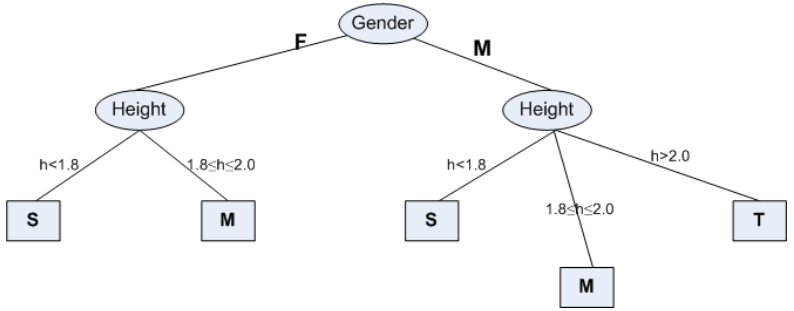
# Basic Concept

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

**Decision Tree with numeric data**



# Some Characteristics



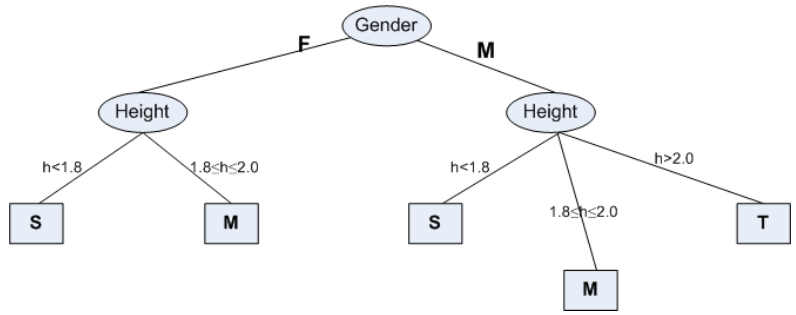

- Decision tree may be *n*-ary, $n \geq 2$.
- There is a special node called root node.
- All nodes drawn with circle (ellipse) are called internal nodes.
- All nodes drawn with rectangle boxes are called terminal nodes or leaf nodes.
- Edges of a node represent the outcome for a value of the node.
- 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?

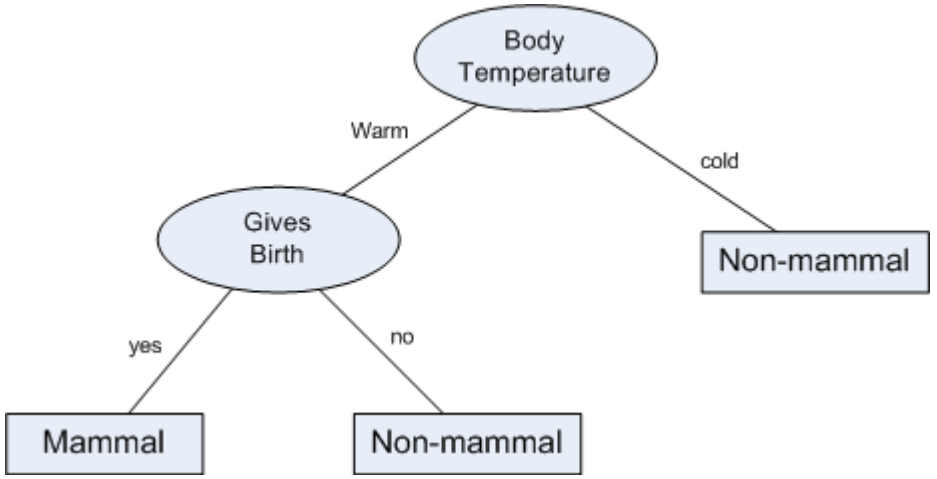
# Decision Tree and Classification Task

**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 |
|---|---|---|---|---|---|---|---|---|
| Sea 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

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

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 of them may give inaccurate result

Two approaches are known

**Greedy strategy**

A top-down recursive divide-and-conquer

**Modification of greedy strategy**

- ID3
- C4.5
- C5
- 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*

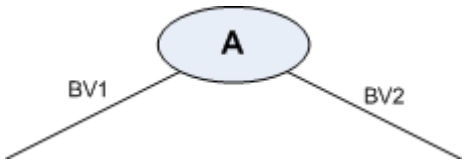
6. Stop

# 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

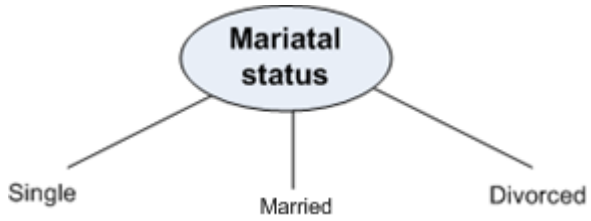
- This is the simplest case of node splitting
- The test condition for a binary attribute generates only two outcomes




# 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



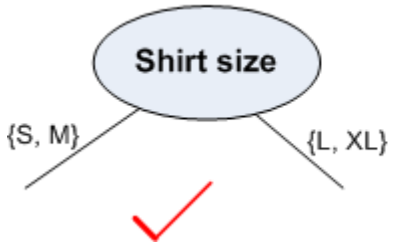

- Binary splitting by grouping attribute values



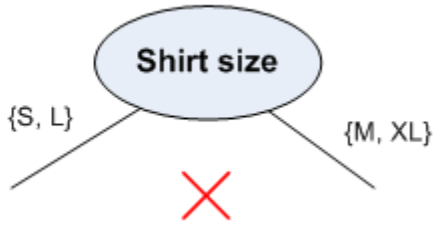
# Node Splitting in **BuildDT** Algorithm

## 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



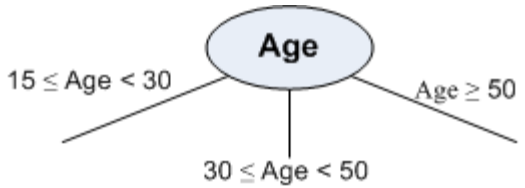




# Node Splitting in **BuildDT** Algorithm

## 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, …, $q$ (if $q$ number of ranges are chosen)
    - Here, q should be decided a priori


For a numeric attribute, decision tree induction is a combinatorial optimization problem

# 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:**

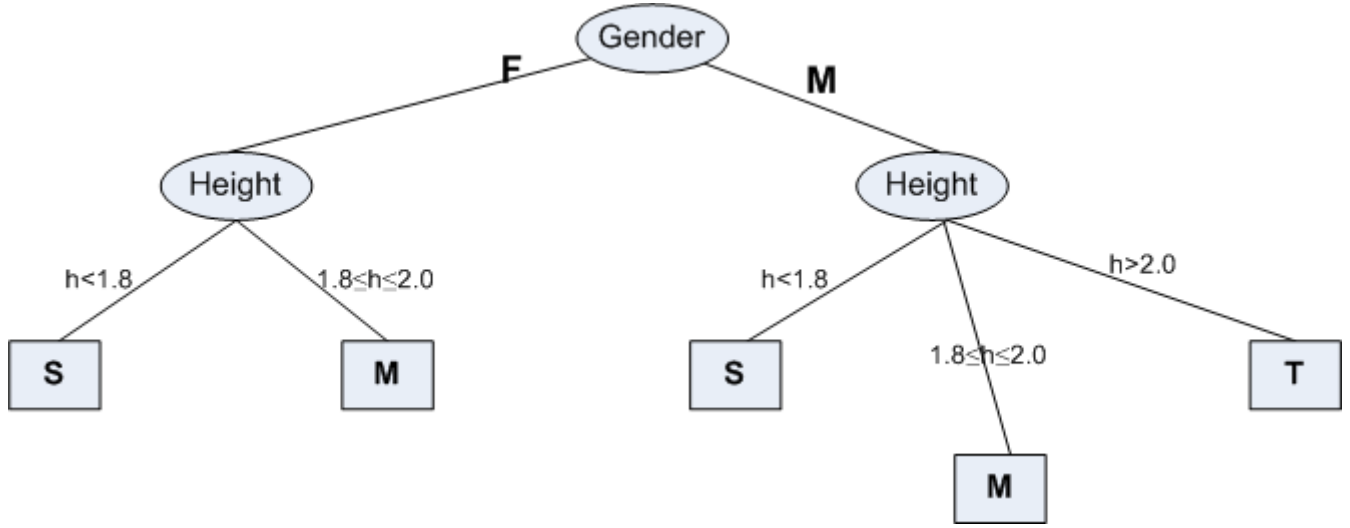
Gender = {Male(M), Female (F)} // Binary attribute
Height = {1.5, …, 2.5} // Continuous attribute

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

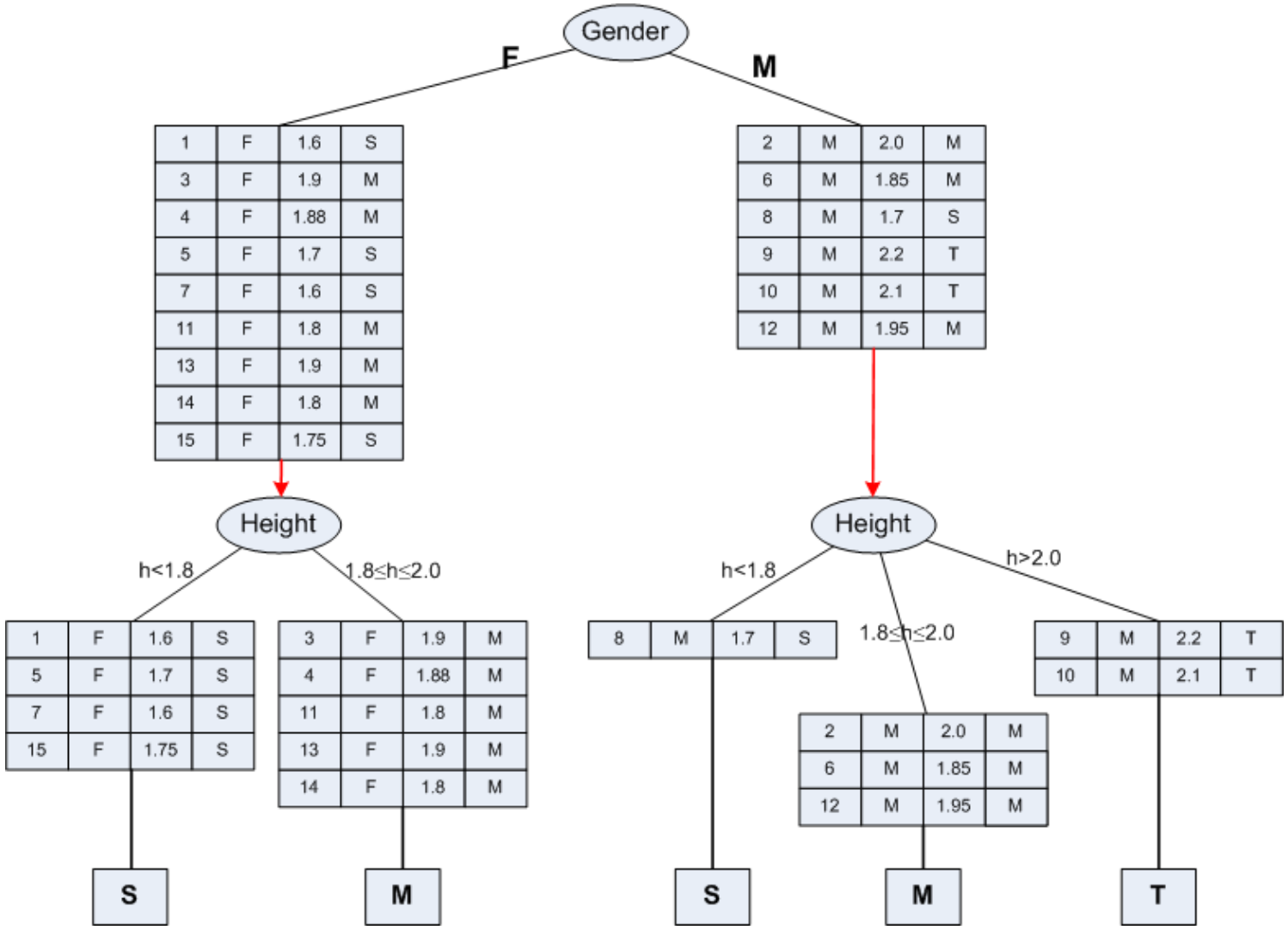
Given a person, we are to test in which class s/he belongs

# 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>**

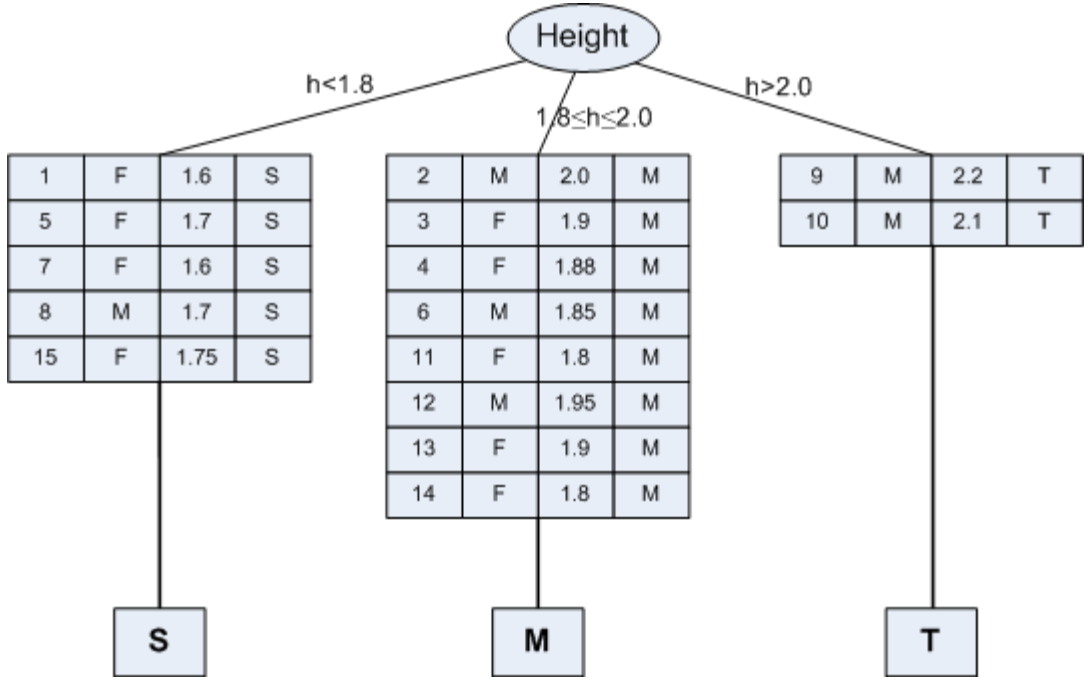


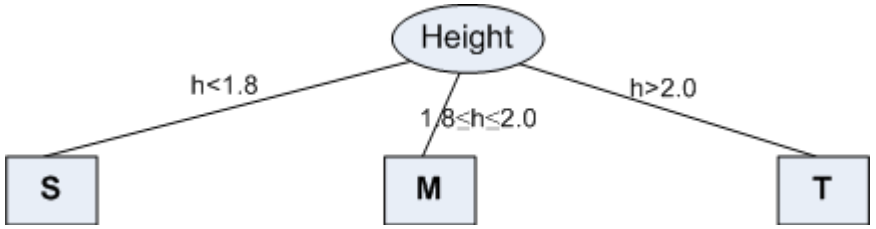
# Illustration : **BuildDT** Algorithm



# Illustration : **BuildDT** Algorithm

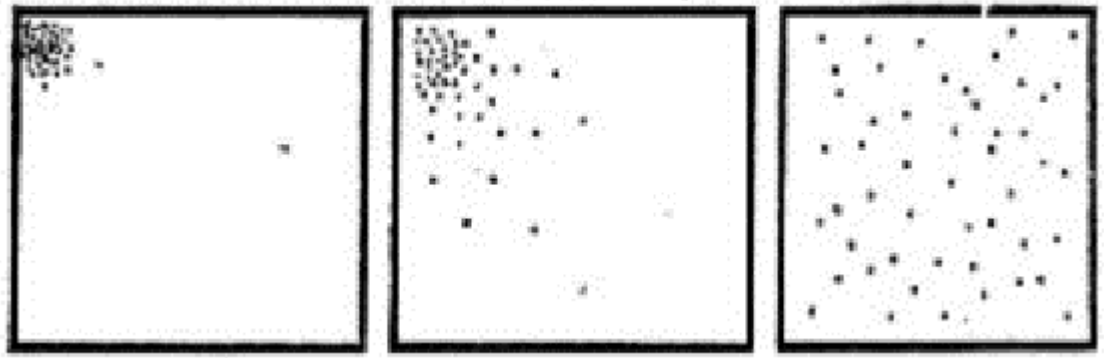
**Approach 2 : <Height, Gender>**



# Concept of Entropy

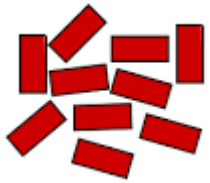
# Concept of Entropy



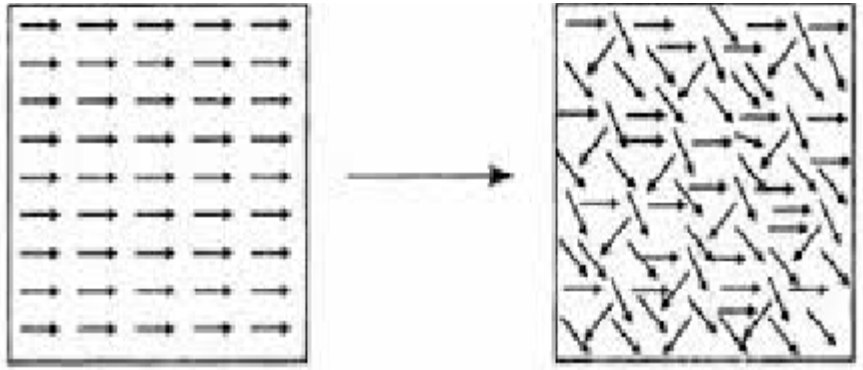
If a point represents a gas molecule, then which system has the more entropy?

How to measure? $\Delta S = \frac{\Delta Q}{T}$ ?

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



**Less** ordered
**higher** entropy



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

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

# Entropy and its Meaning

- Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.
- At a later stage, with the growth of Information Technology, entropy becomes an important concept in Information Theory.
- To deal with the classification job, entropy is an important concept, which is considered as
  - an information-theoretic measure of the “uncertainty” contained in a training data
    - due to the presence of more than one classes.

# Entropy in Information Theory

- The entropy concept in information theory first time coined by Claude Shannon (1850).
- The first time it was used to measure the “information content” in messages.
- According to his concept of entropy, presently entropy is widely being used as a way of representing messages for efficient transmission by Telecommunication Systems.

# Measure of Information Content

People, in general, are information hungry!

Everybody wants to acquire information (from newspaper, library, nature, fellows, etc.)

> Think how a crime detector do it to know about the crime from crime spot and criminal(s).
>
> Kids annoyed their parents asking questions.

In fact, fundamental thing is that we gather information asking questions (and decision tree induction is no exception).

We may note that information gathering may be with certainty or uncertainty.

# Definition of Entropy

Suppose there are *m* distinct objects, which we want to identify by asking a series of **Yes/No** questions. Further, we assume that *m* is an exact power of 2, say $m = 2^n$, where $n \geq 1$.

The entropy of a set of *m* distinct values is the minimum number of yes/no questions needed to determine an unknown values from these *m* possibilities.

The minimum number of *yes/no* questions needed to identify an unknown object from $m = 2^n$ equally likely possible object is *n.*

If *m* is not a power of 2, then the entropy of a set of *m* distinct objects that are equally likely is $\log_2 m$

# Entropy in Messages

We know that the most conventional way to code information is using binary bits, that is, using 0s and 1s.

The answer to a question that can only be answered *yes/no* (with equal probability) can be considered as containing one **unit of information**, that is, one bit.

In other words, the unit of information can also be looked at as the amount of information that can be **coded** using only 0s and 1s.

# Entropy in Messages

**Information coding**

- If we have **two** possible objects say **male** and **female,** then we use the coding
  0 = female
  1 = male

$$m = 2(= 2^n, n = 1)$$

- We can encode **four** possible objects say **East**, **West**, **North**, **South** using two bits, for example
  00 : North
  01 : East
  10 : West
  11 : South

$$m = 4(= 2^n, n = 2)$$

- We can encode **eight** values say eight different colours, we need to use **three** bits, such as
  000 : Violet
  001 : Indigo
  010 : Blue
  011 : Green
  100 : Yellow
  101 : Orange
  110 : Red
  111 : White

$$m = 8(= 2^n, n = 3)$$

Thus, in general, to code *m* values, each in a distinct manner, we need *n* bits such that $m = 2^n$.

# Entropy of Messages when ($m \neq 2^n$)

**Entropy Calculation**

The entropy of a set of *m* distinct objects is $\log_2 m$ even when *m* is not exactly a power of 2.

We have arrived at a conclusion that *E* = $\log_2 m$ for any value of *m,* irrespective of weather it is a power of 2 or not.

Note: *E* is not necessarily be an integer always.

Next, we are to have our observation, if all *m* objects are not equally probable.

Suppose, $p_i$ denotes the frequency with which the $i^{th}$ of the *m* objects occurs, where $0 \leq p_i \leq 1$ for all $p_i$ such that

$$\sum_{i=1}^{m} p_i = 1$$

# Information Content

If an object occurs with frequency *p*, then the most efficient way to represent it with $\log_2(1/p)$ bits.

If there are *m* objects with frequencies $p_1$, $p_2$……., $p_m$, then the average number of bits (i.e. questions) that need to be examined a value, that is, entropy is the frequency of occurrence of the $i^{th}$ value multiplied by the number of bits that need to be determined, summed up values of $i$ from *1* to *m*.

If $p_i$ denotes the frequencies of occurrences of *m* distinct objects, then the entropy *E* is

$$E = \sum_{i=1}^{m} p_i \log(1/p_i) \; and \sum_{i=1}^{m} p_i = 1$$

# 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 three important of them

- **ID3**
- **C 4.5**
- **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

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.

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 $\alpha$ should be chosen as the splitting attribute for *D*.

# 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.

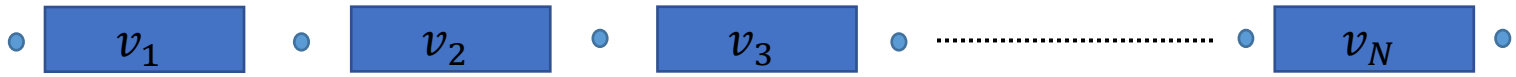
On the other hand, when a training set is such that if there are $k$ classes, and the entropy of training set takes the largest value i.e., $\log_2 k$ (this occurs when the classes are balanced), then the information gain will be zero.

# 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.

**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.

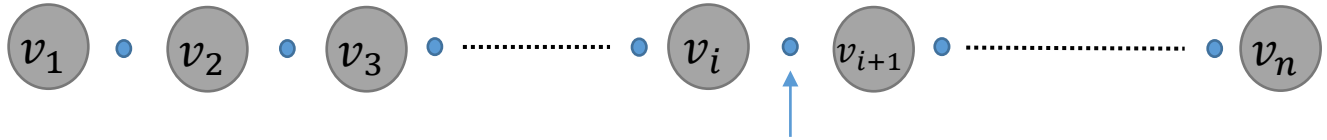



So, if there are $n - groups$ of discrete values, then we have $(n + 1)$ split points.

# Splitting of Continuous attribute values

**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 $\gamma$.

# 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

$G(D)$ 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 and CART

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

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

This $\gamma(A, D)$ is called the Gini Index or "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.

# Algorithm C4.5

# Algorithm C 4.5: Introduction

J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotometer 3).

Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.

ID3 uses information gain measure, which is, in fact biased towards splitting attribute having a large number of outcomes.

For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.

> In such a case, note that each partition is pure, and hence the purity measure of the partition, that is $E_A(D) = 0$

# Algorithm: C 4.5 : Introduction

Although, the previous situation is an extreme case, intuitively, we can infer that ID3 favours splitting attributes having a large number of values

compared to other attributes, which have a less variations in their values.

Such a partition appears to be useless for classification.

This type of problem is called **overfitting problem**.

**Note:**

Decision Tree Induction Algorithm ID3 may suffer from overfitting problem.

# Algorithm: C 4.5

The overfitting problem in ID3 is due to the measurement of information gain.

In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as $\beta$.

Gain Ratio is a kind of normalization to information gain using a **split information**.

The gain ratio can be defined as follows. We first define split information $E_A^*(D)$ as

$$E_A^*(D) = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} . \log \frac{|D_j|}{|D|}$$

Here, *m* is the number of distinct values in *A*.

The gain ratio is then defined as $\beta(A, D) = \frac{\alpha(A,D)}{E_A^*(D)}$ , where $\alpha(A, D)$ denotes the information gain on splitting the attribute *A* in the dataset *D*.

# Notes on Decision Tree algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms employ a heuristic based approach to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.

2. **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.

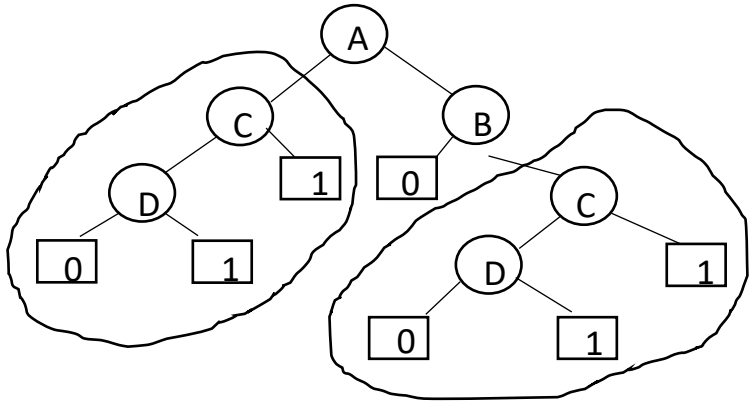
3. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.

4. **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 algorithms

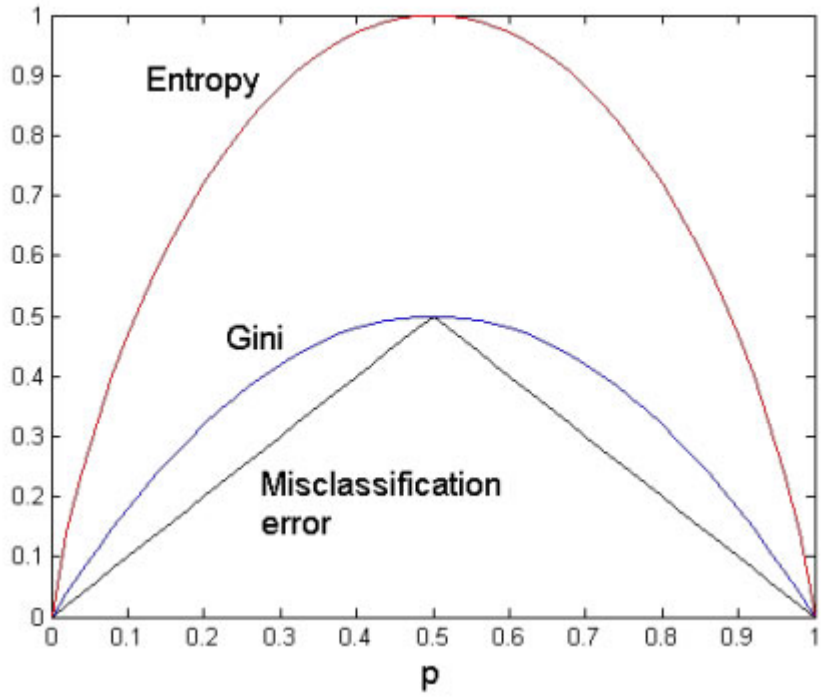
5. **Data Fragmentation Problem:** Since the decision tree algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem 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.

6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



# Notes on Decision Tree algorithms

7. **Decision tree equivalence:** The different splitting criteria followed in different decision tree induction algorithms have little effect on the performance of the algorithms. This is because the different heuristic measures (such as information gain ($\alpha$), Gini index ($\gamma$) and Gain ratio ($\beta$) are quite consistent with each other); also see the figure below.



# Summary of Decision Tree Algorithms

We have learned the building of a decision tree given a training data.

The decision tree is then used to classify a test data.

For a given training data *D*, the important task is to build the decision tree so that:

All test data can be classified accurately

The tree is balanced and with as minimum depth as possible, thus the classification can be done at a faster rate.

In order to build a decision tree, several algorithms have been proposed. These algorithms differ from the chosen splitting criteria, so that they satisfy the above mentioned objectives as well as the decision tree can be induced with minimum time complexity. We have studied three decision tree induction algorithms namely ID3, CART and C4.5.