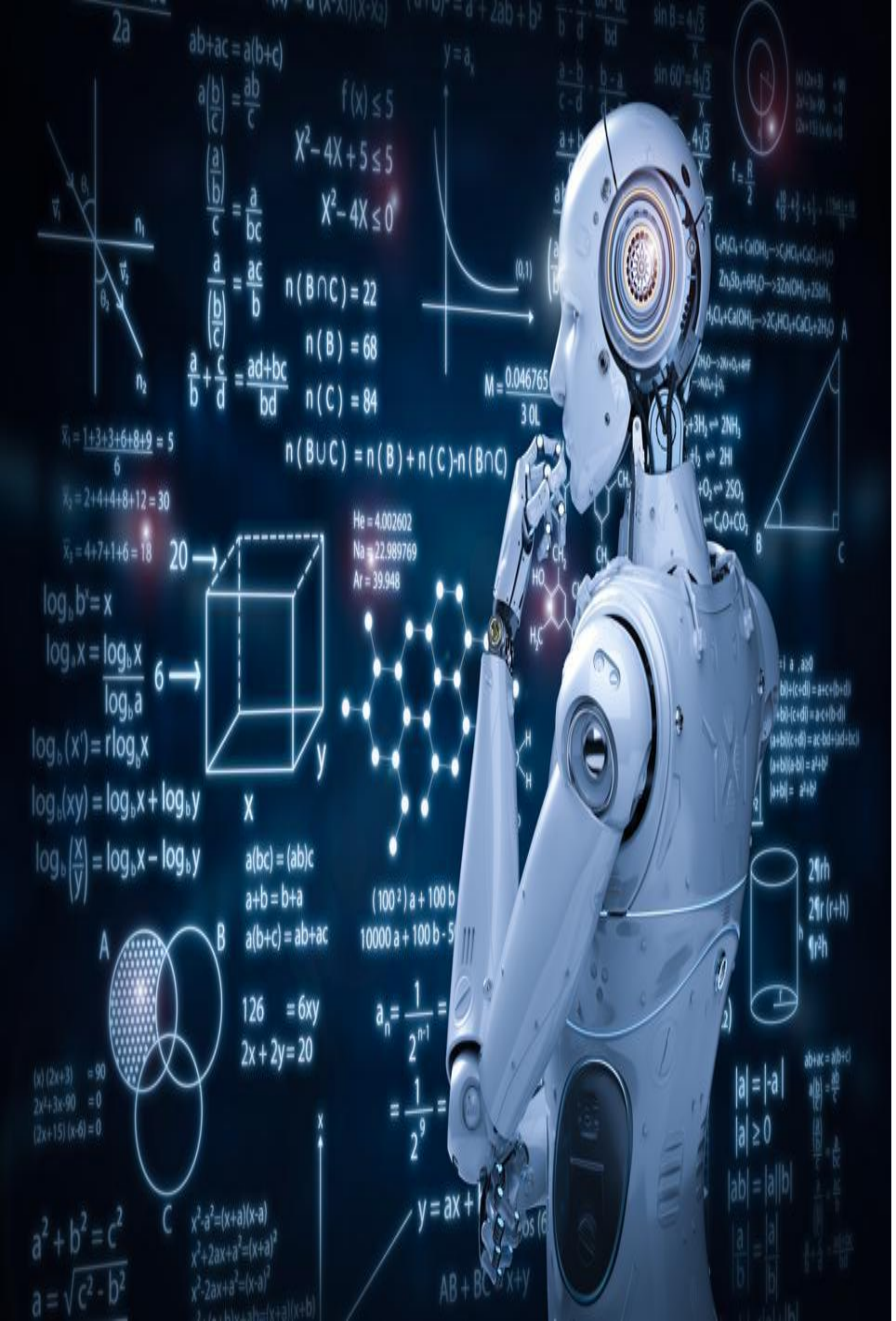


Practical Machine Learning

Day 7: MAR24 DBDA

Kiran Waghmare

Agenda

- Decision Tree

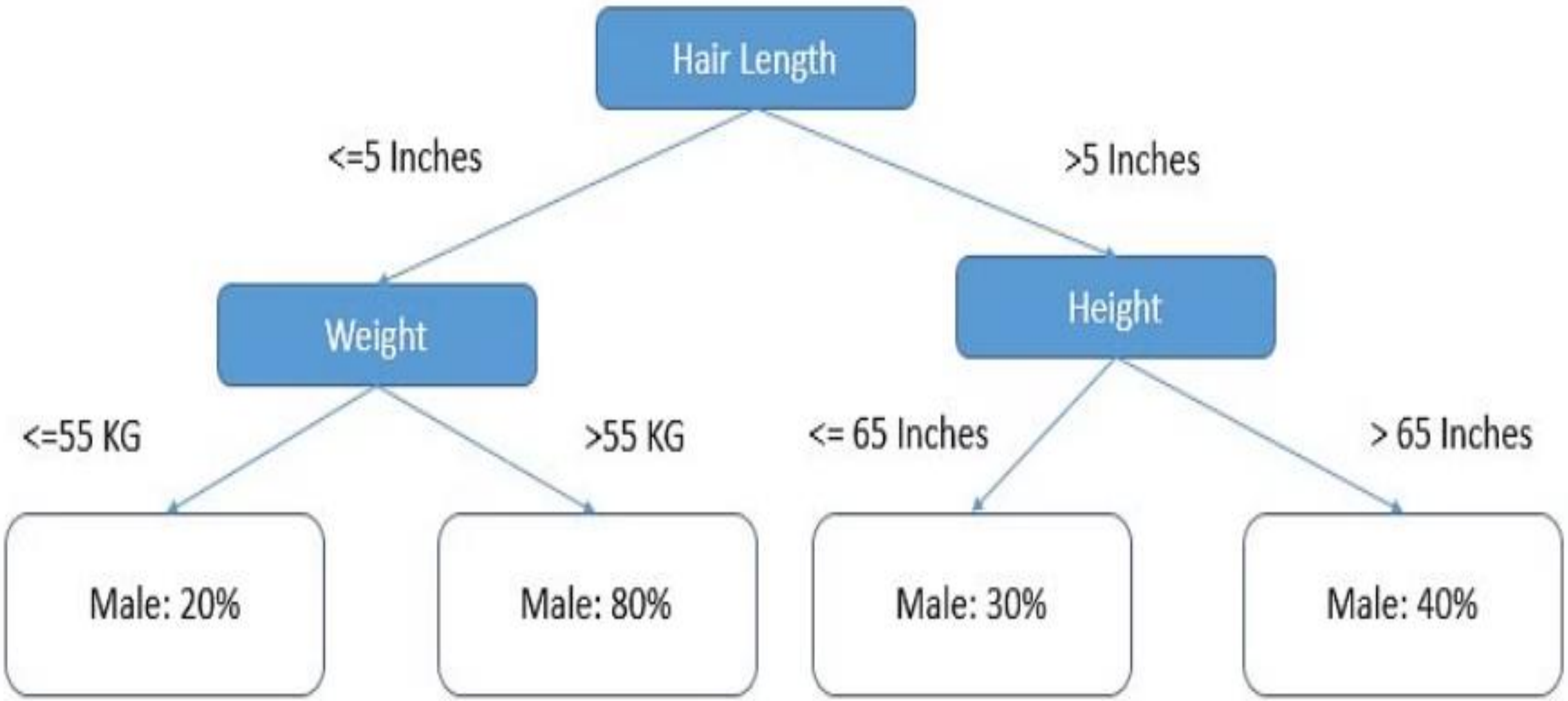
Problem statement

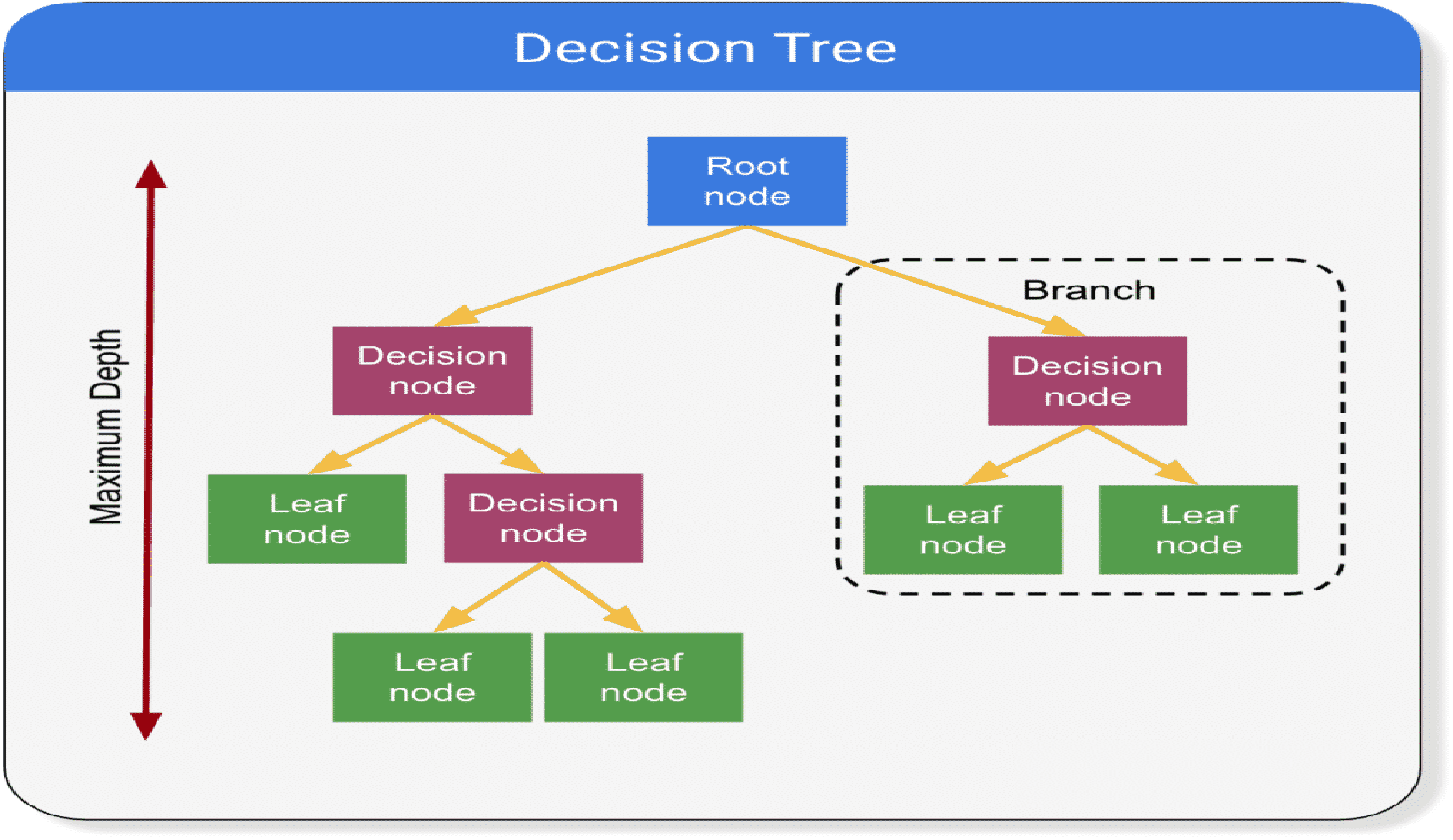
- Assume that you are given a characteristic information of 10,000 people living in your town. You are asked to study them and come up with the algorithm which should be able to tell whether a **new person coming to the town is male or a female**.

  - Primarily you are given information about:
  - Skin colour
  - Hair length
  - Weight
  - Height

- Based on the information you can divide the information in such a way that it somehow indicates the characteristics of Males vs. Females.

Example 1:


Hair Length
<=5 Inches
>5 Inches
Weight
Height
<=55 KG
>55 KG
<= 65 Inches
> 65 Inches
Male: 20%
Male: 80%
Male: 30%
Male: 40%


Decision Tree
Root node
Branch
Decision node
Decision node
Leaf node
Decision node
Leaf node
Leaf node
Leaf node
Leaf node
Maximum Depth

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts
- It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node
- , and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree
- is called the parent node, and other nodes are called the child nodes.

Decision Tree

- A decision tree is one of the **most powerful tools** of supervised learning algorithms used for **both classification and regression tasks.**

- It builds a **flowchart-like tree structure** where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

- It is **constructed by recursively splitting the training data into subsets** based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

Attribute Selection Measures

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**
- By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:
  - **Information Gain**
  - **Gini Index**

$$Entropy\,(P) = -\sum_{i=1}^{n} p_i \log_2 (p_i)$$

Information Gain and Gini Index in Decision Tree

$$Gini\,(P) = 1 - \sum_{i=1}^{n} (p_i)^2$$

1. Information Gain:

- Information gain is the **measurement of changes in entropy after the segmentation of a dataset based** on an attribute.

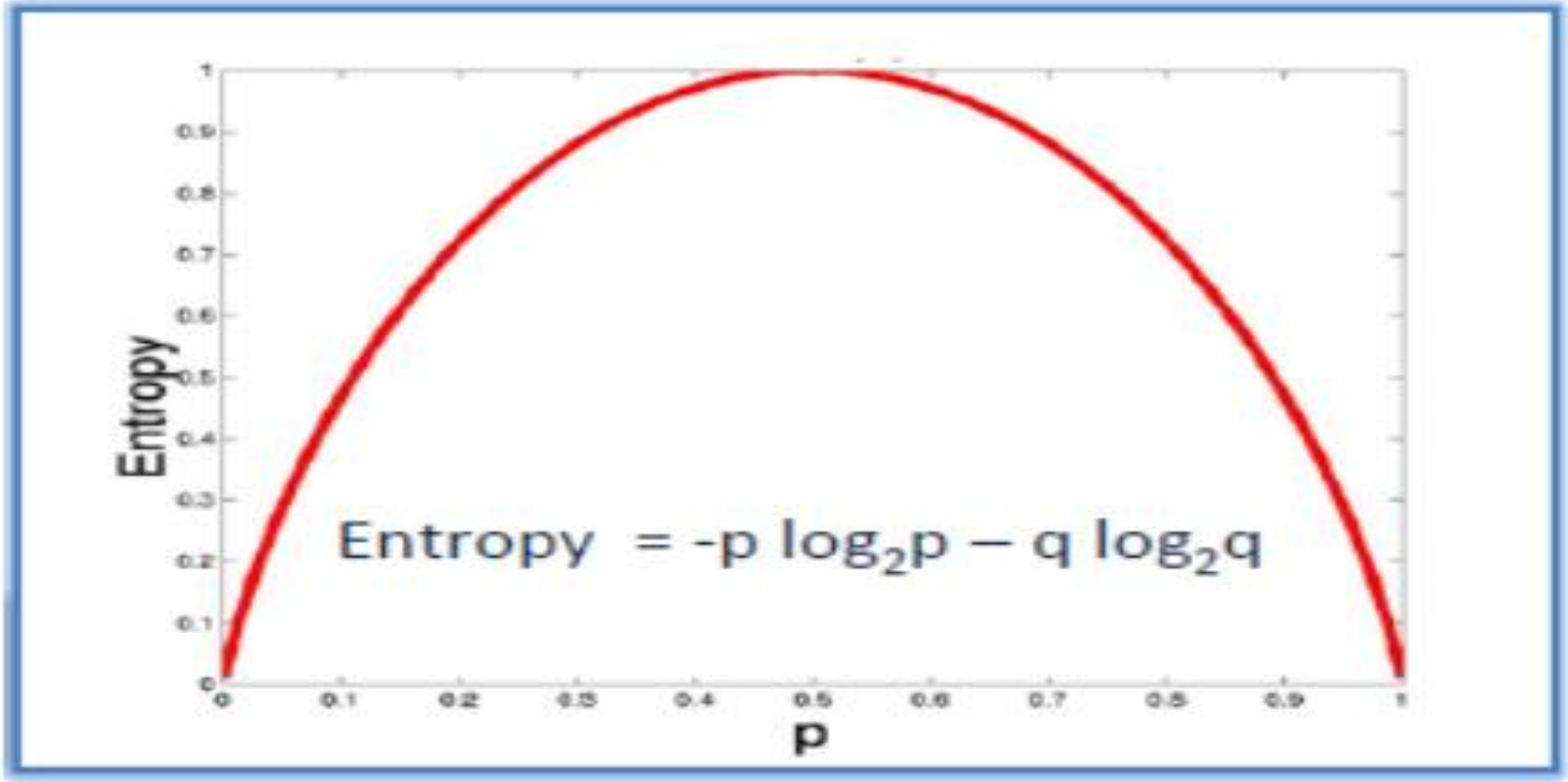
- It **calculates how much information a feature provides** us about a class.

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes})\log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$$

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**



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

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

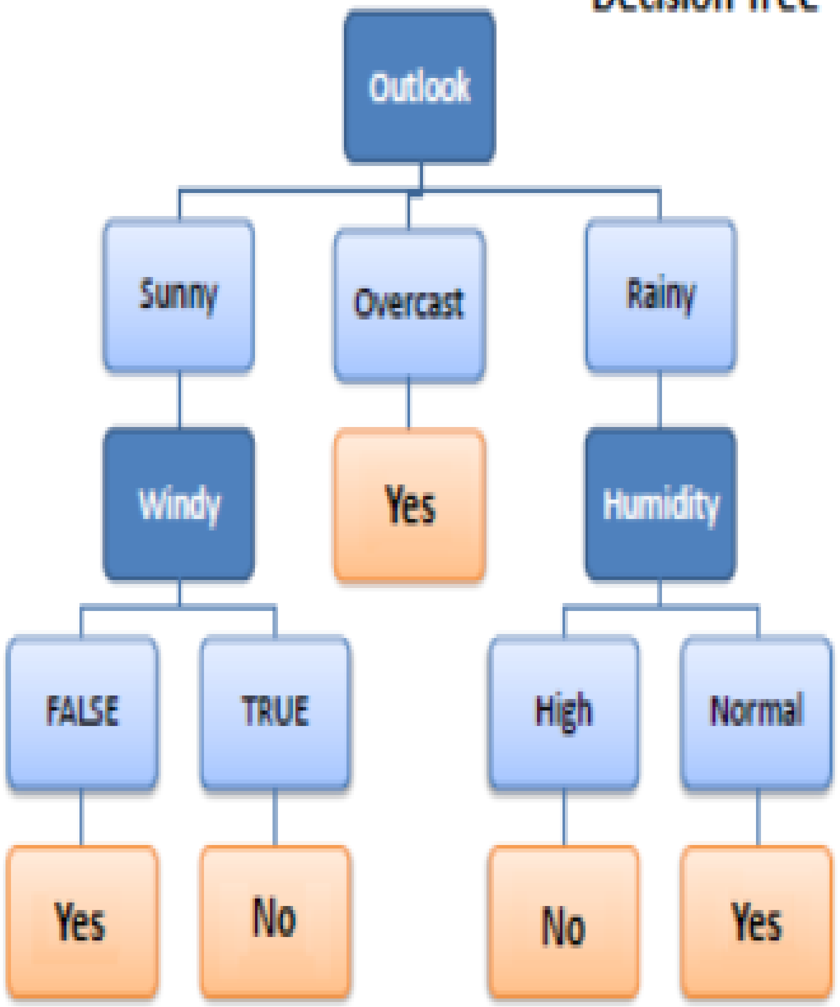
$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

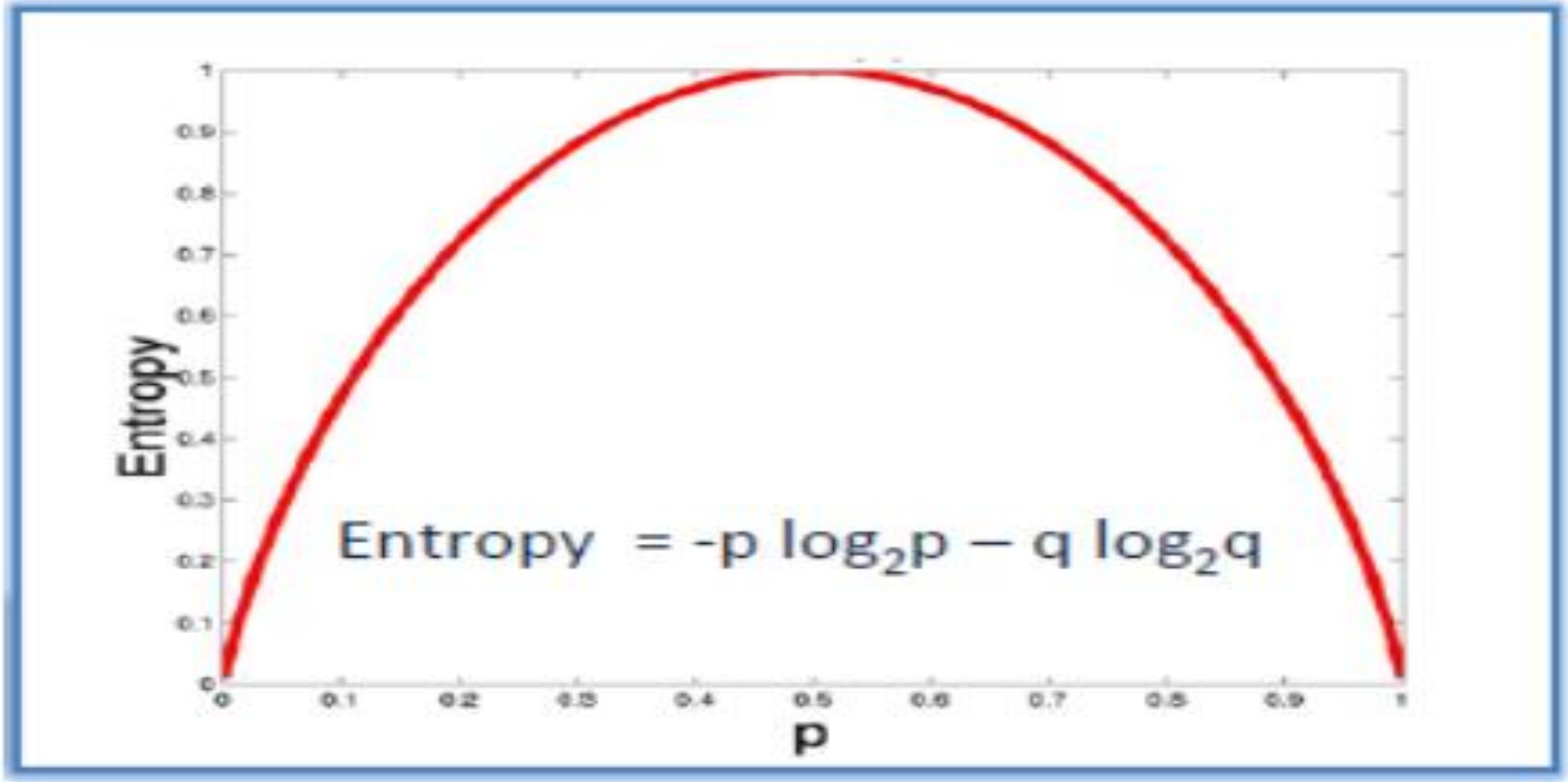
- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

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

Decision Tree





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

a) Entropy using the frequency table of one attribute:

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

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

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

= 0.94

b) Entropy using the frequency table of two attributes:

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

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

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

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

= 0.693

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

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

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

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

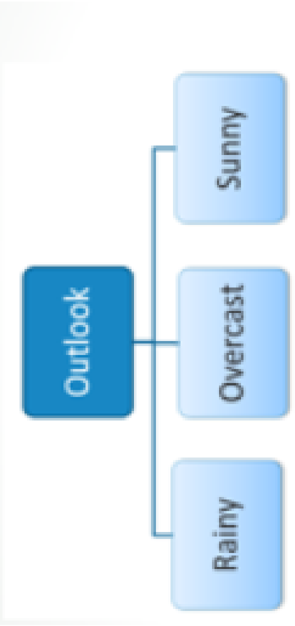
$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

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

= 0.940 – 0.693 = 0.247

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

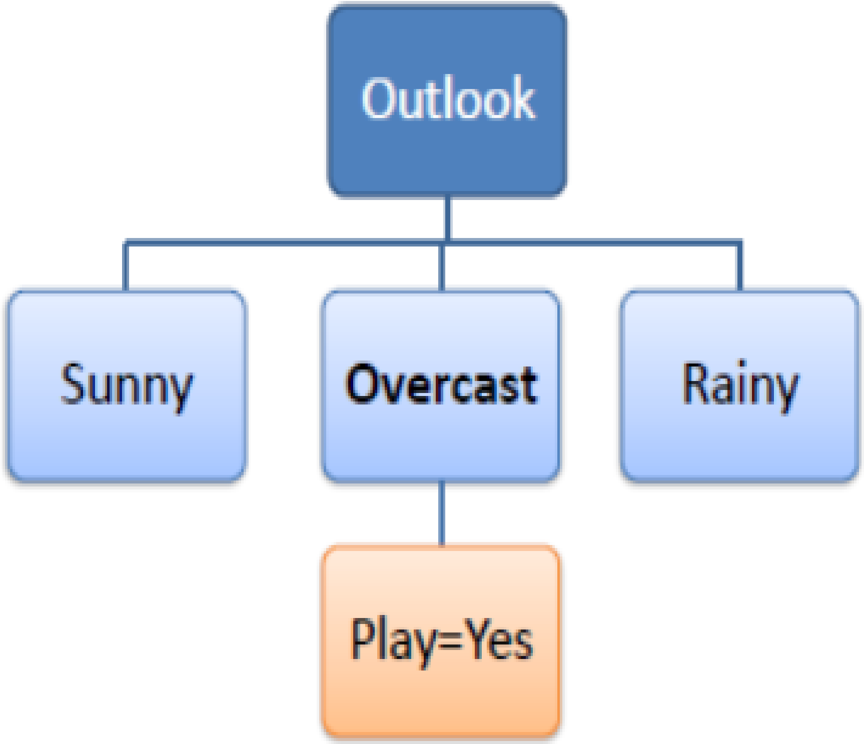
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



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

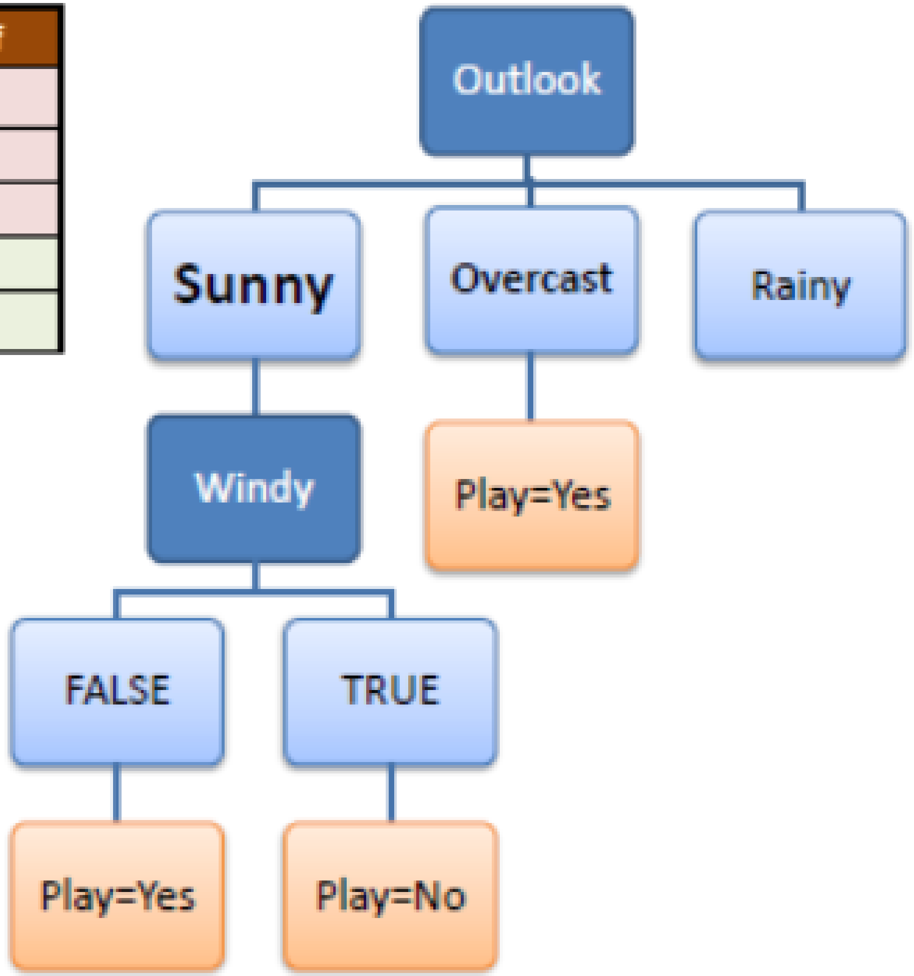
Step 4a: A branch with entropy of 0 is a leaf node.

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



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

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



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

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

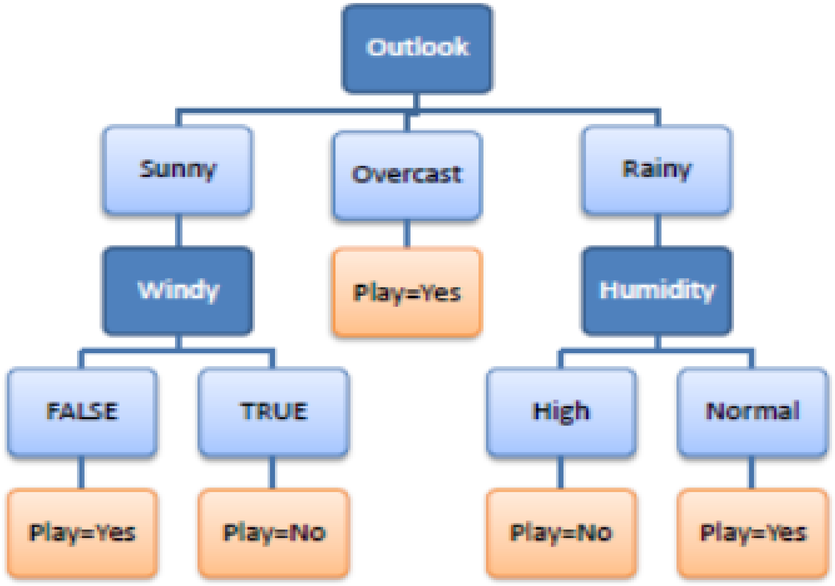
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

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

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

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

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



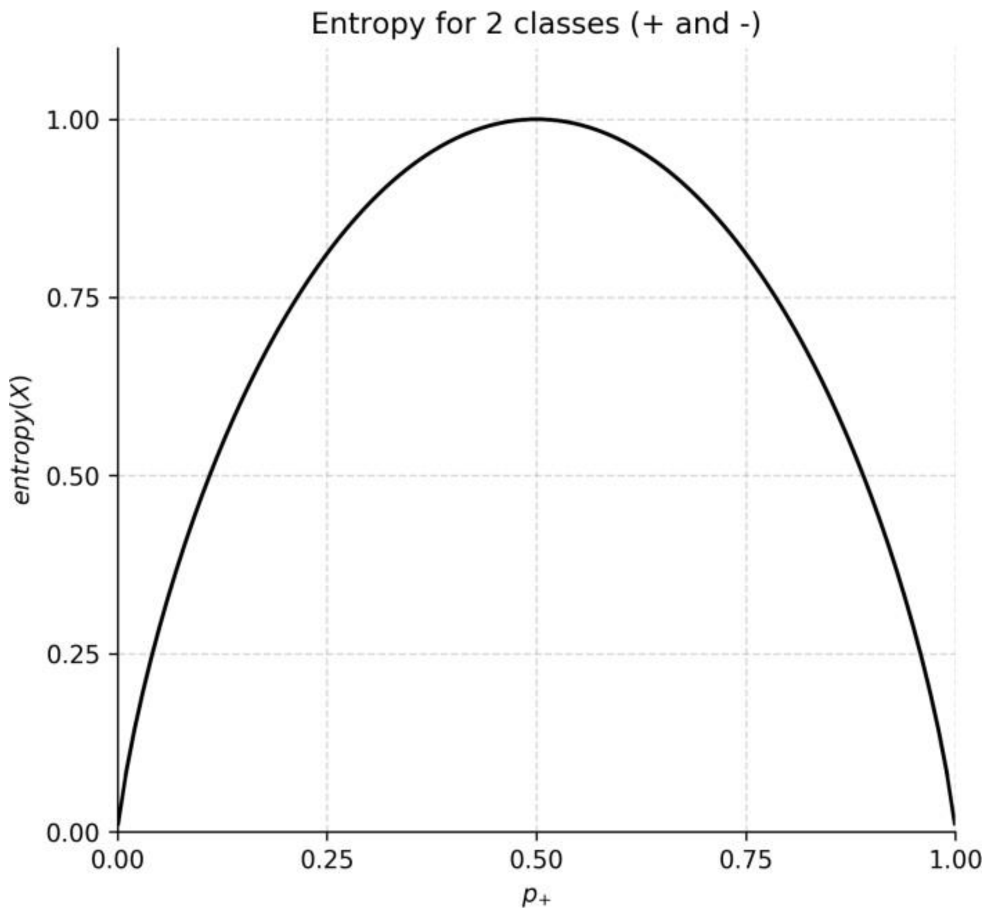
Homework

ID	Fever	Cough	Breathing issues	Infected
1	NO	NO	NO	NO
2	YES	YES	YES	YES
3	YES	YES	NO	NO
4	YES	NO	YES	YES
5	YES	YES	YES	YES
6	NO	YES	NO	NO
7	YES	NO	YES	YES
8	YES	NO	YES	YES
9	NO	YES	YES	YES
10	YES	YES	NO	YES
11	NO	YES	NO	NO
12	NO	YES	YES	YES
13	NO	YES	YES	NO
14	YES	YES	NO	NO

Radom Forest

Entropy

- Entropy measures the degree of randomness in data



- For a set of samples X with k classes:

$$entropy(X) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

where p_i is the proportion of elements of class i

- Lower entropy implies greater predictability!

Information Gain

- The information gain of an attribute a is the expected reduction in entropy due to splitting on values of a:

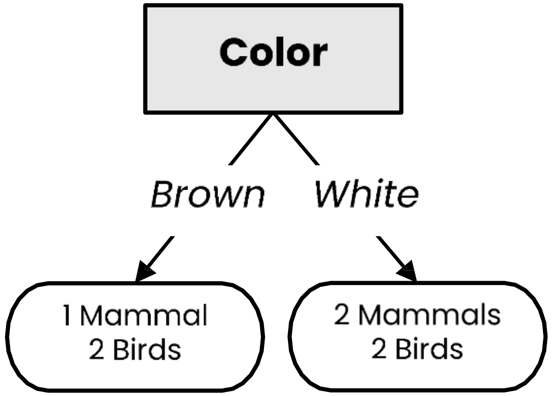
$$gain(X, a) = entropy(X) - \sum_{v \in Values(a)} \frac{|X_v|}{|X|} entropy(X_v)$$

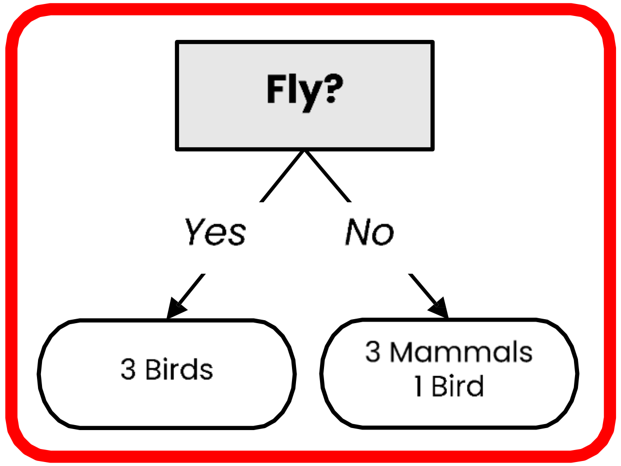
where X_v is the subset of X for which $a = v$

Best attribute = highest information gain

In practice, we compute $entropy(X)$ only once!

Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**





$$entropy\,(X) = -p_{\mathrm{mammal}}\log_2 p_{\mathrm{mammal}} - p_{\mathrm{bird}}\log_2 p_{\mathrm{bird}} = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} \approx 0.985$$

$$entropy\,(X_{color=brown}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} \approx 0.918 \qquad entropy\,(X_{color=white}) = 1$$

$$\boldsymbol{gain\,(X, color) = 0.985 - \frac{3}{7}\cdot 0.918 - \frac{4}{7}\cdot 1 \approx 0.020}$$

$$entropy\,(X_{fly=yes}) = 0 \qquad entropy\,(X_{fly=no}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} \approx 0.811$$

$$\boldsymbol{gain\,(X, fly) = 0.985 - \frac{3}{7}\cdot 0 - \frac{4}{7}\cdot 0.811 \approx \boxed{0.521}}$$

Gini Impurity

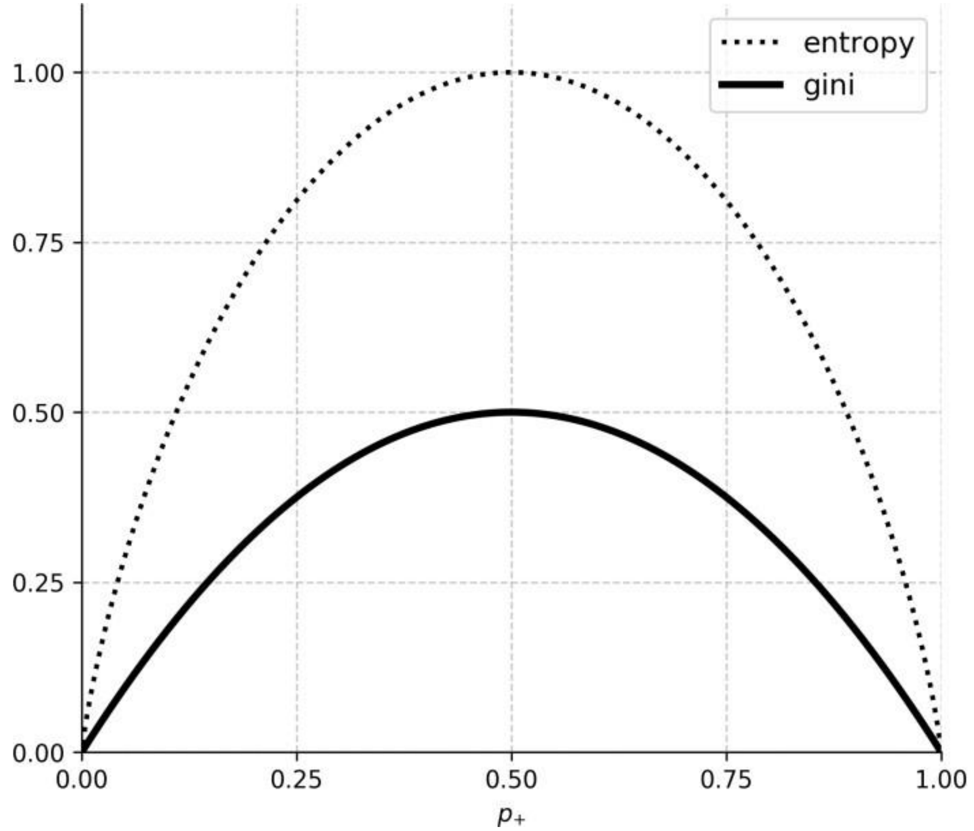
Gini Impurity

- Gini impurity measures how often a randomly chosen example would be incorrectly labeled if it was randomly labeled according to the label distribution

- For a set of samples X with k classes:

$$gini(X) = 1 - \sum_{i=1}^{k} p_i^2$$

where p_i is the proportion of elements of class i

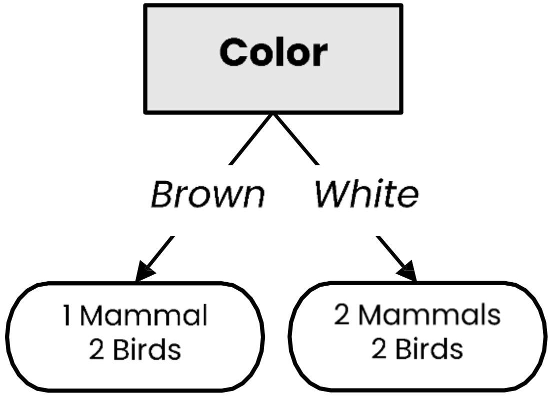


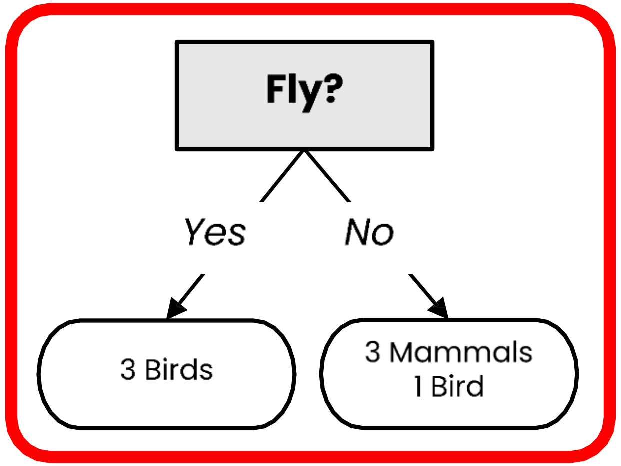
- Can be used as an alternative to entropy for selecting attributes!

Best attribute = highest impurity decrease

In practice, we compute $gini(X)$ only once!

Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**





$$gini\ (X) = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 \approx 0.489$$

$$gini\ (X_{color=brown}) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 \approx 0.444 \qquad gini\ (X_{color=white}) = 0.5$$

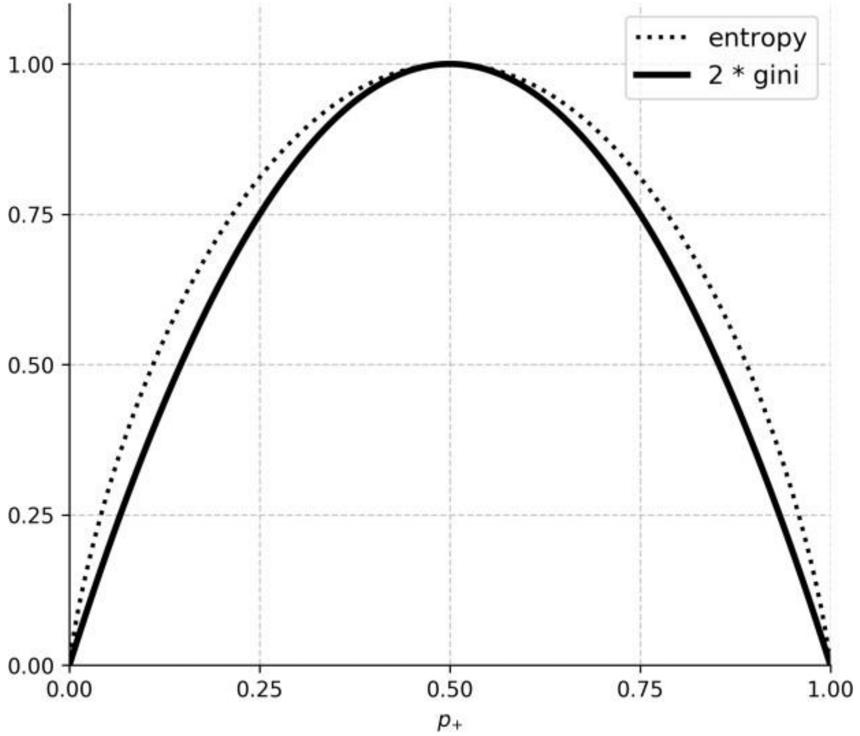
$$\triangle\, \boldsymbol{gini\ (X, color) = 0.489 - \frac{3}{7} \cdot 0.444 - \frac{4}{7} \cdot 0.5 \approx 0.013}$$

$$gini\ (X_{fly=yes}) = 0 \qquad gini\ (X_{fly=no}) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 \approx 0.375$$

$$\triangle\, \boldsymbol{gini\ (X, fly) = 0.489 - \frac{3}{7} \cdot 0 - \frac{4}{7} \cdot 0.375 \approx \boxed{0.274}}$$

Entropy versus Gini Impurity

- Entropy and Gini Impurity give similar results in practice
  - ➢ They only disagree in about 2% of cases
    "Theoretical Comparison between the Gini Index and Information Gain Criteria" [Răileanu & Stoffel, AMAI 2004]
  - ➢ Entropy might be slower to compute, because of the log



Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

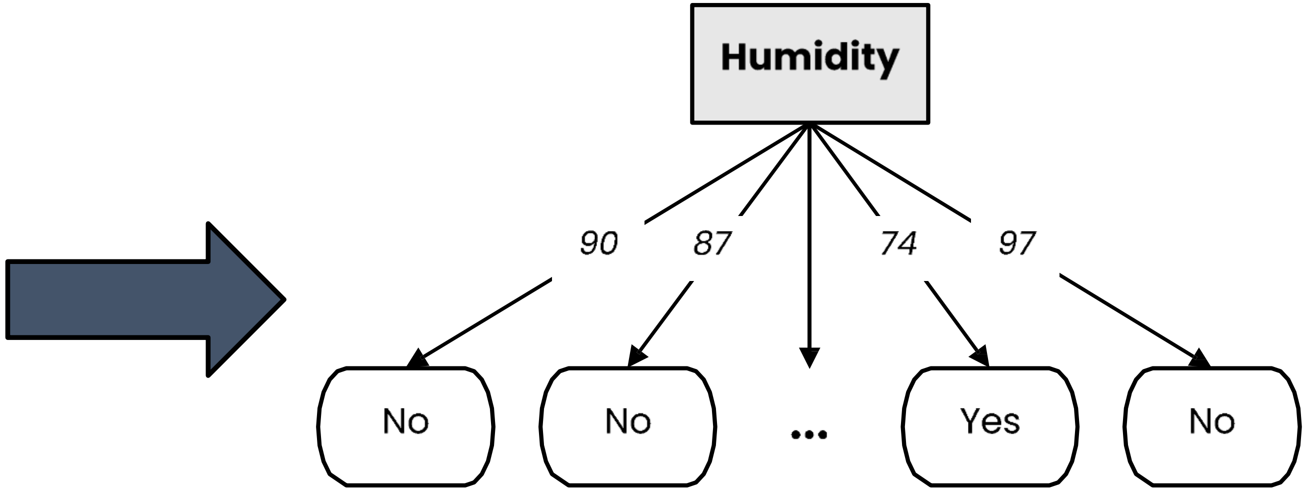
Handling Numerical Attributes

Handling numerical attributes

- How does the ID3 algorithm handle numerical attributes?
  - Any numerical attribute would almost always bring entropy down to zero
  - This means it will completely overfit the training data

Consider a numerical value for humidity

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	90	Weak	**No**
Sunny	Hot	87	Strong	**No**
Overcast	Hot	93	Weak	**Yes**
Rainy	Mild	89	Weak	**Yes**
Rainy	Cool	79	Weak	**Yes**
Rainy	Cool	59	Strong	**No**
Overcast	Cool	77	Strong	**Yes**
Sunny	Mild	91	Weak	**No**
Sunny	Cool	68	Weak	**Yes**
Rainy	Mild	80	Weak	**Yes**
Sunny	Mild	72	Strong	**Yes**
Overcast	Mild	96	Strong	**Yes**
Overcast	Hot	74	Weak	**Yes**
Rainy	Mild	97	Strong	**No**



Handling numerical attributes

- Numerical attributes have to be treated differently
  - ➢ Find the best splitting value

$$gain(X, a, t) = entropy(X) - \frac{|X_{a\leq t}|}{|X|} entropy(X_{a\leq t}) - \frac{|X_{a>t}|}{|X|} entropy(X_{a>t})$$

Humidity	Play Tennis?
90	No
87	No
93	Yes
89	Yes
79	Yes
59	No
77	Yes
91	No
68	Yes
80	Yes
72	Yes
96	Yes
74	Yes
97	No

Sort

Humidity	Play Tennis?
59	No
68	Yes
72	Yes
74	Yes
77	Yes
79	Yes
80	Yes
87	No
89	Yes
90	No
91	No
93	Yes
96	Yes
97	No

Mean of each consecutive pair

Candidate split values
63
70
73
75.5
78
79.5
83.5
88
89.5
90.5
92
94.5
96.5

$$gain\,(X, humidity, 83.5) =$$

Handling numerical attributes

- Numerical attributes have to be treated differently
  - Find the best splitting value

$$gain(X, a, t) = \boxed{entropy(X)} - \frac{|X_{a \le t}|}{|X|} entropy(X_{a \le t}) - \frac{|X_{a > t}|}{|X|} entropy(X_{a > t})$$

Humidity	Play Tennis?
90	No
87	No
93	Yes
89	Yes
79	Yes
59	No
77	Yes
91	No
68	Yes
80	Yes
72	Yes
96	Yes
74	Yes
97	No

Sort

Humidity	Play Tennis?
59	No
68	Yes
72	Yes
74	Yes
77	Yes
79	Yes
80	Yes
87	No
89	Yes
90	No
91	No
93	Yes
96	Yes
97	No

Mean of each consecutive pair

Candidate split values
63
70
73
75.5
78
79.5
83.5
88
89.5
90.5
92
94.5
96.5

$$gain\ (X, humidity, 83.5) = 0.94$$

Handling numerical attributes

- Numerical attributes have to be treated differently
  - ➢ Find the best splitting value

Humidity	Play Tennis?
90	No
87	No
93	Yes
89	Yes
79	Yes
59	No
77	Yes
91	No
68	Yes
80	Yes
72	Yes
96	Yes
74	Yes
97	No

Sort

Humidity	Play Tennis?
59	No
68	Yes
72	Yes
74	Yes
77	Yes
79	Yes
80	Yes
87	No
89	Yes
90	No
91	No
93	Yes
96	Yes
97	No

Mean of each consecutive pair

Candidate split values
63
70
73
75.5
78
79.5
83.5
88
89.5
90.5
92
94.5
96.5

Gain for every candidate

Information gain
0.113
0.01
0.0004
0.015
0.045
0.09
0.152
0.048
0.102
0.025
0.0004
0.01
0.113

83.5 is the best splitting value with an information gain of 0.152

Handling Missing Values

Handling missing values at training time

Does it fly?	Color	Class
No	***White***	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**

- Data sets might have samples with missing values for some attributes
- Simply ignoring them would mean throwing away a lot of information
- There are better ways of handling missing values:
  - ➢ Set them to the most common value
  - ➢ Set them to the most probable value given the label

$$P(Yes|Bird) = \frac{2}{3} = 0.66$$

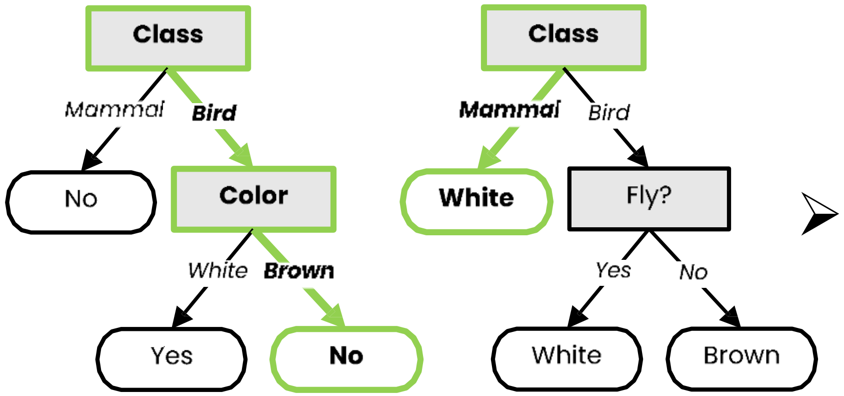
$$P(No|Bird) = \frac{1}{3} = 0.33$$

$$P(Brown|Mammal) = 0$$

$$P(White|Mammal) = 1$$

Handling missing values at training time

Does it fly?	Color	Class
No	***White***	**Mammal**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**



- Data sets might have samples with missing values for some attributes
- Simply ignoring them would mean throwing away a lot of information
- There are better ways of handling missing values:

➢ Set them to the most common value

➢ Set them to the most probable value given the label

➢ Add a new instance for each possible value

➢ Leave them unknown, but discard the sample when evaluating the gain of that attribute

(if the attribute is chosen for splitting, send the instances with unknown values to all children)

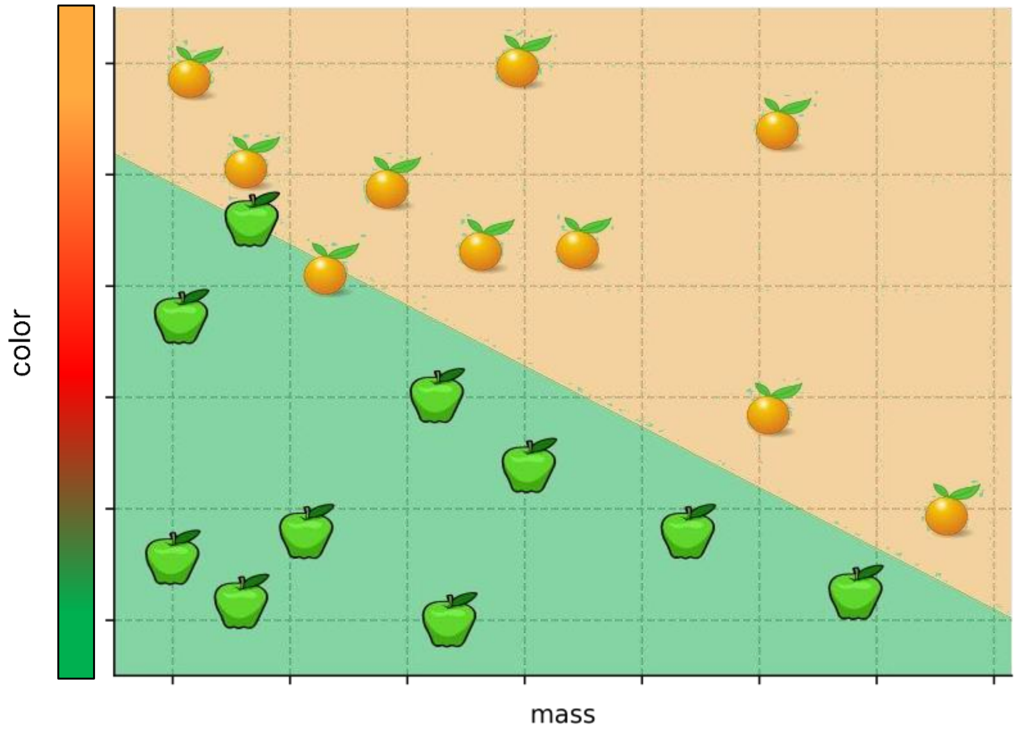
➢ Build a decision tree on all other attributes (including label) to predict missing values

(use instances where the attribute is defined as training data)

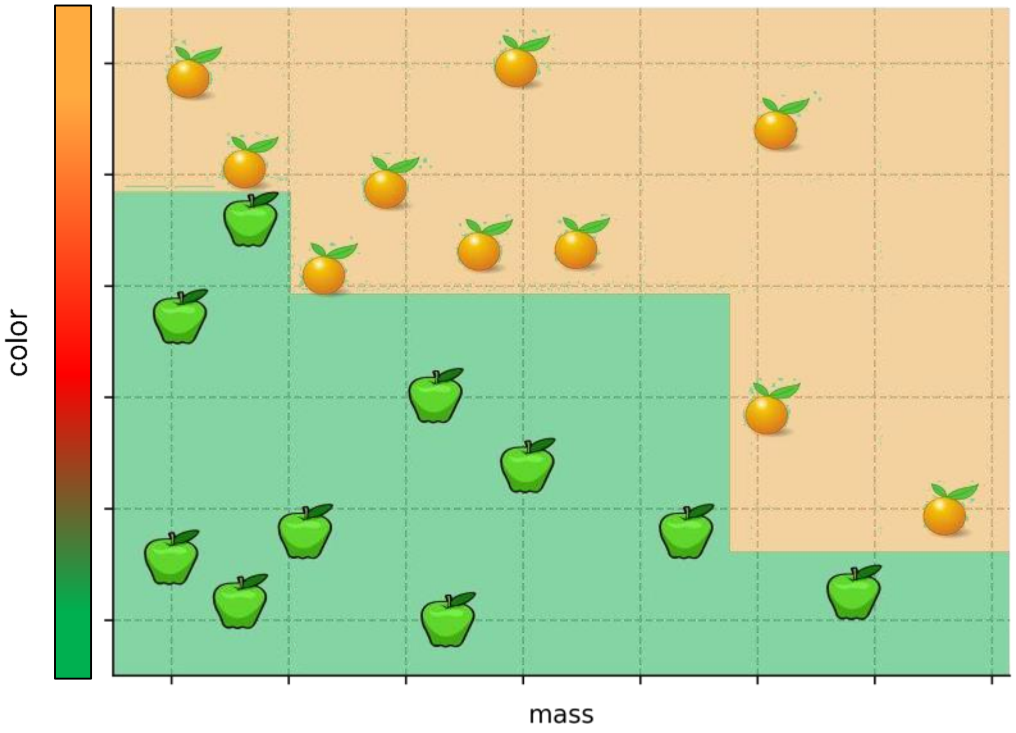
Decision Boundaries

- Decision trees produce non-linear decision boundaries

Support Vector Machines

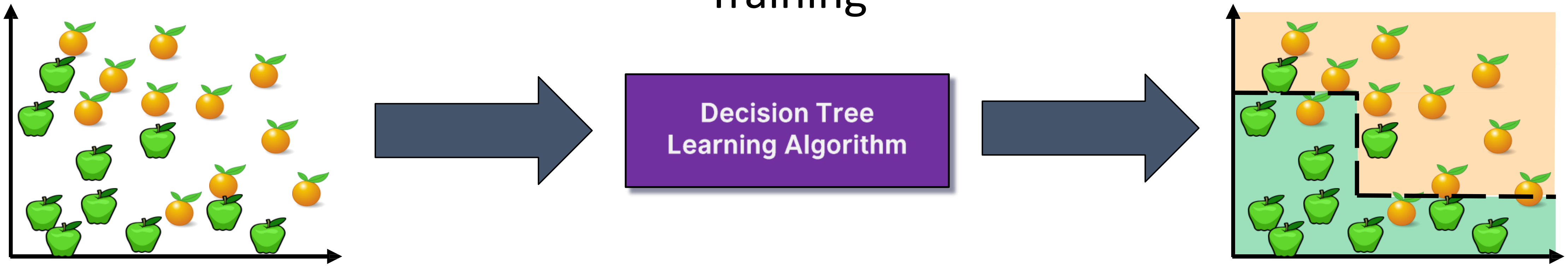


Decision Tree

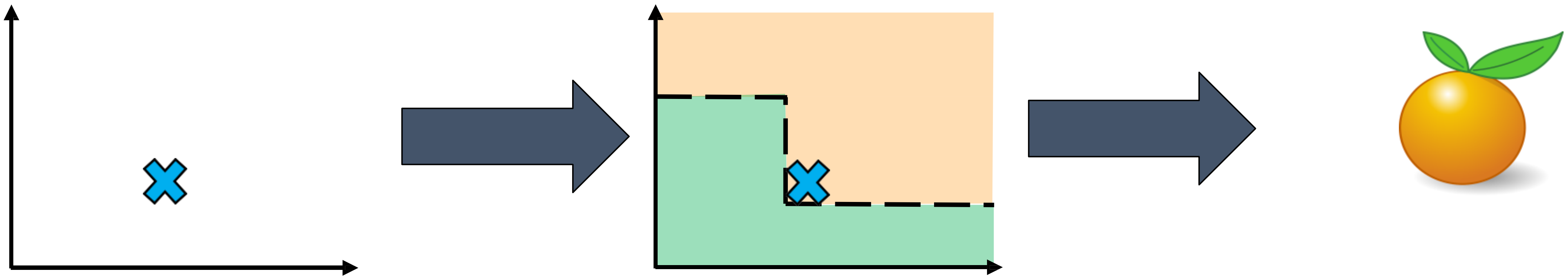


Decision Trees: Training and Inference

Training



Inference



Random Forests

(Ensemble learning with decision trees)

Random Forests

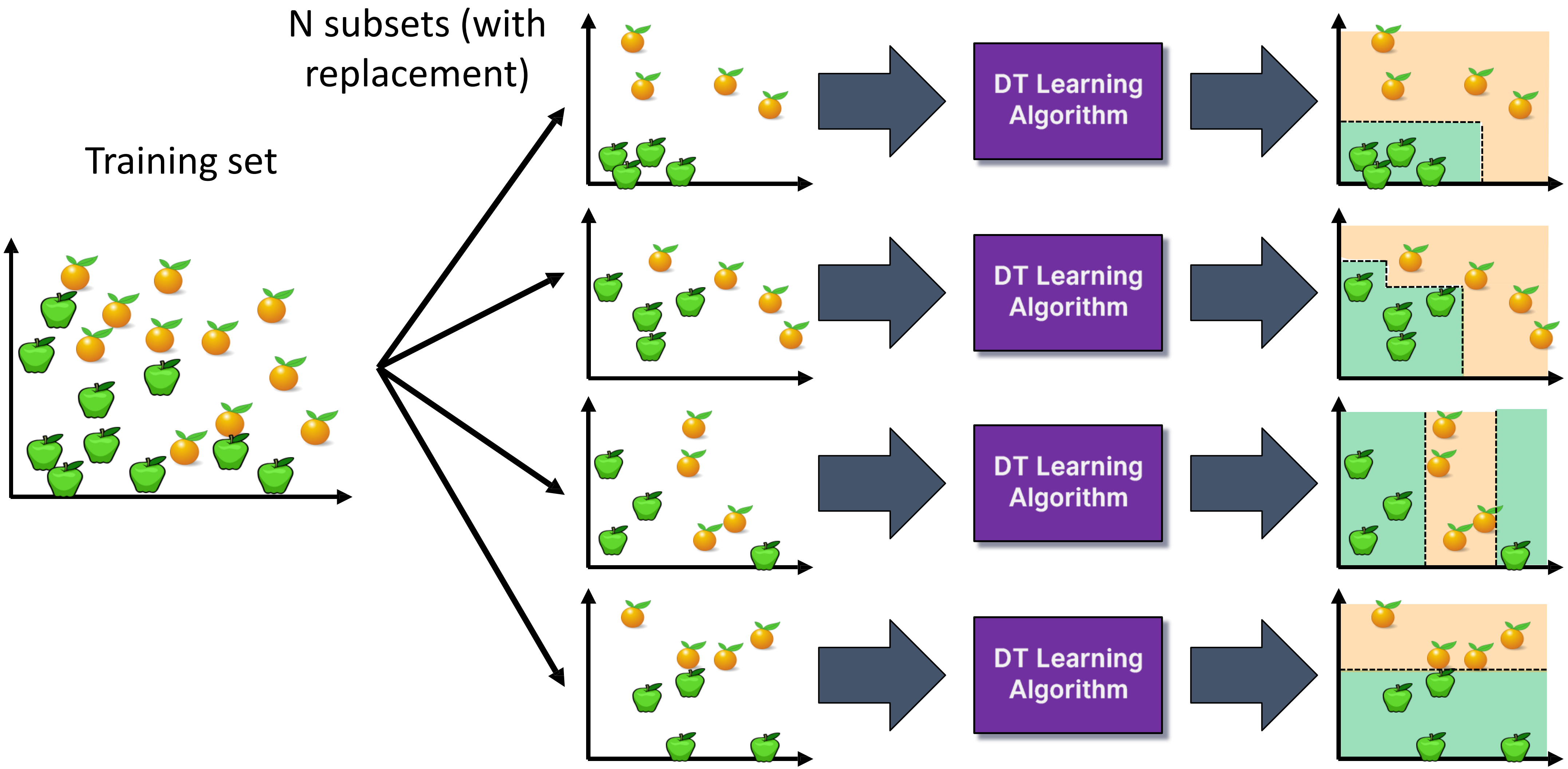
- Random Forests:
  - Instead of building a single decision tree and use it to make predictions, build many slightly different trees and combine their predictions

- We have a single data set, so how do we obtain slightly different trees?

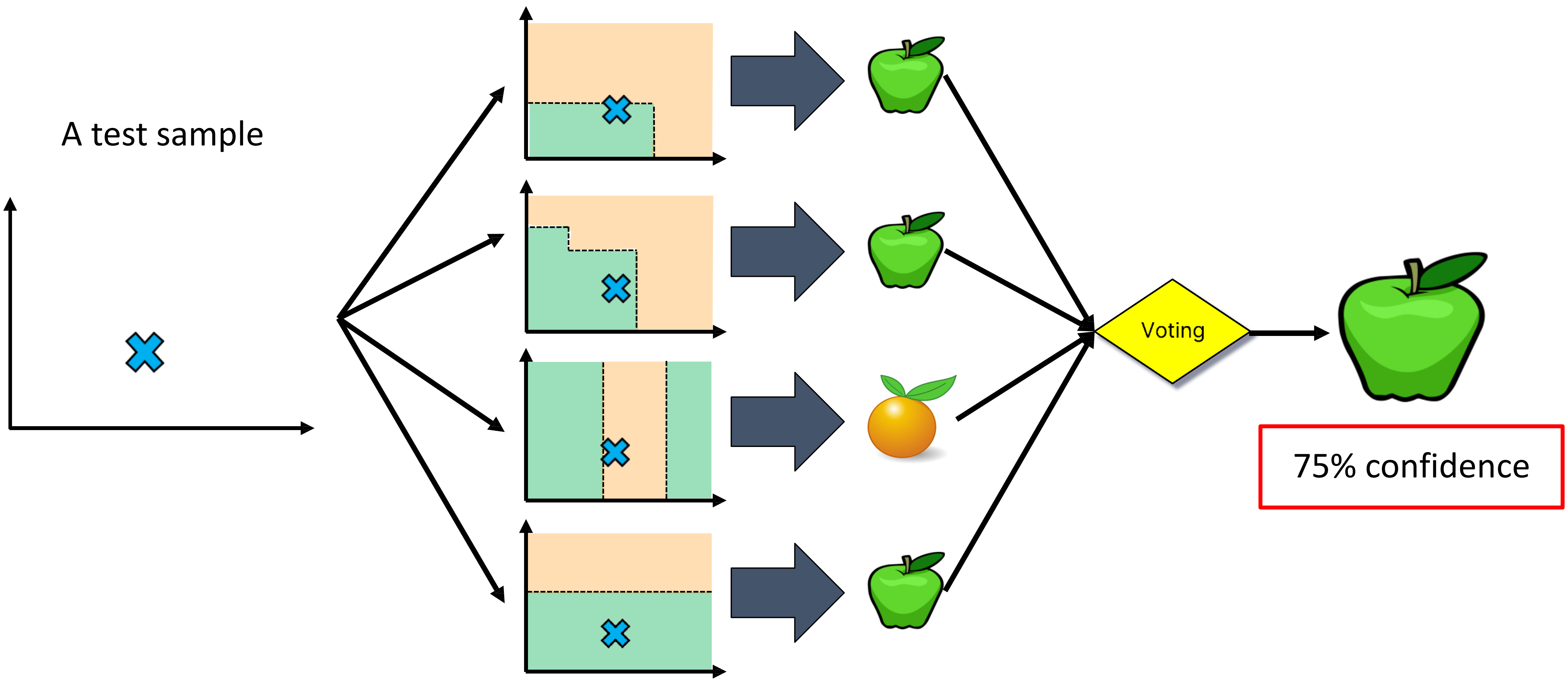
1. Bagging (**B**ootstrap **Agg**regat**ing**):
   - Take random subsets of data points from the training set to create N smaller data sets
   - Fit a decision tree on each subset

2. Random Subspace Method (also known as Feature Bagging):
   - Fit N different decision trees by constraining each one to operate on a random subset of features

Bagging at training time



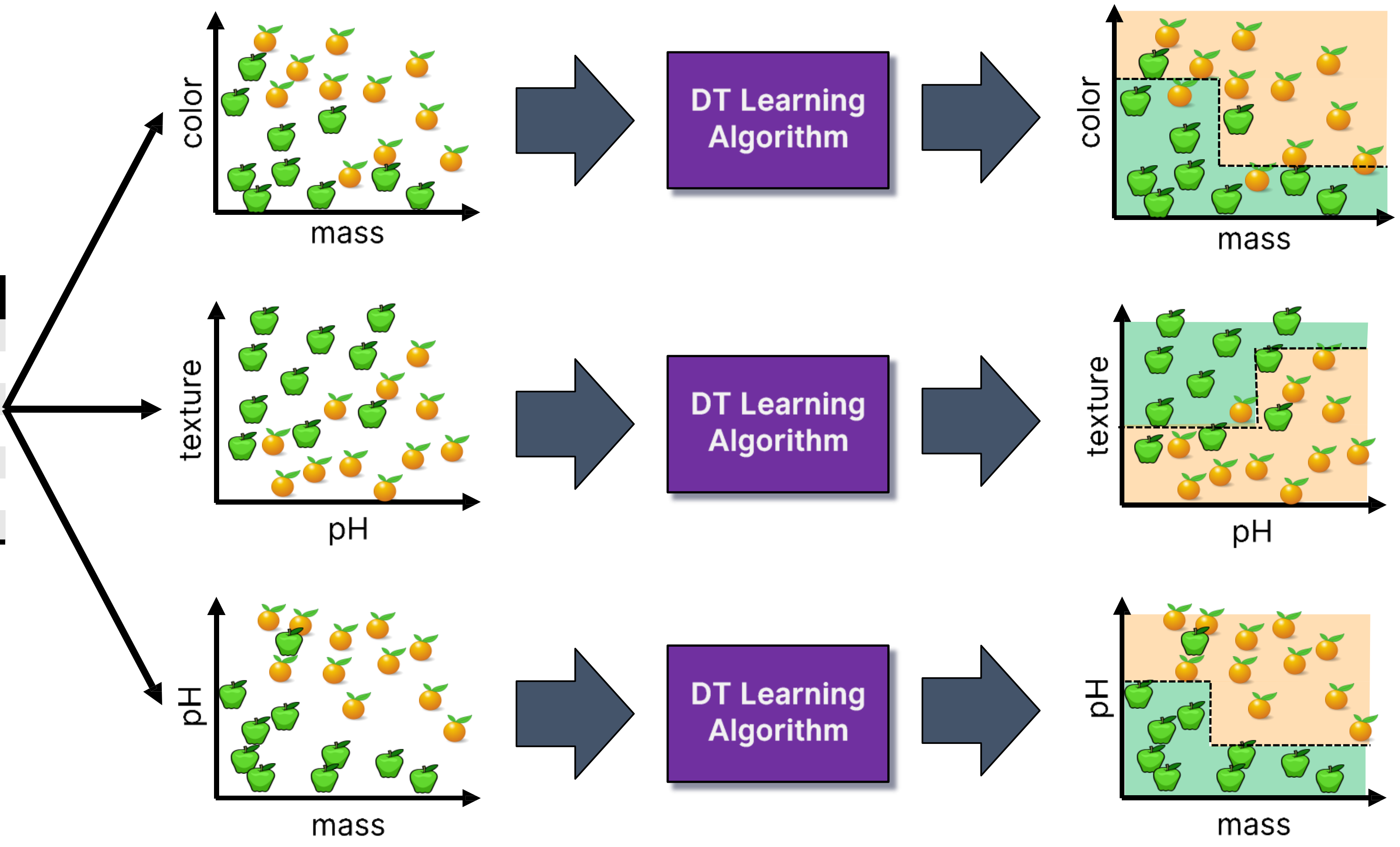
Bagging at inference time



Random Subspace Method at training time

Training data

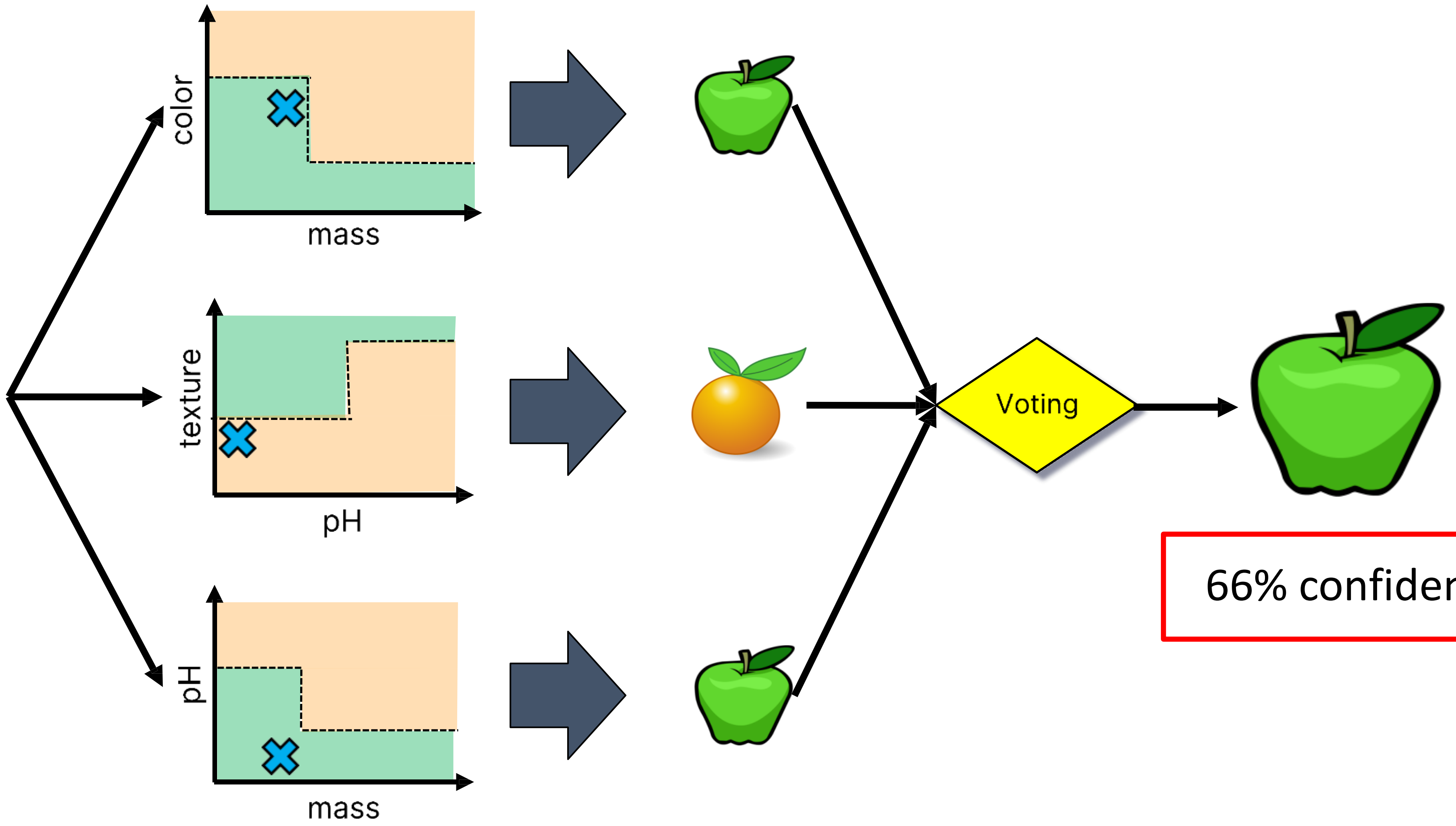
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



Random Subspace Method at inference time

A test sample

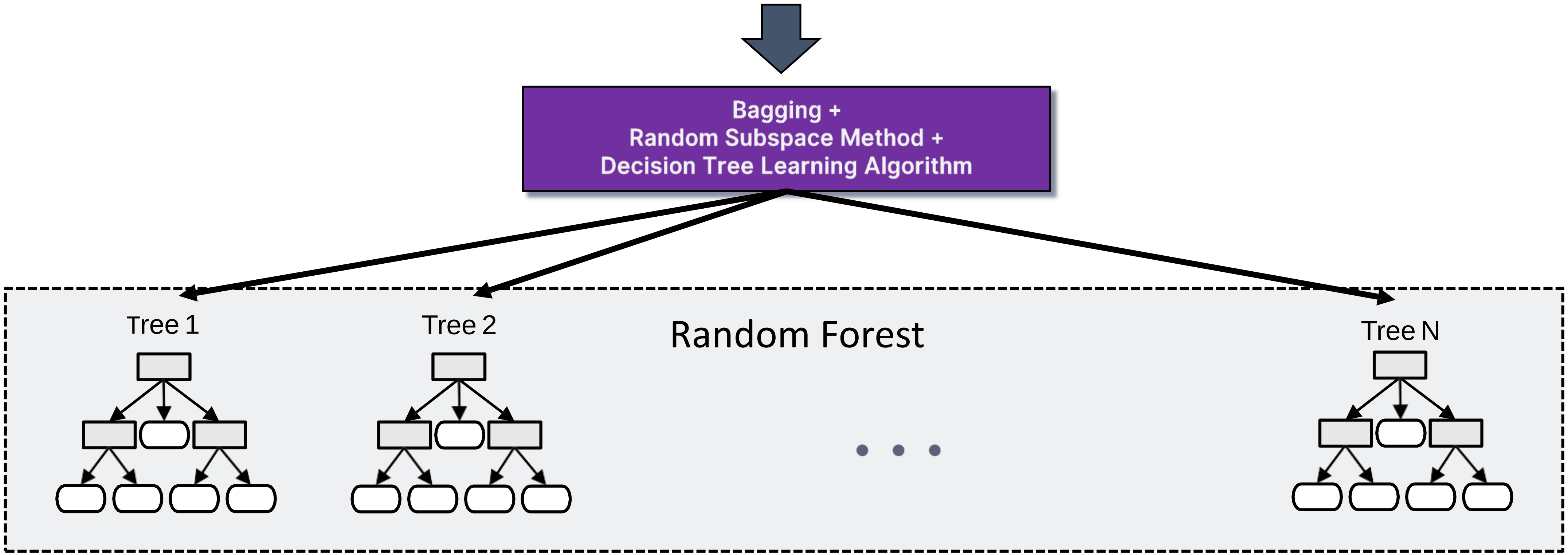
87	Red	Smooth	3.1



66% confidence

Random Forests

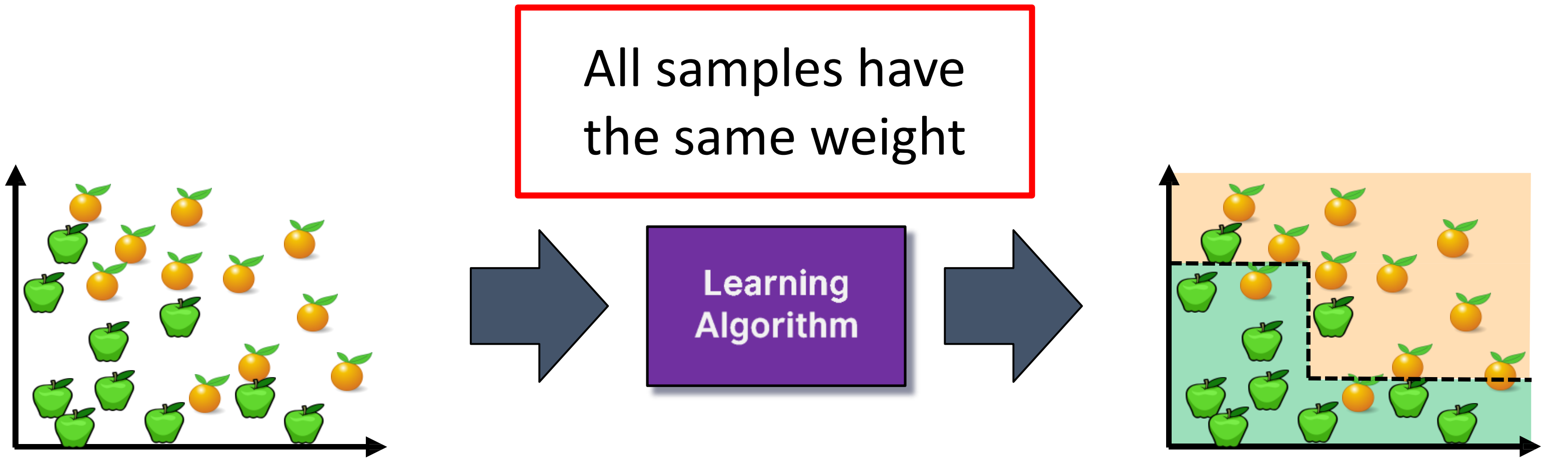
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



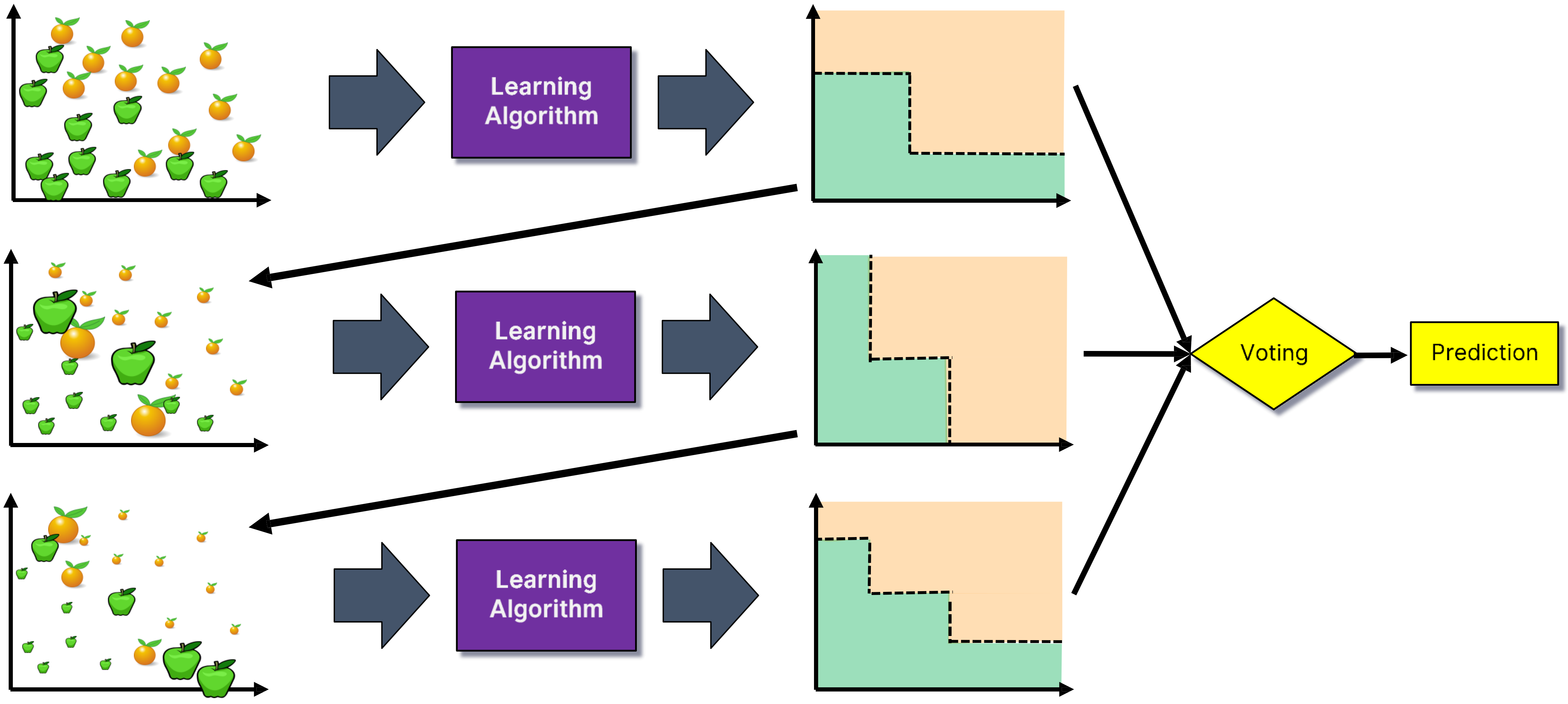
Ensemble Learning

- Ensemble Learning:
  - Method that combines multiple learning algorithms to obtain performance improvements over its components
- **Random Forests** are one of the most common examples of ensemble learning
- Other commonly-used ensemble methods:
  - Bagging: multiple models on random subsets of data samples
  - Random Subspace Method: multiple models on random subsets of features
  - Boosting: train models iteratively, while making the current model focus on the mistakes of the previous ones by increasing the weight of misclassified samples

Boosting

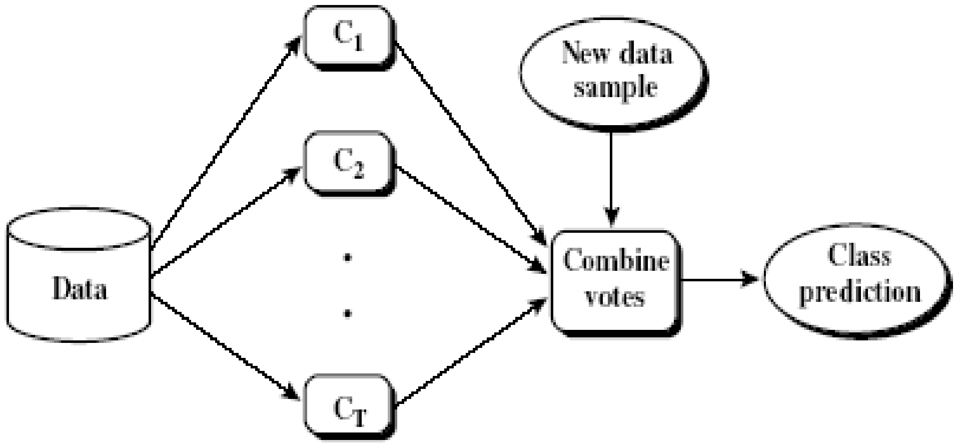


Boosting



Learning Algorithm
Learning Algorithm
Learning Algorithm
Voting
Prediction

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Summary

- Ensemble Learning methods combine multiple learning algorithms to obtain performance improvements over its components

- Commonly-used ensemble methods:
  - Bagging (multiple models on random subsets of data samples)
  - Random Subspace Method (multiple models on random subsets of features)
  - Boosting (train models iteratively, while making the current model focus on the mistakes of the previous ones by increasing the weight of misclassified samples)

- **Random Forests** are an ensemble learning method that employ decision tree learning to build multiple trees through **bagging** and **random subspace method**.
  - They rectify the overfitting problem of decision trees!