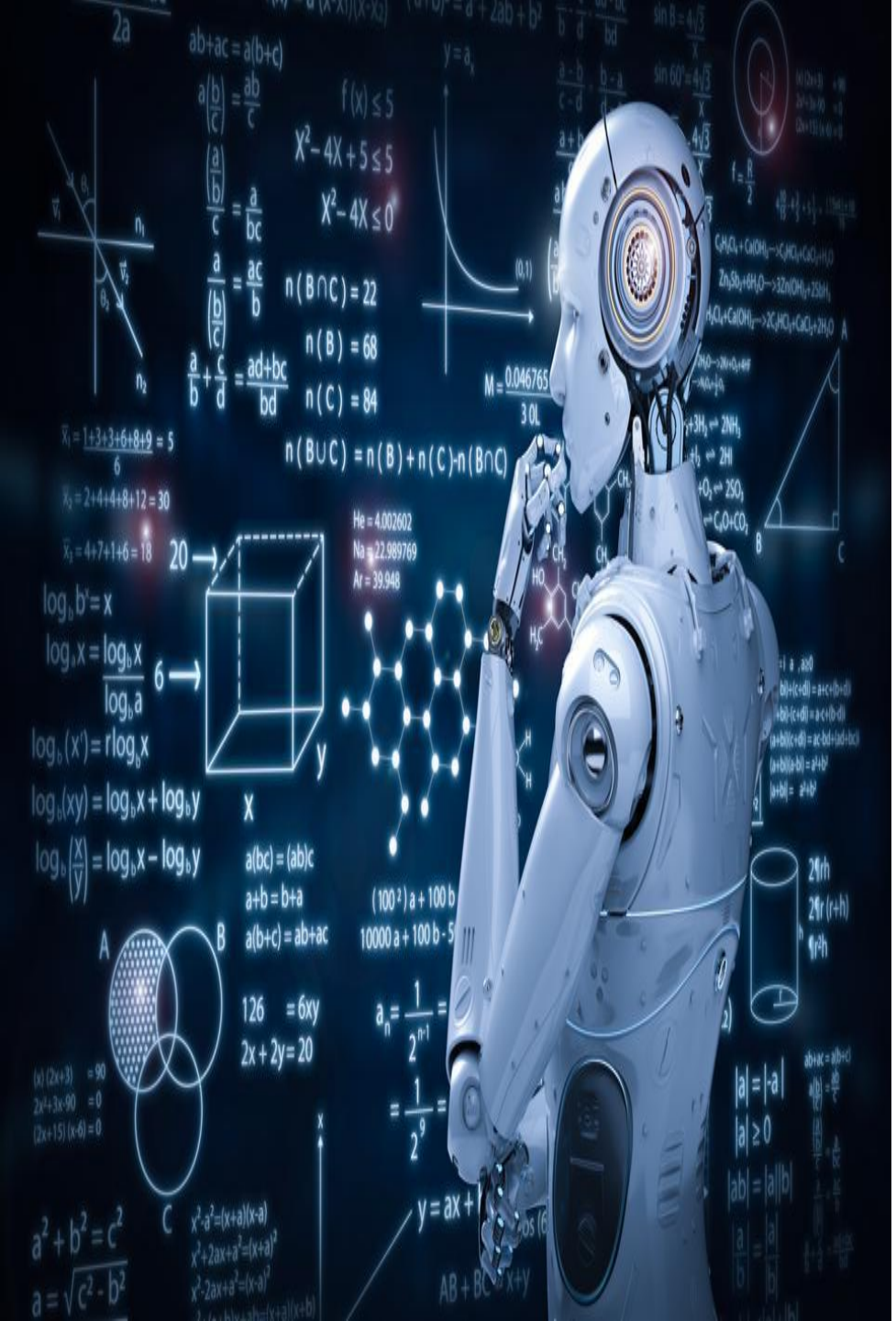


Practical Machine Learning

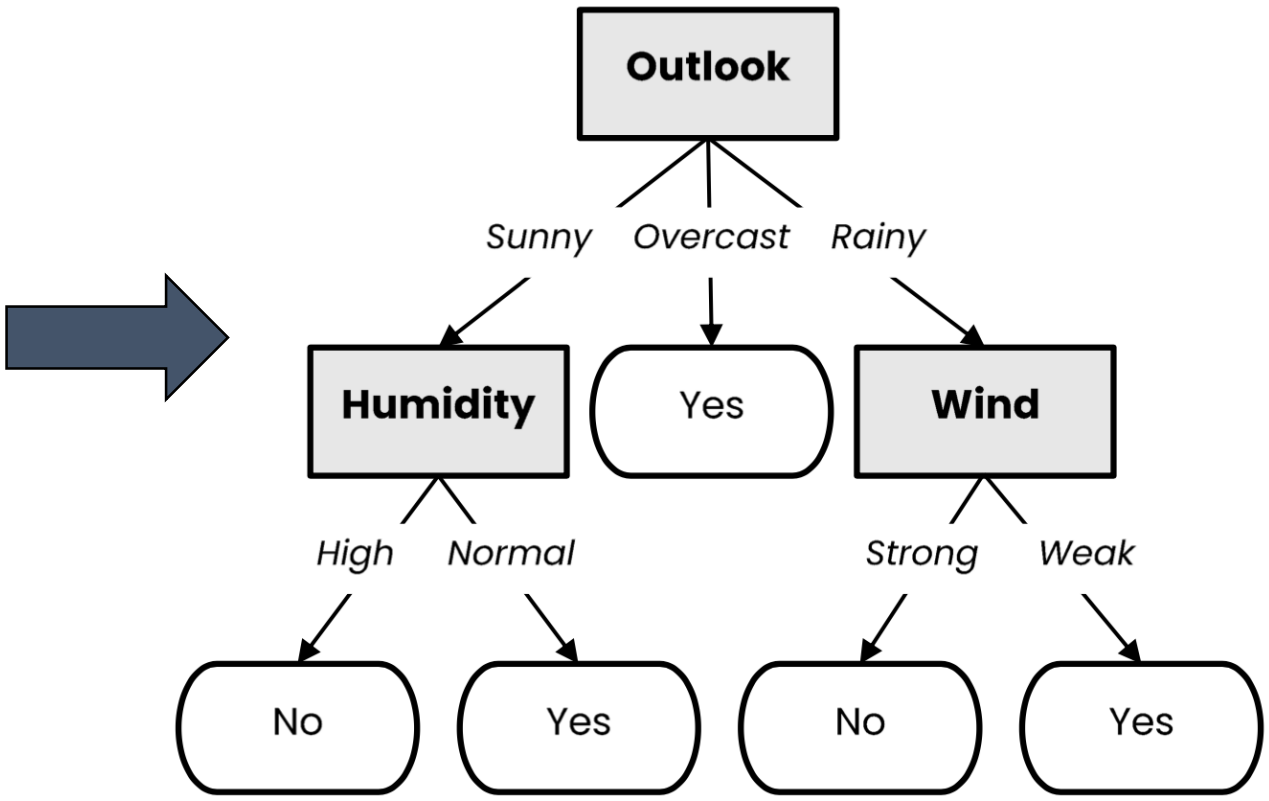
Day 8: SEP23 DBDA

Kiran Waghmare

Decision Tree Learning

- We use labeled data to obtain a suitable decision tree for future predictions
  - ➢ We want a decision tree that works well on unseen data, while asking as few questions as possible

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	**No**
Sunny	Hot	High	Strong	**No**
Overcast	Hot	High	Weak	**Yes**
Rainy	Mild	High	Weak	**Yes**
Rainy	Cool	Normal	Weak	**Yes**
Rainy	Cool	Normal	Strong	**No**
Overcast	Cool	Normal	Strong	**Yes**
Sunny	Mild	High	Weak	**No**
Sunny	Cool	Normal	Weak	**Yes**
Rainy	Mild	Normal	Weak	**Yes**
Sunny	Mild	Normal	Strong	**Yes**
Overcast	Mild	High	Strong	**Yes**
Overcast	Hot	Normal	Weak	**Yes**
Rainy	Mild	High	Strong	**No**



Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
  - ➢ Recursively repeat this step until we can surely decide the label

Outlook = *Sunny*

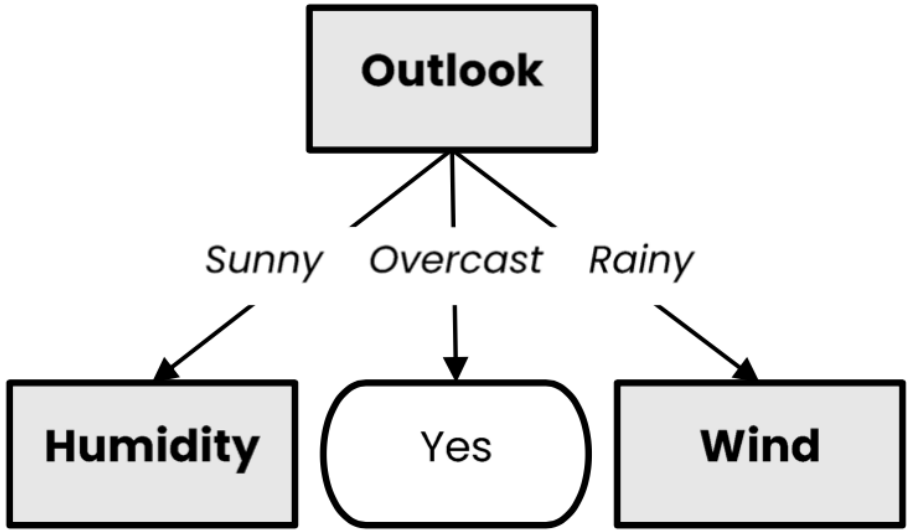
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**No**
Hot	High	Strong	**No**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**
Mild	Normal	Strong	**Yes**

Outlook = *Overcast*

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Mild	High	Strong	**Yes**
Hot	Normal	Weak	**Yes**

Outlook = *Rainy*

Temperature	Humidity	Wind	Play Tennis?
Mild	High	Weak	**Yes**
Cool	Normal	Weak	**Yes**
Cool	Normal	Strong	**No**
Mild	Normal	Weak	**Yes**
Mild	High	Strong	**No**



Decision Tree Learning

- Basic step: choose an attribute and, based on its values, split the data into smaller sets
  - ➢ Recursively repeat this step until we can surely decide the label

Outlook = *Sunny*

Humidity = High

Temperature	Wind	Play Tennis?
Hot	Weak	**No**
Hot	Strong	**No**
Mild	Weak	**No**

Humidity = Normal

Temperature	Wind	Play Tennis?
Cool	Weak	**Yes**
Mild	Strong	**Yes**

Outlook = *Overcast*

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Mild	High	Strong	**Yes**
Hot	Normal	Weak	**Yes**

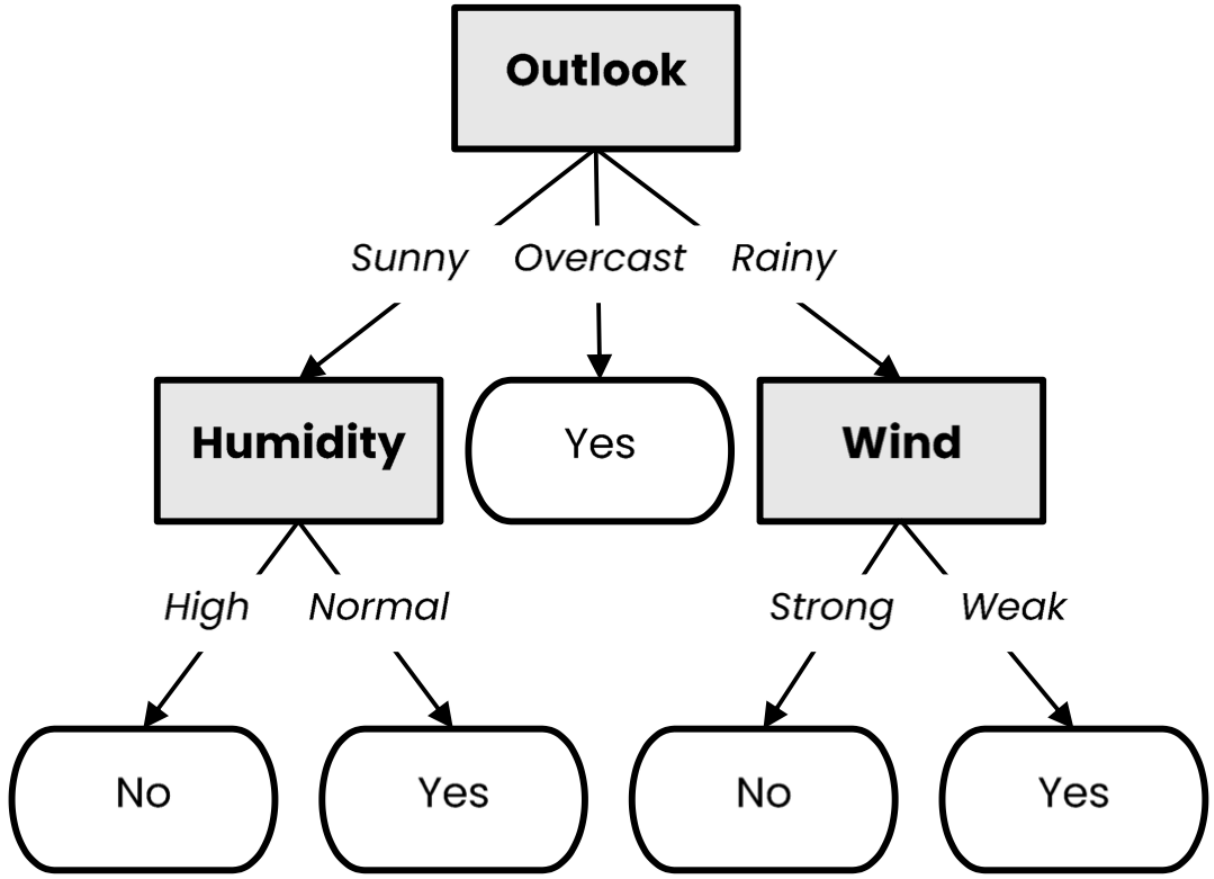
Outlook = *Rainy*

Wind = Strong

Temperature	Humidity	Play Tennis?
Cool	Normal	**No**
Mild	High	**No**

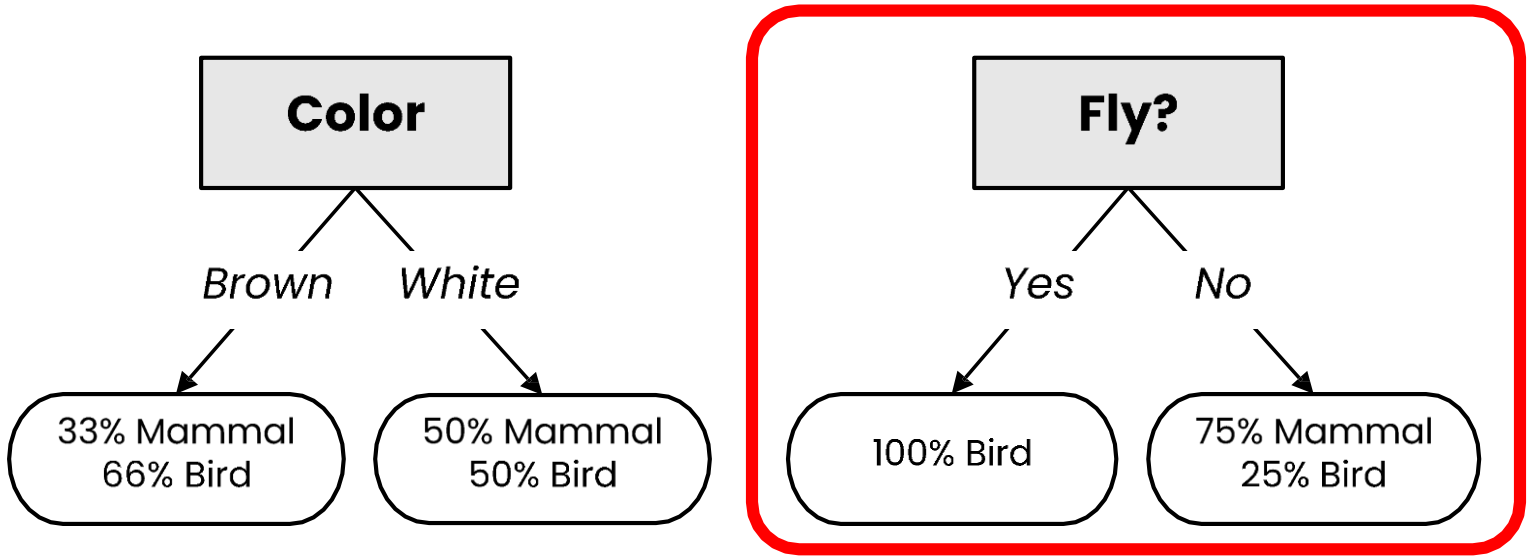
Wind = Weak

Temperature	Humidity	Play Tennis?
Mild	High	**Yes**
Cool	Normal	**Yes**
Mild	Normal	**Yes**



What is a good attribute?

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**

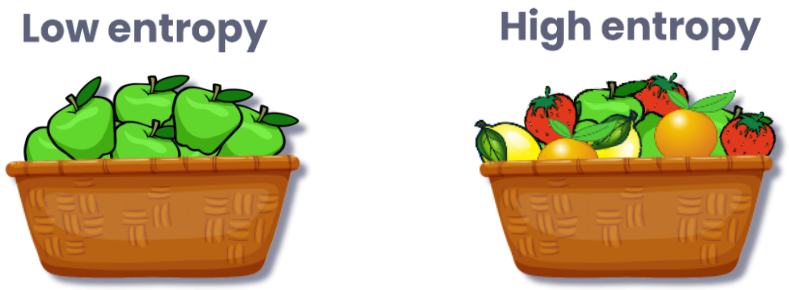


- Which attribute provides better splitting?
- Why?
  - Because the resulting subsets are more pure
  - Knowing the value of this attribute gives us more information about the label
    (the entropy of the subsets is lower)

Information Gain

Entropy

- Entropy measures the degree of randomness in data

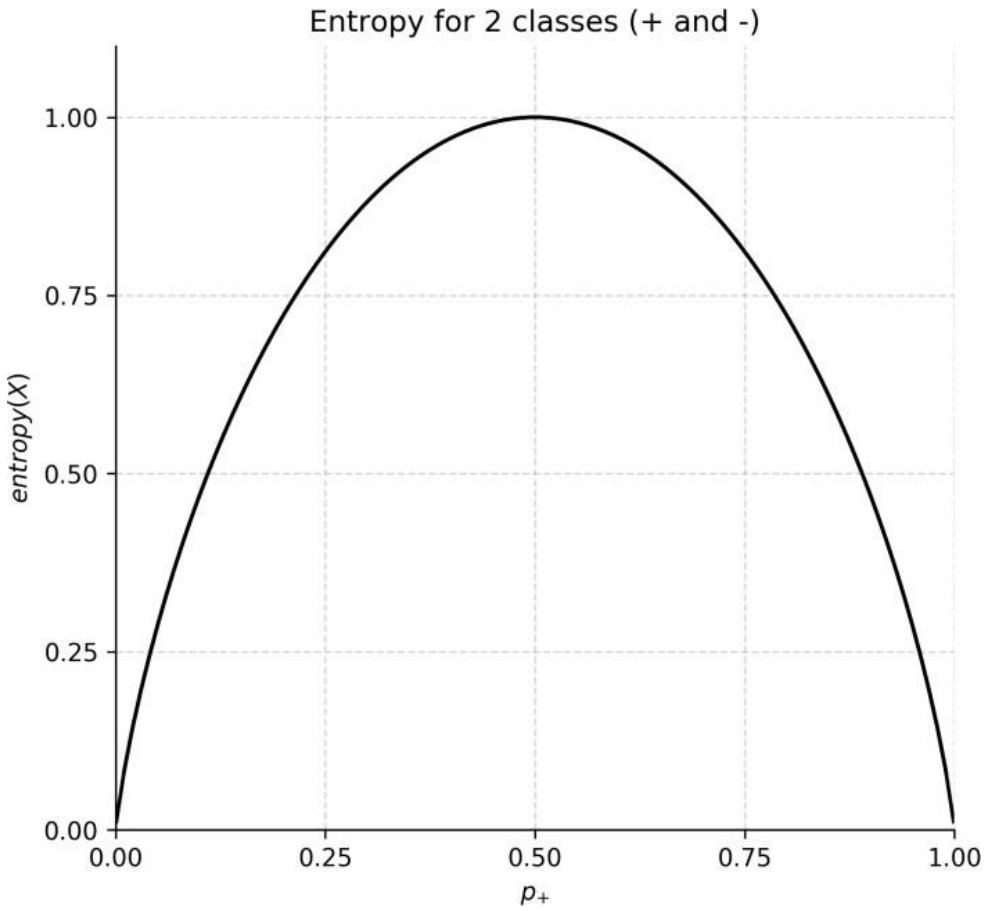


- For a set of samples X with k classes:

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

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:

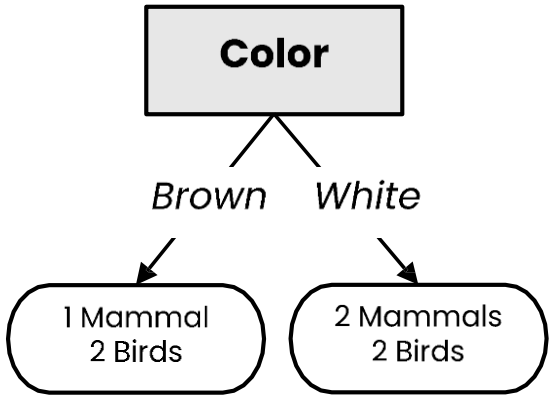
$$gain(X, a) = entropy(X) - \sum_{v \in Values(a)} \frac{|X_v|}{|X|} entropy(X_v)$$

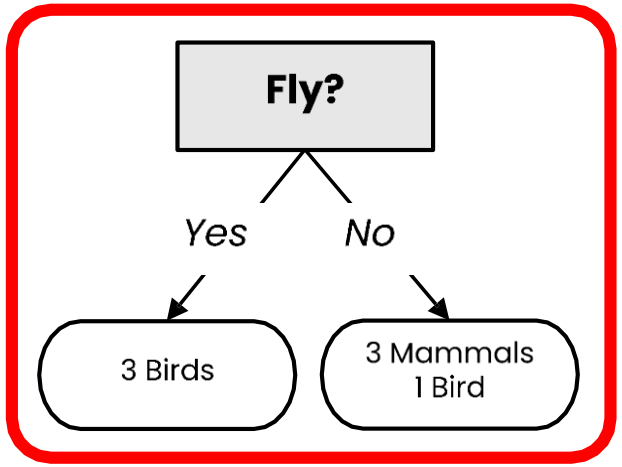
where X_v is the subset of X for which $a = v$

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**





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

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

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

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

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

Gini Impurity

Gini Impurity

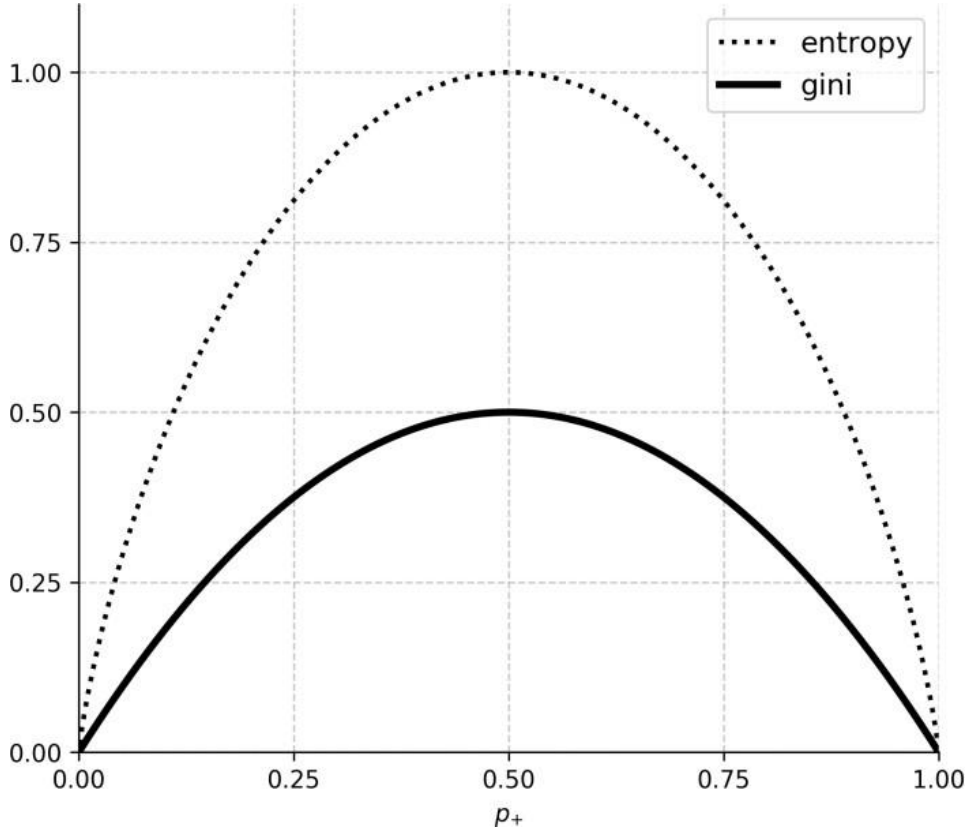
- Gini impurity measures how often a randomly chosen example would be incorrectly labeled if it was randomly labeled according to the label distribution

Error of classifying randomly picked fruit with randomly picked label

- 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

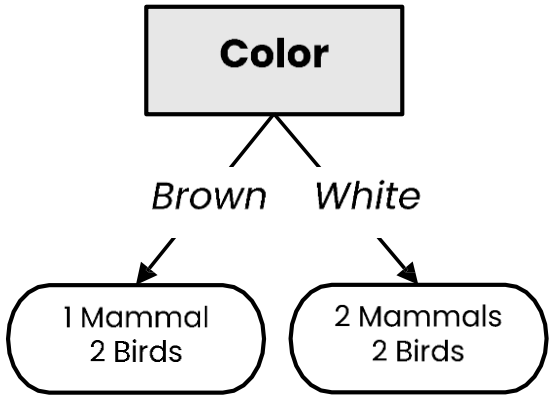


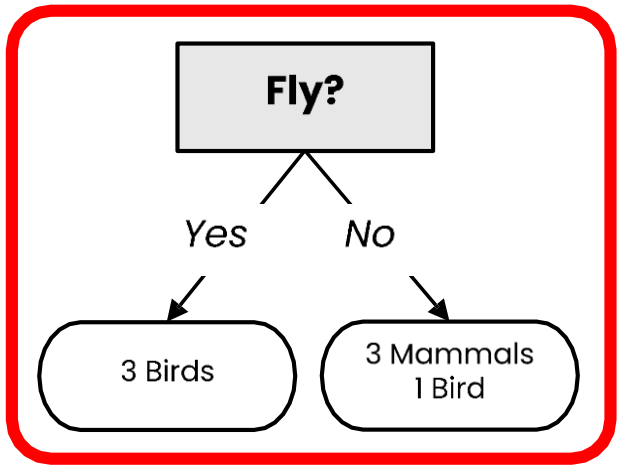
- Can be used as an alternative to entropy for selecting attributes!

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**





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

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

$$gini\ (X_{color=white}) = 0.5$$

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

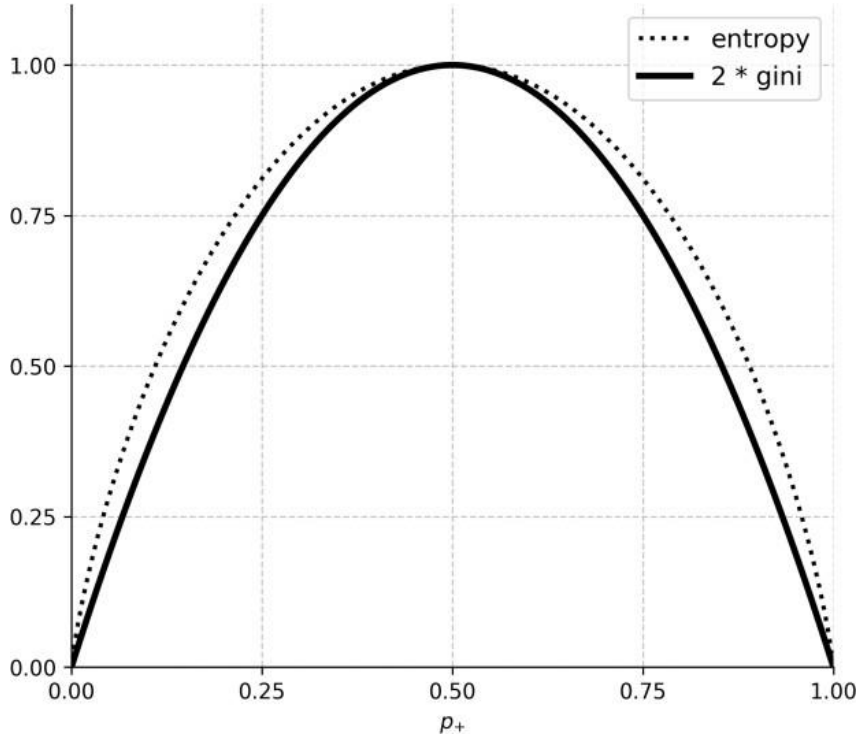
$$gini\ (X_{fly=yes}) = 0$$

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

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

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"

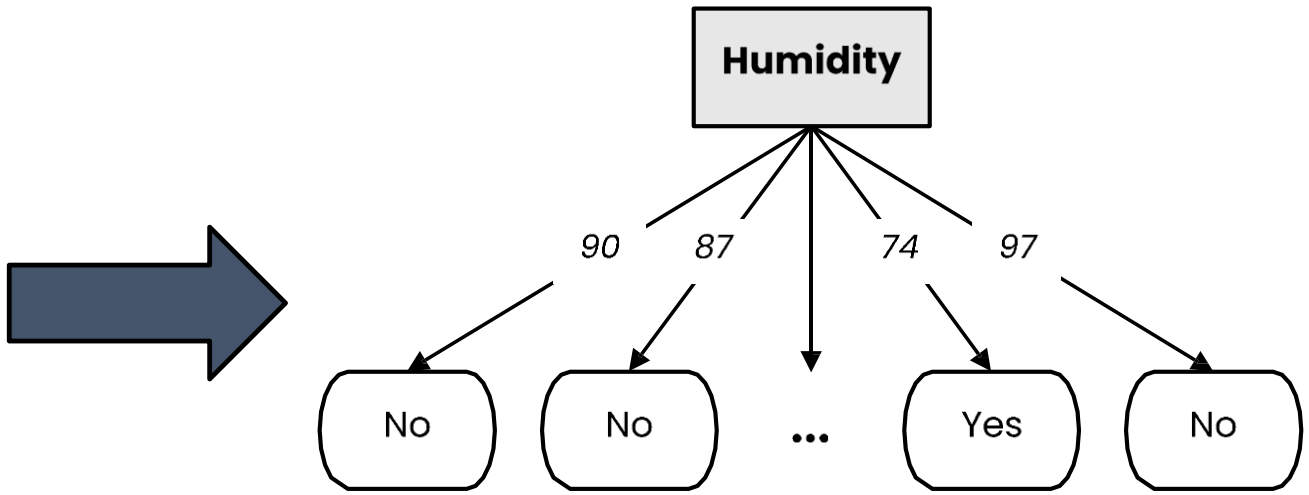
Handling Numerical Attributes

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

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

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

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

Handling numerical attributes

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

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

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

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

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

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

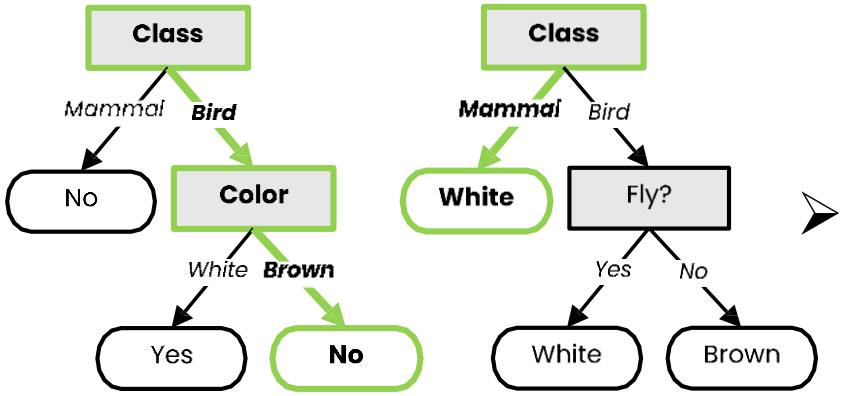
$$= 0.33$$

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

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

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)

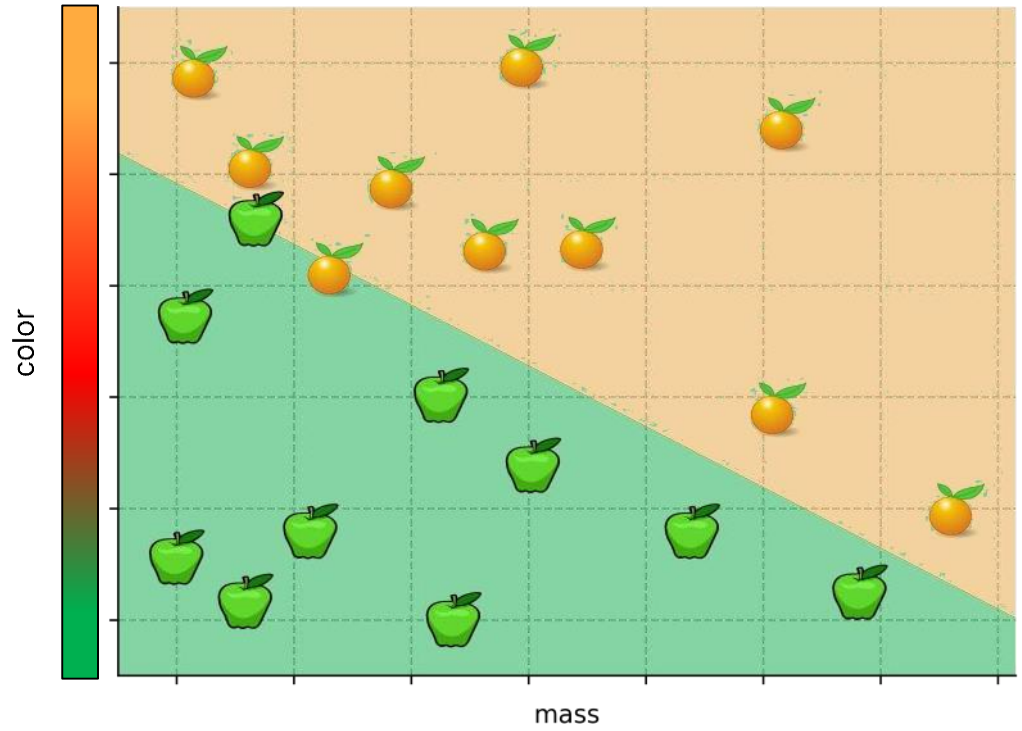
➢ Build a decision tree on all other attributes (including label) to predict missing values

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

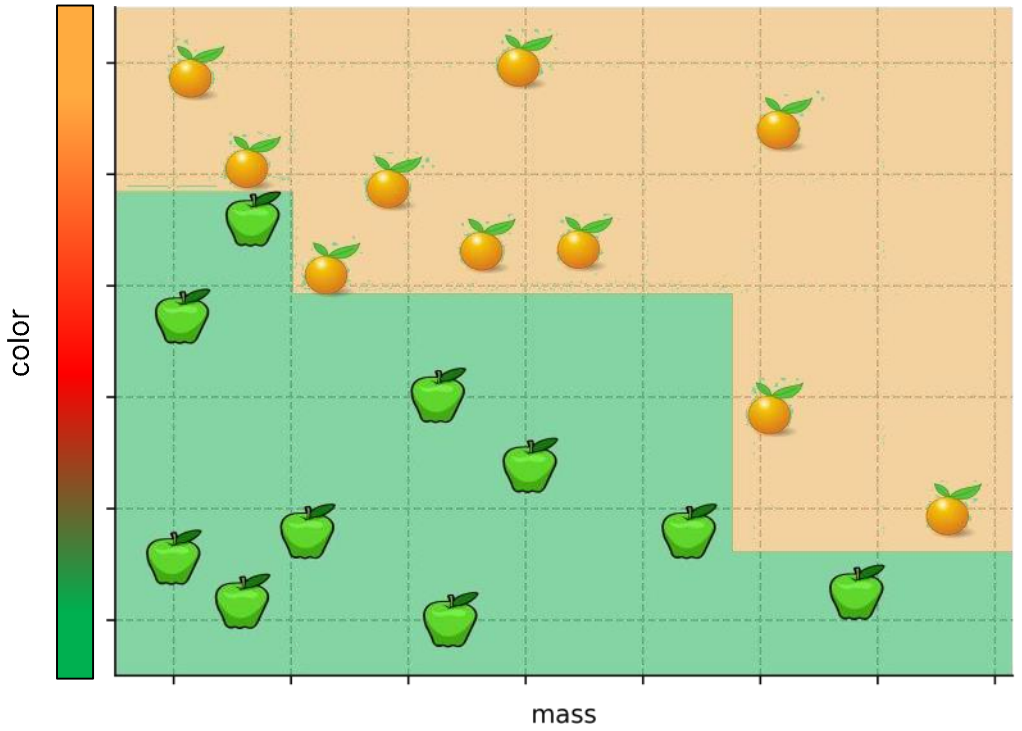
Decision Boundaries

- Decision trees produce non-linear decision boundaries

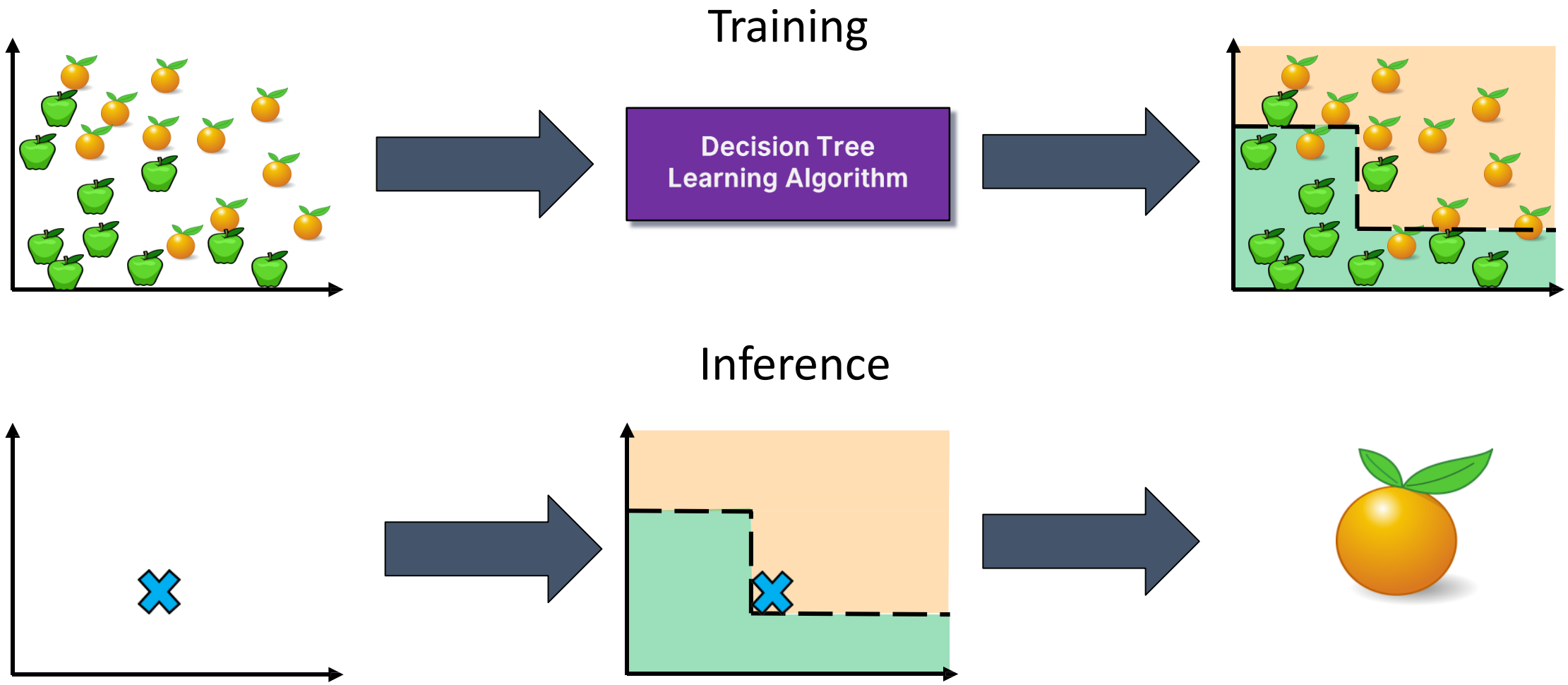
Support Vector Machines



Decision Tree



Decision Trees: Training and Inference



Random Forests

(Ensemble learning with decision trees)

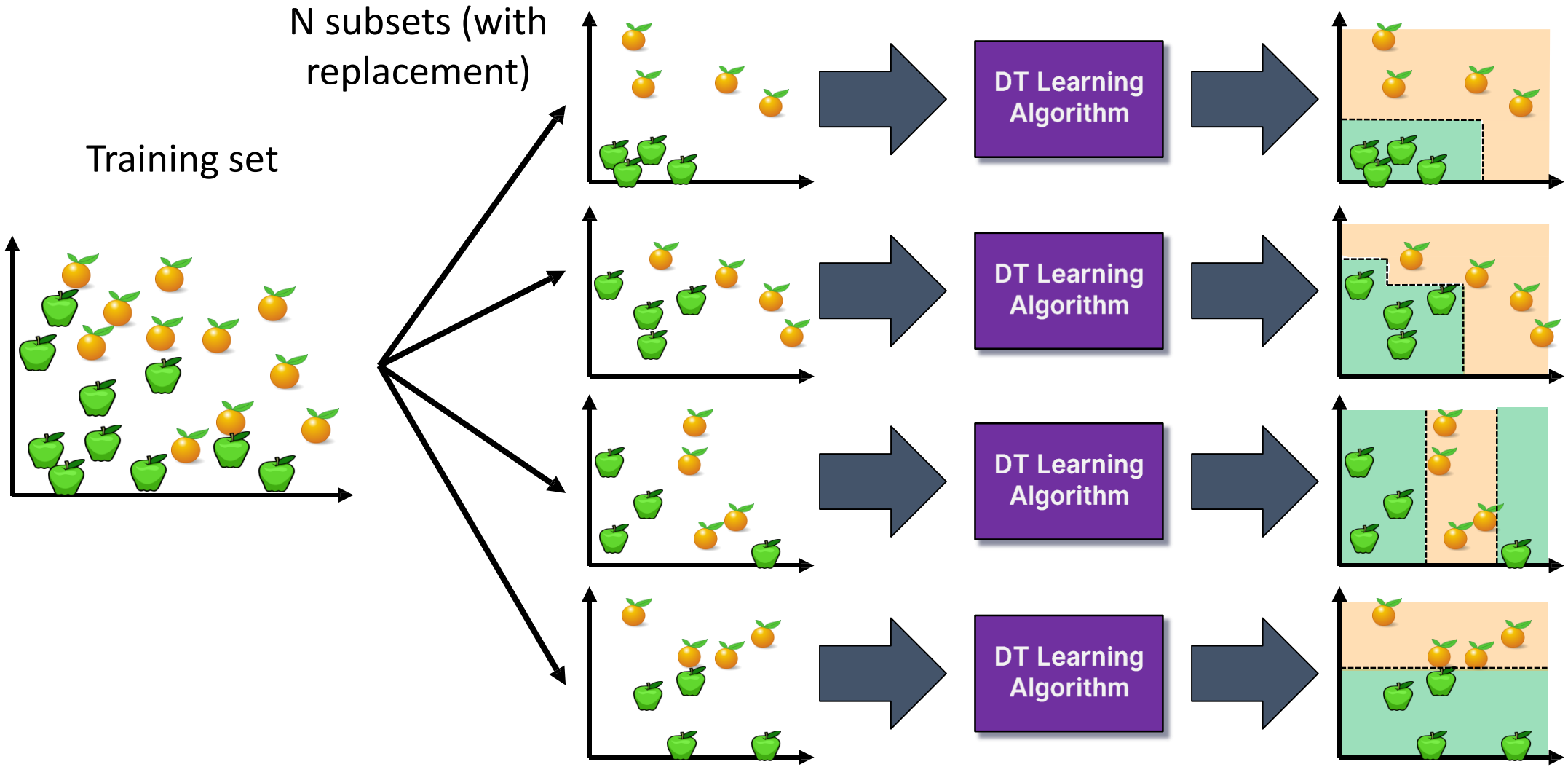
Random Forests

- 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?

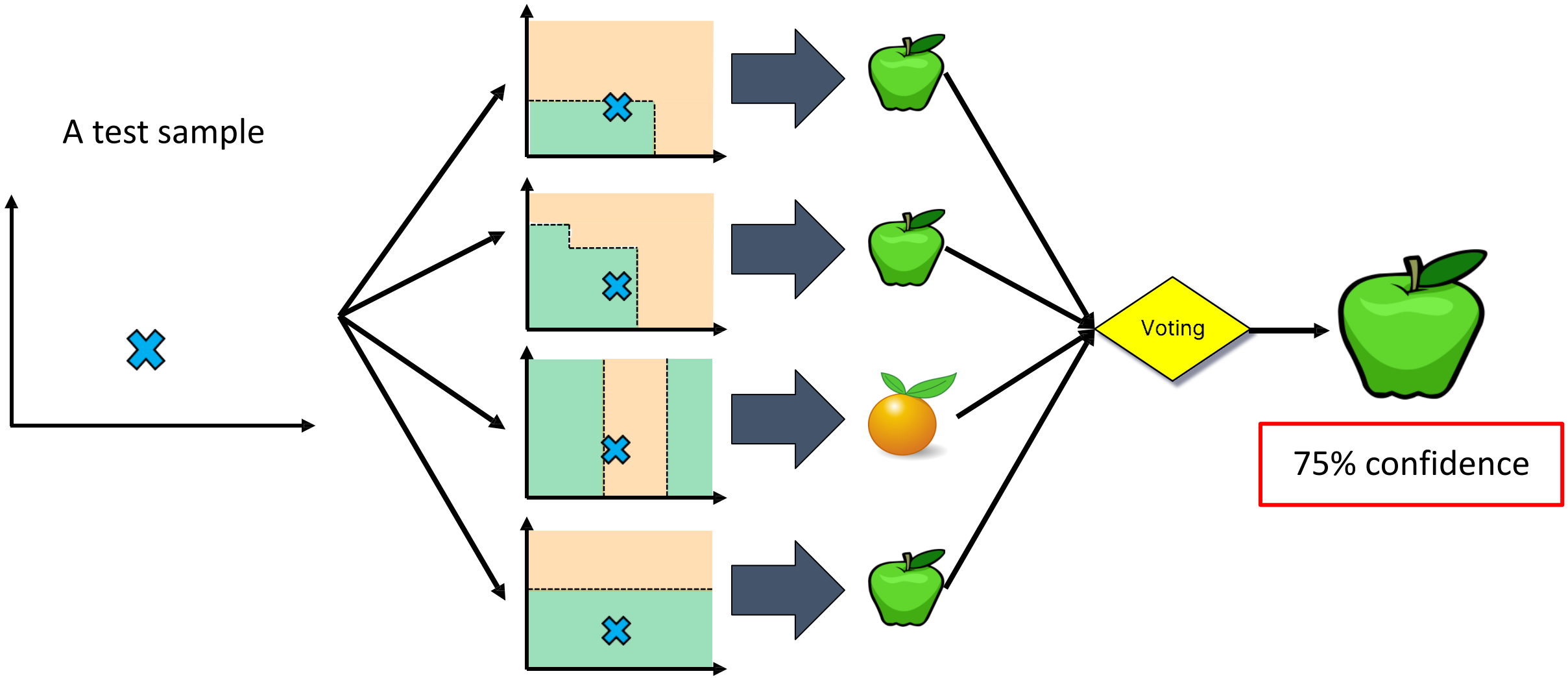
1. Bagging (**B**ootstrap **Agg**regat**ing**):
   - Take random subsets of data points from the training set to create N smaller data sets
   - Fit a decision tree on each subset

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



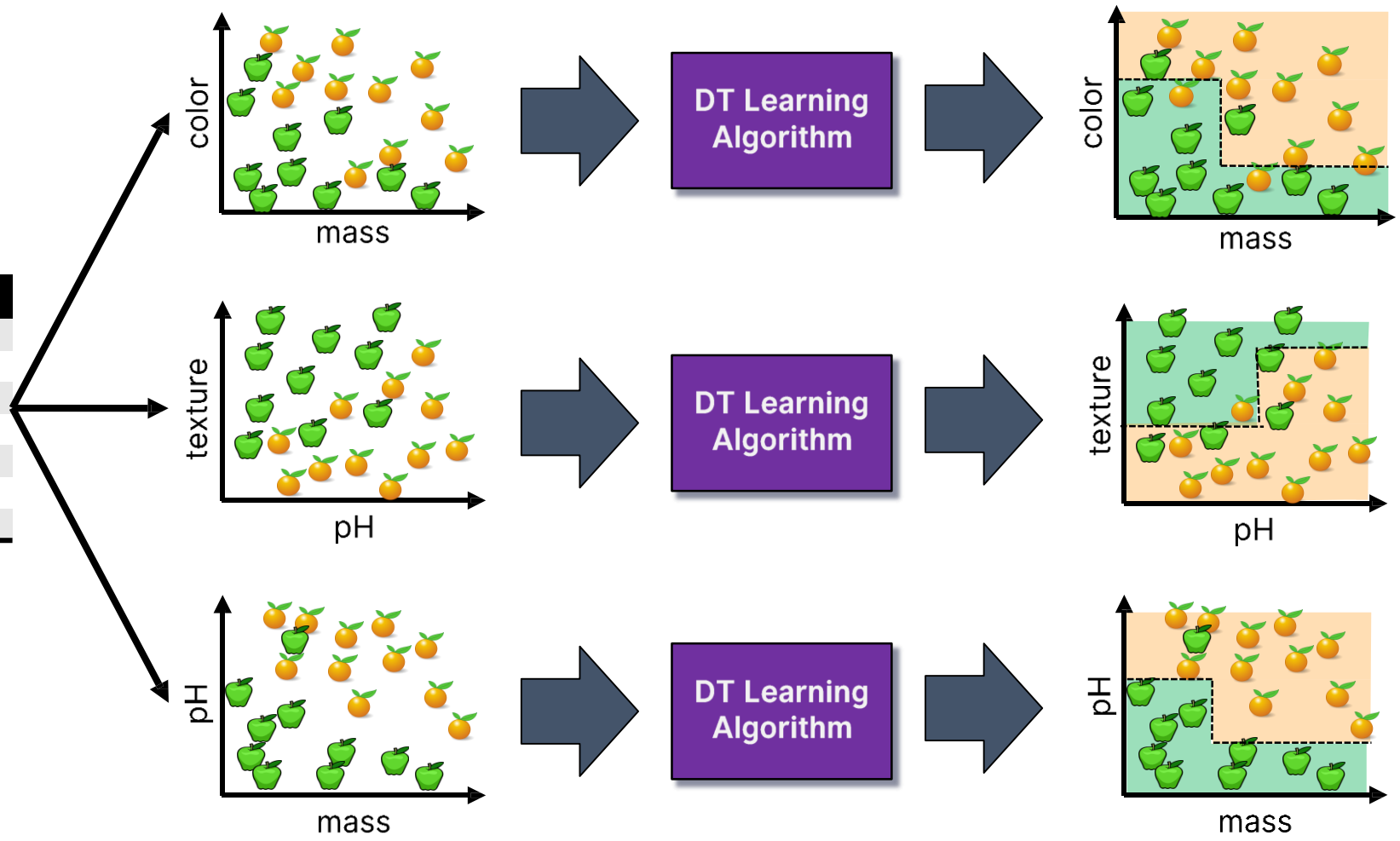
Bagging at inference time



Random Subspace Method at training time

Training data

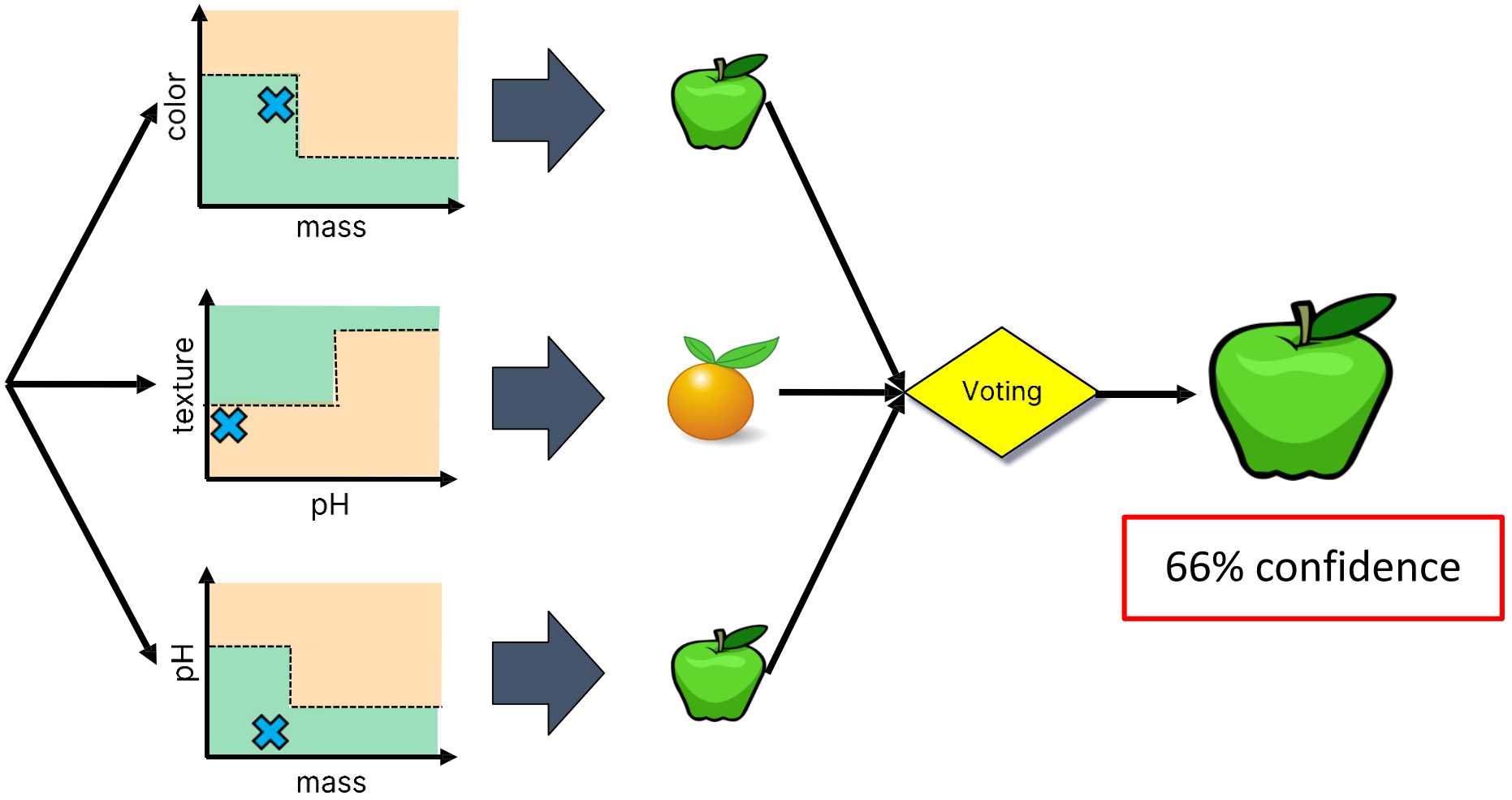
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



Random Subspace Method at inference time

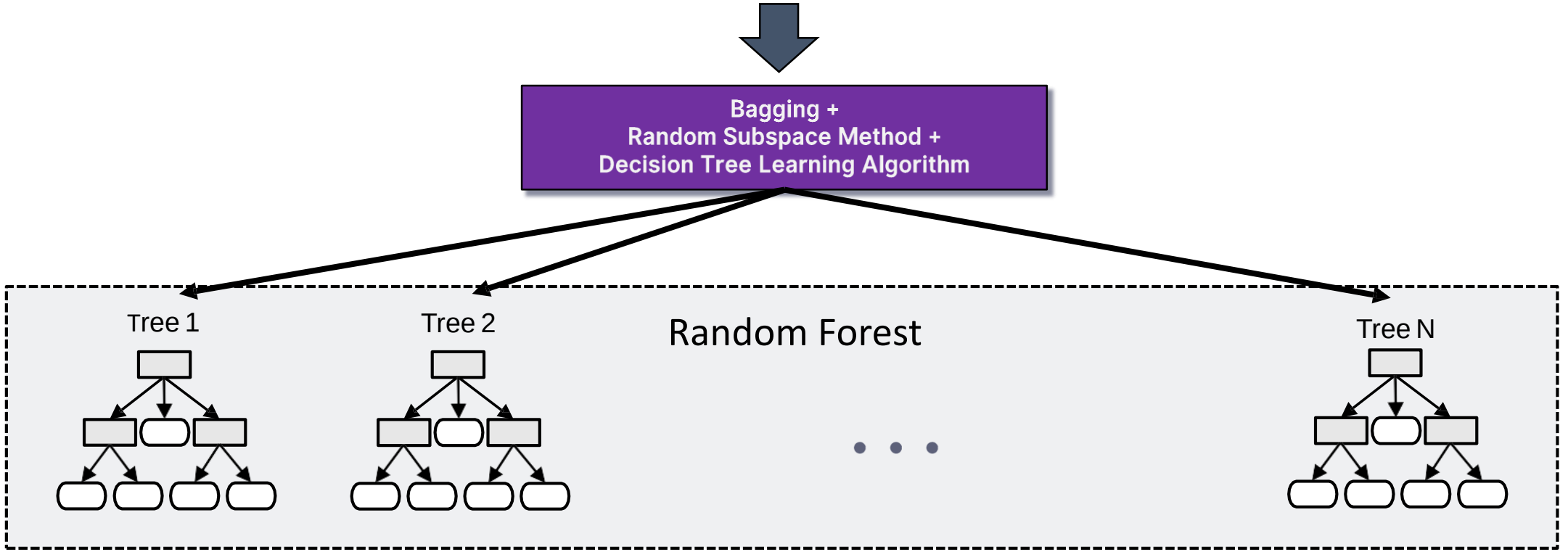
A test sample

87	Red	Smooth	3.1



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**

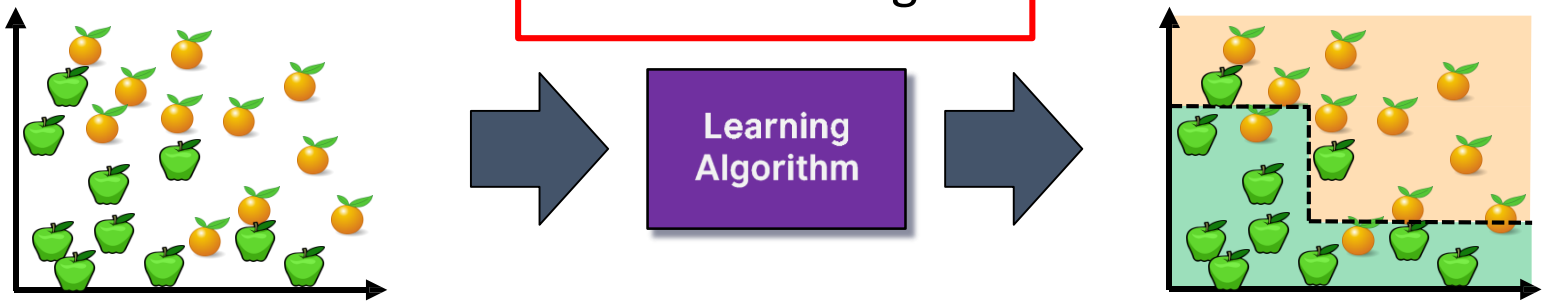


Ensemble Learning

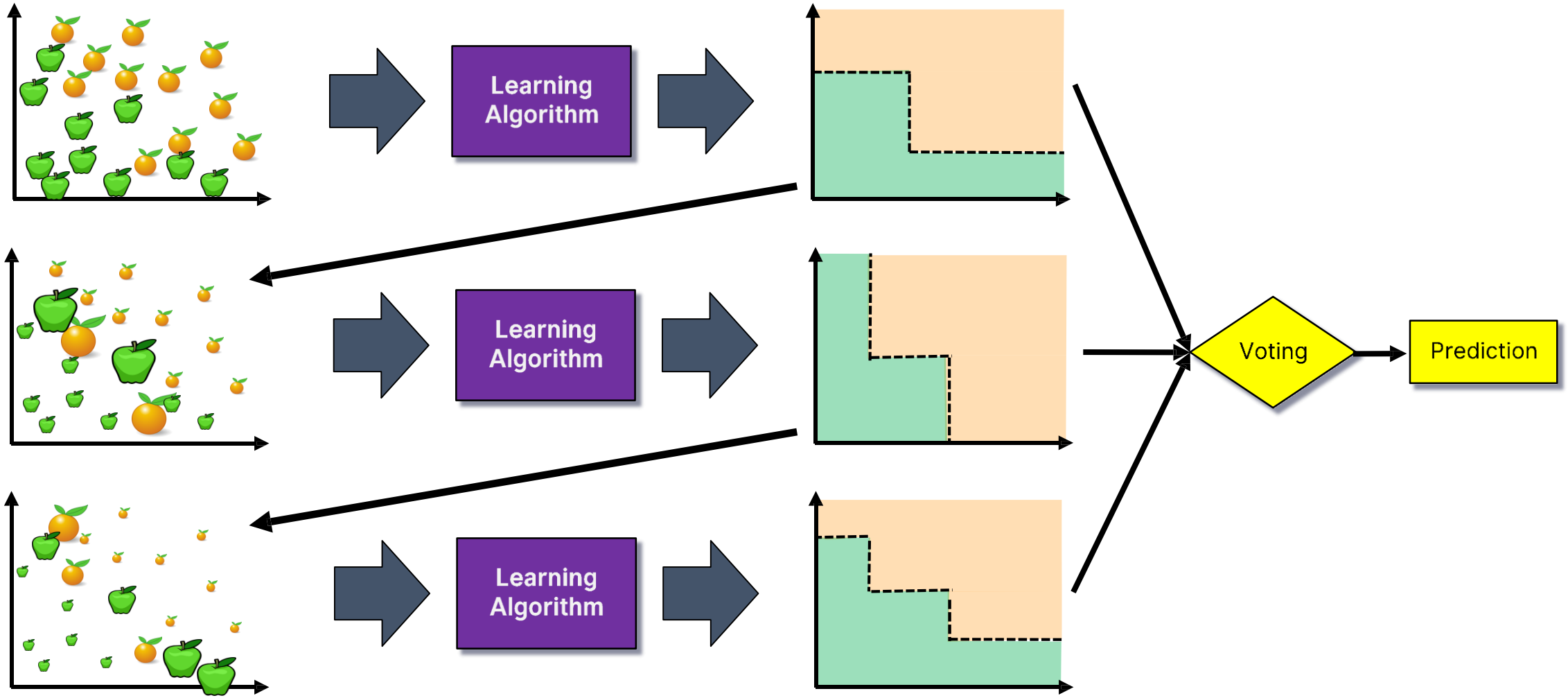
- Ensemble Learning:
  - Method that combines multiple learning algorithms to obtain performance improvements over its components
- **Random Forests** are one of the most common examples of ensemble learning
- Other commonly-used ensemble methods:
  - Bagging: multiple models on random subsets of data 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

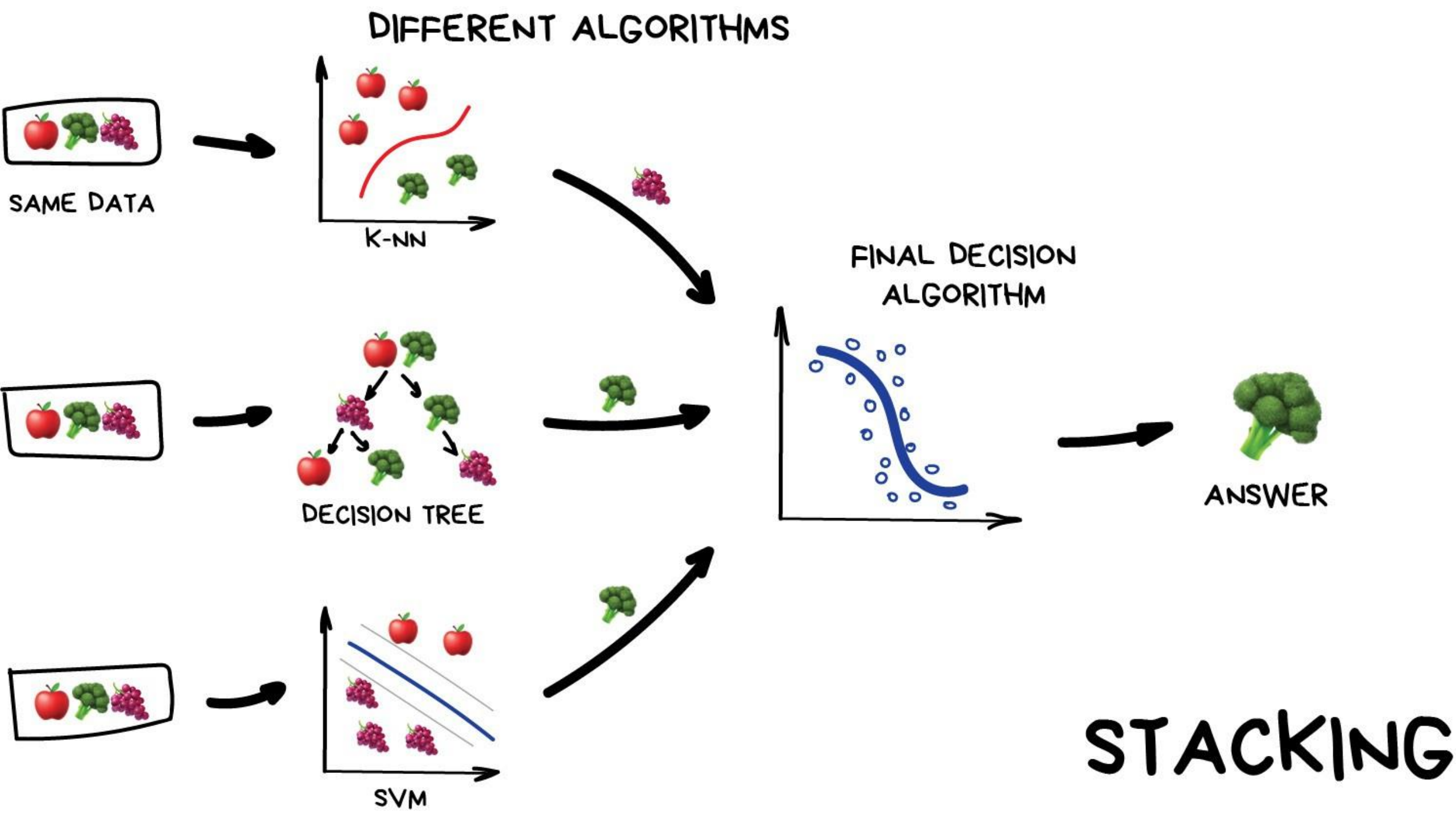
Boosting

All samples have the same weight

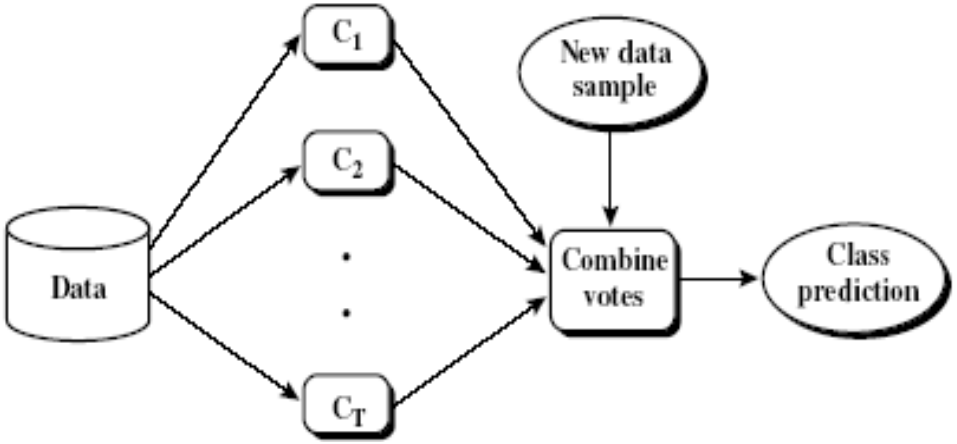


Boosting


Learning Algorithm
Learning Algorithm
Learning Algorithm
Voting
Prediction


DIFFERENT ALGORITHMS
SAME DATA
K-NN
FINAL DECISION ALGORITHM
DECISION TREE
ANSWER
SVM
STACKING

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!