Decision Trees.
Random Forests.

Slides are adopted from Radu Ionescu

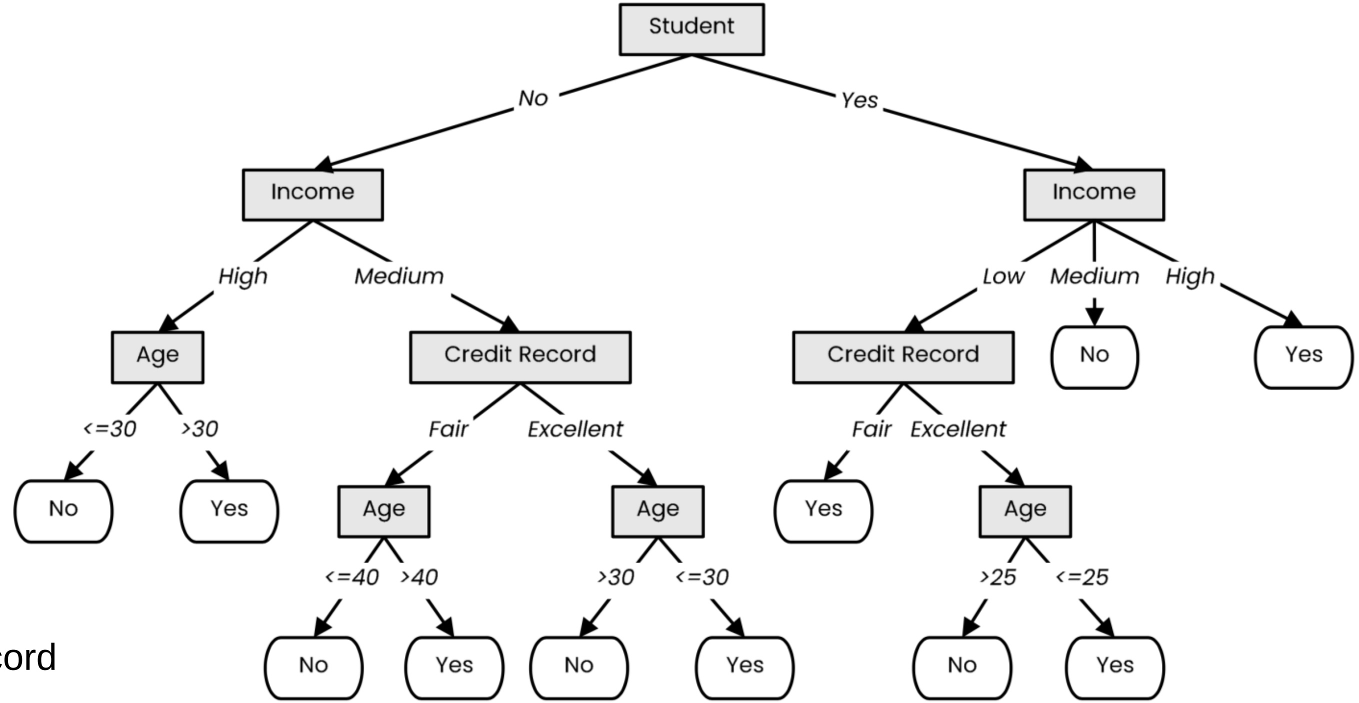
Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

Who to loan?

- Not a student
- 45 years old
- Medium income
- Fair credit record

- Student
- 27 years old
- Low income
- Excellent credit record



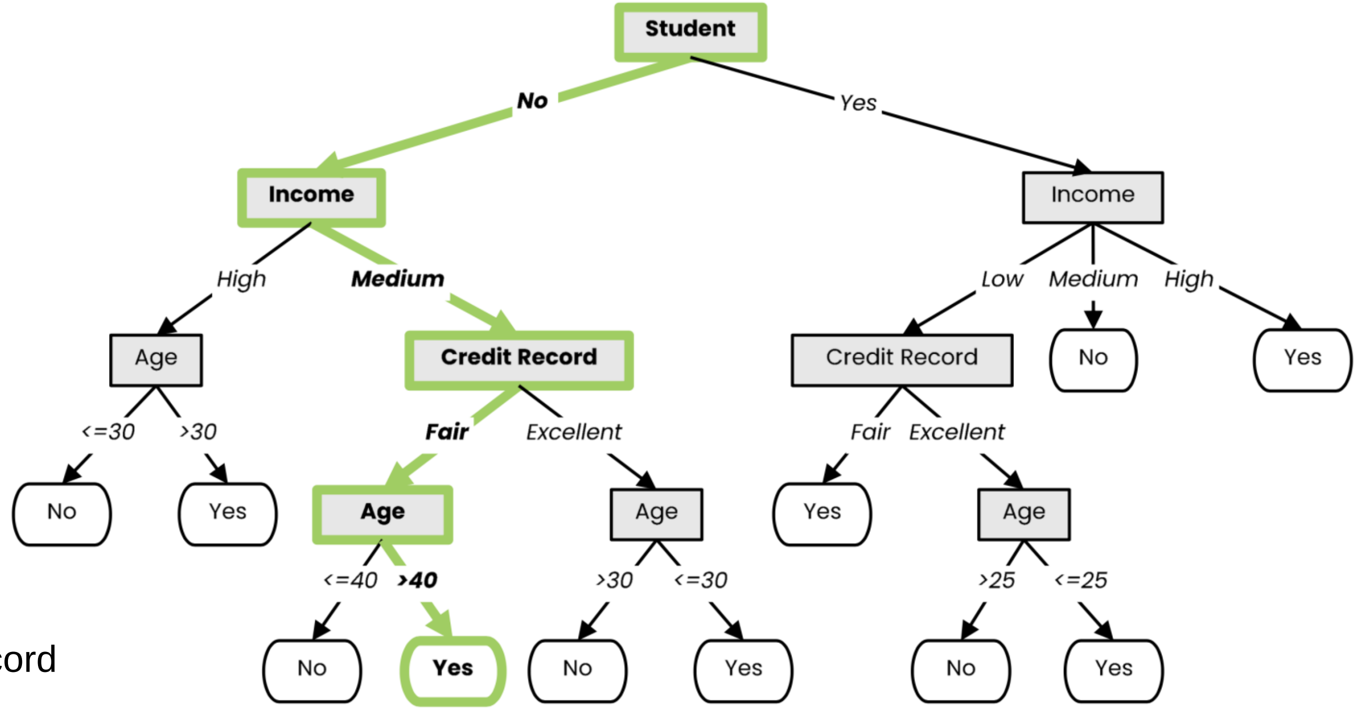
Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

Who to loan?

- Not a student
- 45 years old
- Medium income
- Fair credit record
- ➢ Yes

- Student
- 27 years old
- Low income
- Excellent credit record



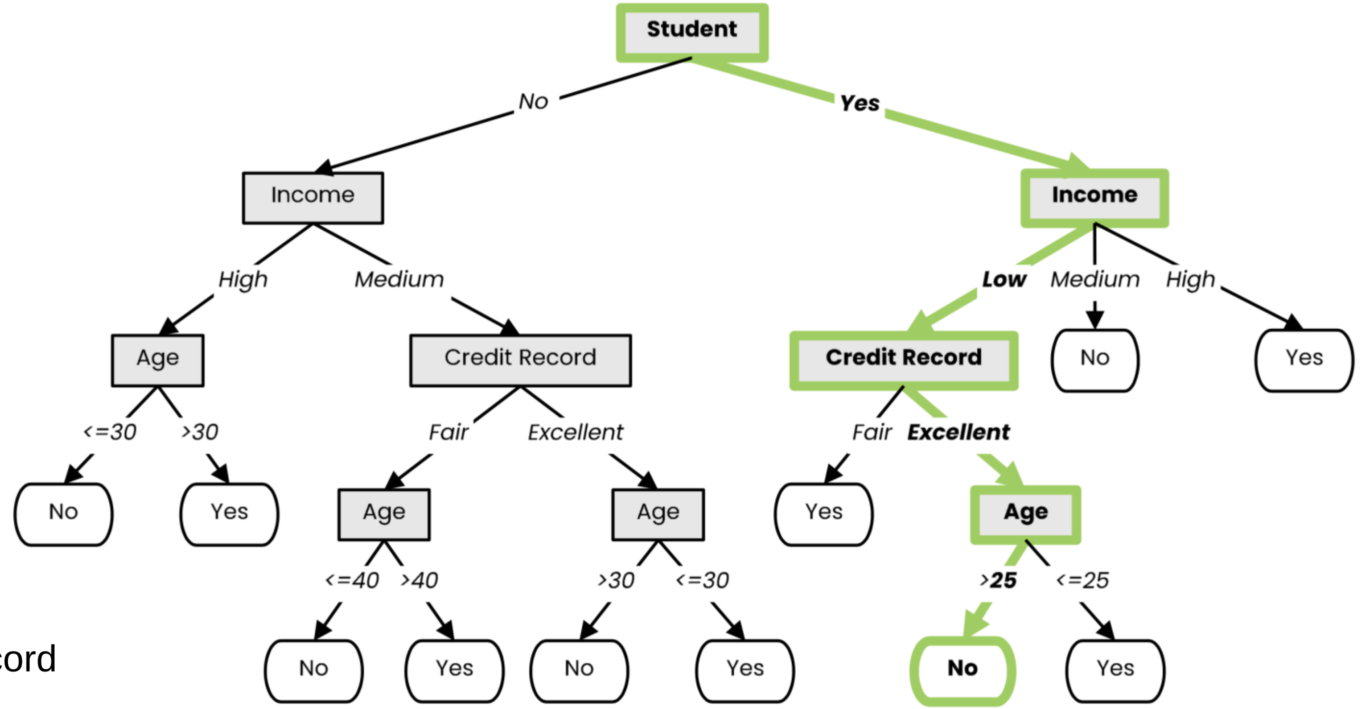
Definition

- A tree-like model that illustrates series of events leading to certain decisions
- Each node represents a test on an attribute and each branch is an outcome of that test

Who to loan?

- Not a student
- 45 years old
- Medium income
- Fair credit record
- ➢ Yes

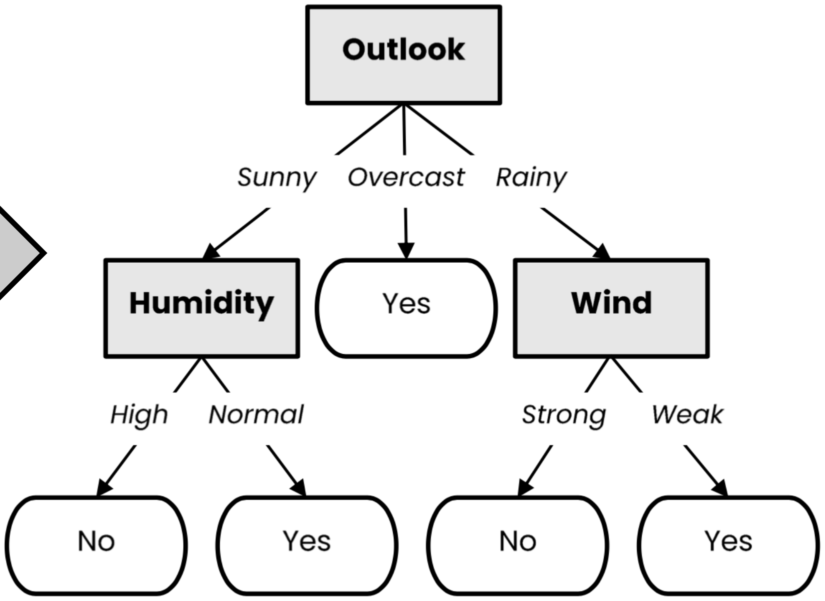
- Student
- 27 years old
- Low income
- Excellent credit record
- ➢ No



Decision Tree Learning

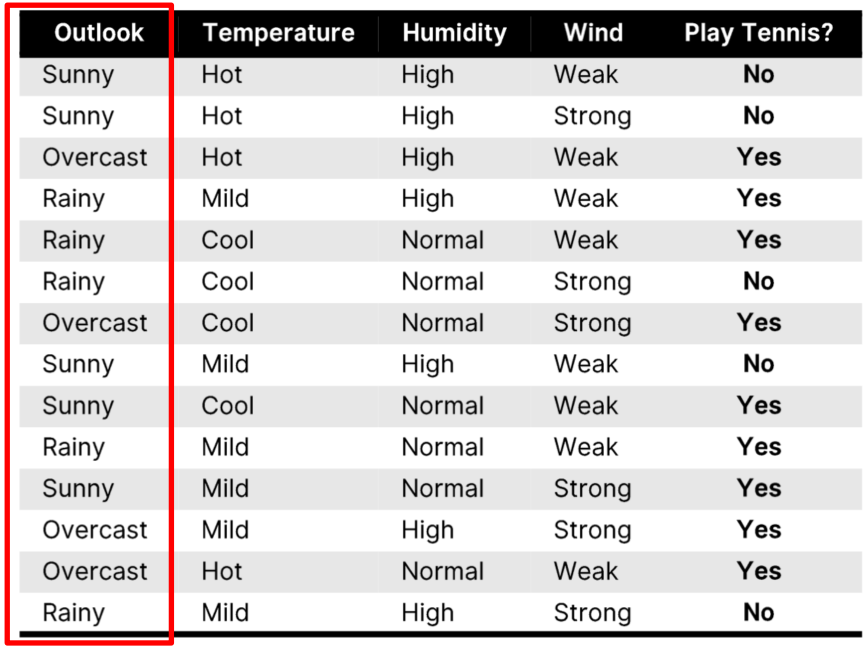
- We use labeled data to obtain a suitable decision tree for future predictions
  - ➢ We want a decision tree that works well on unseen data, while asking as few questions as possible

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**

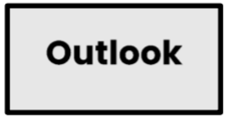


Decision Tree Learning

- 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	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*

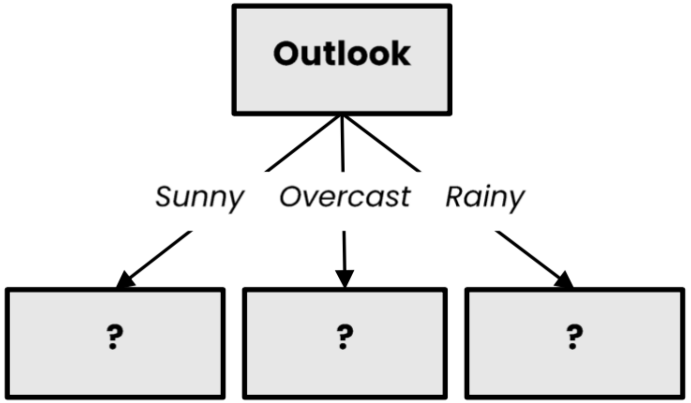
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**No**
Hot	High	Strong	**No**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**
Mild	Normal	Strong	**Yes**

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*

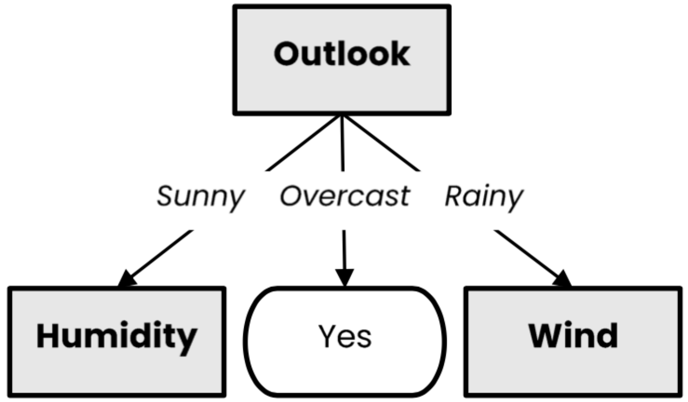
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**No**
Hot	High	Strong	**No**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**
Mild	Normal	Strong	**Yes**

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**

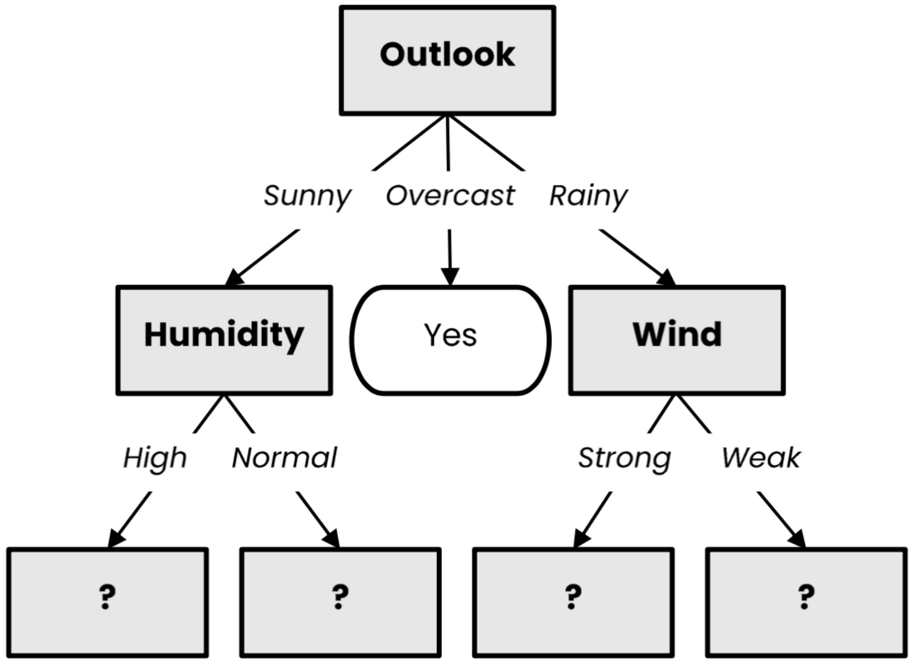
Outlook = *Rainy*

Wind = Strong

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

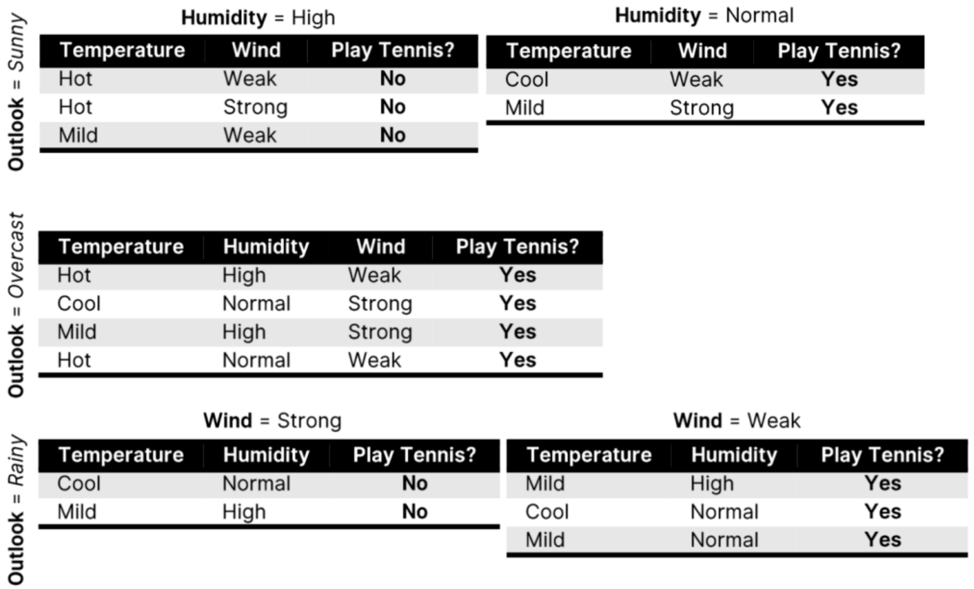
Wind = Weak

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



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**

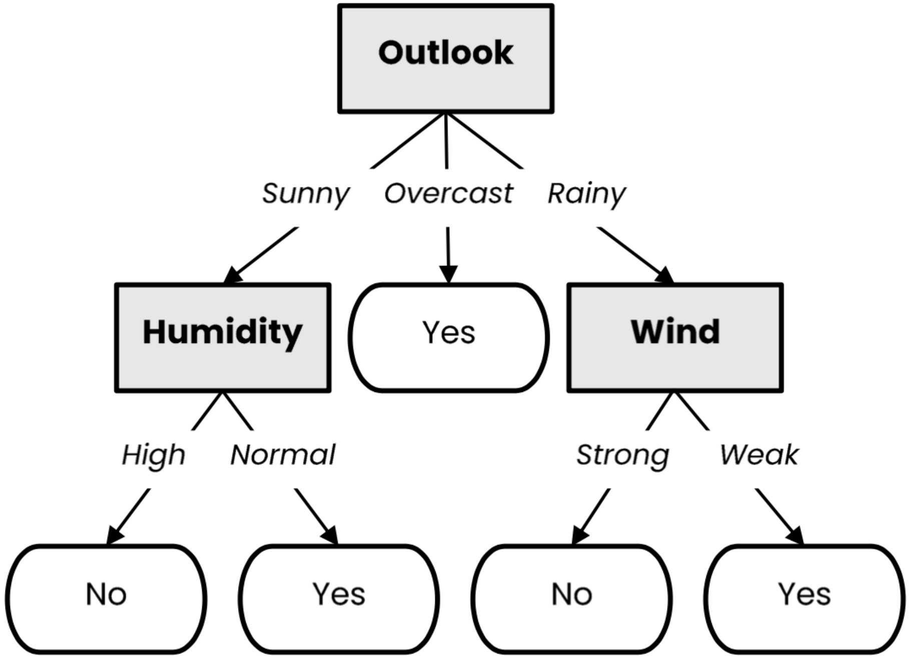
Outlook = *Rainy*

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**

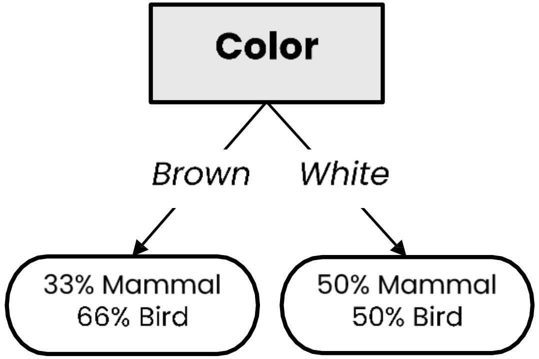


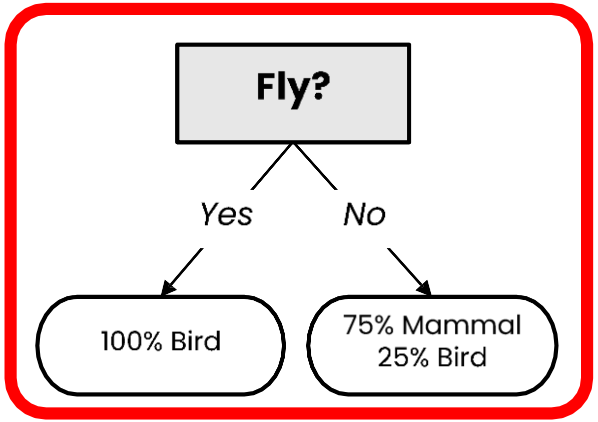
Decision Tree Learning (Python)

```
def make_tree(X):
    node = TreeNode(X)
    if should_be_leaf_node(X):
        node.label = majority_label(X)
    else:
        a = select_best_splitting_attribute(X)
        for v in values(a):
            $X_v = \{x \in X | x[a] == v\}$
            node.children.append(make_tree($X_v$))
    return node
```

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

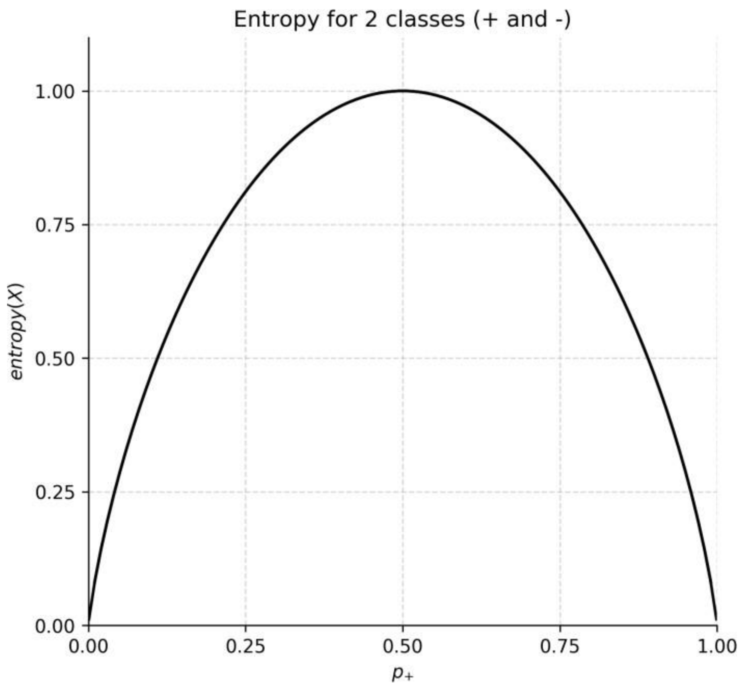
Low entropy
High entropy

- 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!



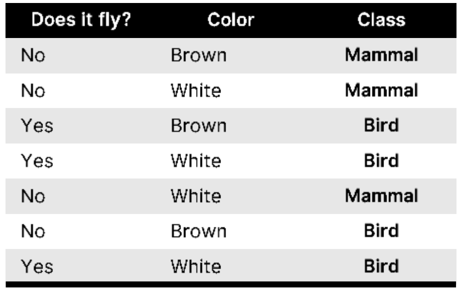
Information Gain

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

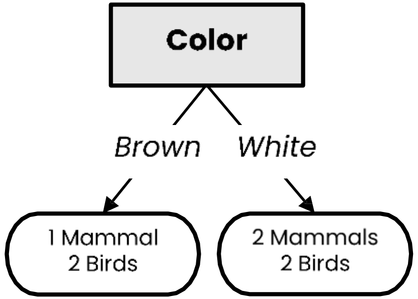
$$gain(X, a) = entropy(X) - \sum_{v \in Values(a)} \frac{|X_v|}{|X|} entropy(X_v)$$

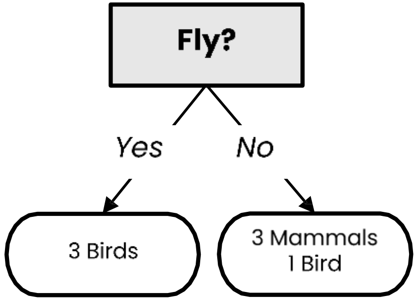
where X_v is the subset of X for which $a = v$

Best attribute = highest information gain



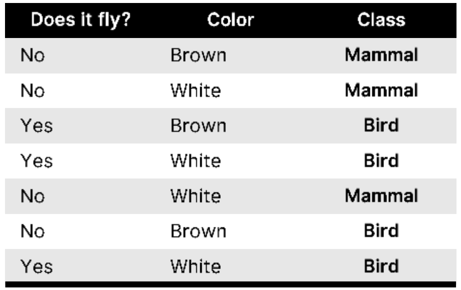
Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**



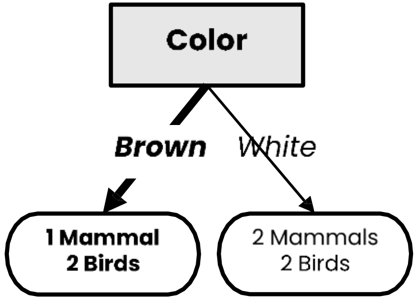


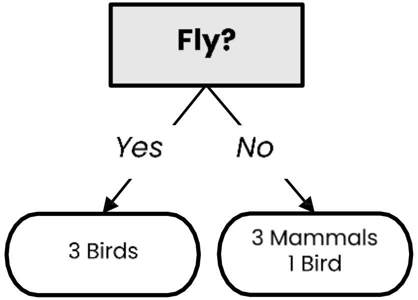
$$entropy\ (X) = -p_{\text{mammal}} \log_2 p_{\text{mammal}} - p_{\text{bird}} \log_2 p_{\text{bird}} = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \approx 0.985$$

Best attribute = highest information gain



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**



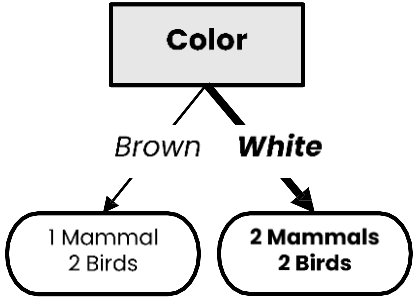


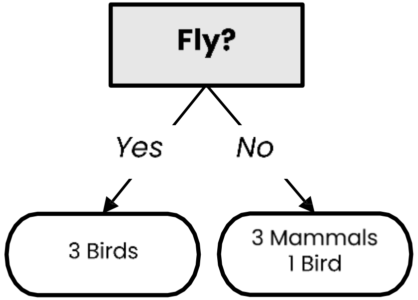
$$entropy\ (X) = -p_{\text{mammal}} \log_2 p_{\text{mammal}} - p_{\text{bird}} \log_2 p_{\text{bird}} = -\frac{3}{7}\ \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \approx\ 0.985$$

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

Best attribute = highest information gain

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**

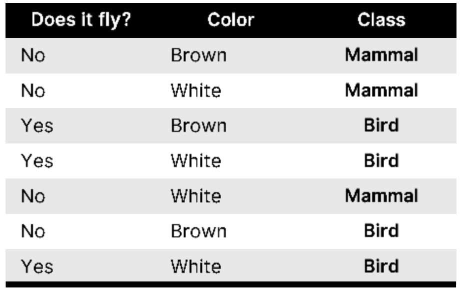




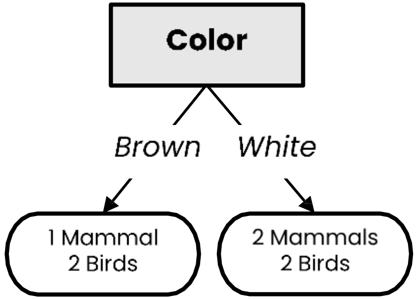
$$entropy\ (X) = -p_{\mathrm{mammal}} \log_2 p_{\mathrm{mammal}} - p_{\mathrm{bird}} \log_2 p_{\mathrm{bird}} = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} \approx 0.985$$

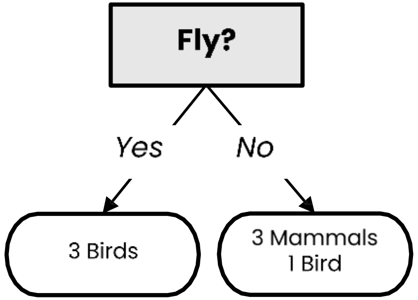
$$entropy\ (X_{color=brown}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} \approx 0.918 \qquad entropy\ (X_{color=white}) = 1$$

Best attribute = highest information gain



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**



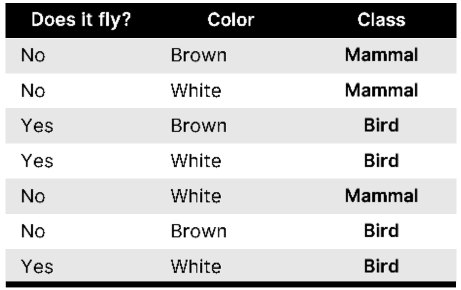


$$entropy\,(X) = -p_{\mathrm{mammal}} \log_2 p_{\mathrm{mammal}} - p_{\mathrm{bird}} \log_2 p_{\mathrm{bird}} = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} \approx 0.985$$

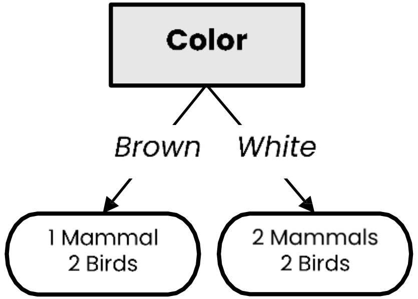
$$entropy\,(X_{color=brown}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} \approx 0.918 \qquad entropy\,(X_{color=white}) = 1$$

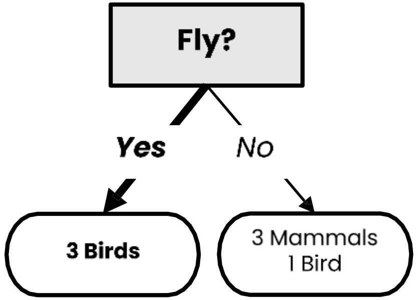
$$\boldsymbol{gain\,(X, color) = 0.985 - \frac{3}{7}\cdot 0.918 - \frac{4}{7}\cdot 1 \approx 0.020}$$

Best attribute = highest information gain



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**





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

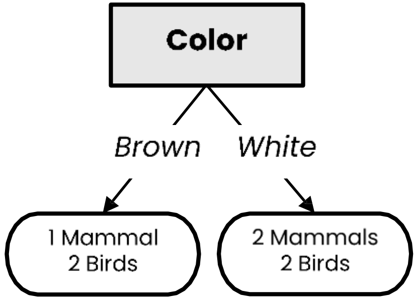
$$entropy\ (X_{color=brown})\ = -\frac{1}{3}\ \log_2 \frac{1}{3} - \frac{2}{3}\ \log_2 \frac{2}{3} \approx\ 0.918 \qquad entropy\ (X_{color=white})\ = 1$$

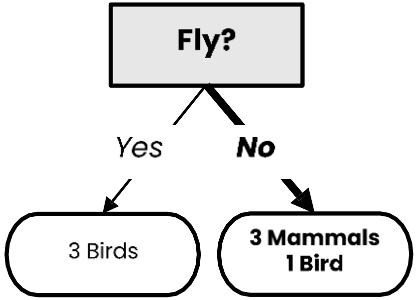
$$\boldsymbol{gain\ (X, color) = 0.985 - \frac{3}{7} \cdot 0.918 - \frac{4}{7} \cdot 1 \approx 0.020}$$

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

Best attribute = highest information gain

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**





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

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

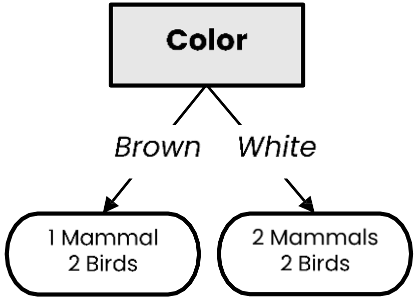
$$\boldsymbol{gain\ (X, color) = 0.985 - \frac{3}{7} \cdot 0.918 - \frac{4}{7} \cdot 1 \approx 0.020}$$

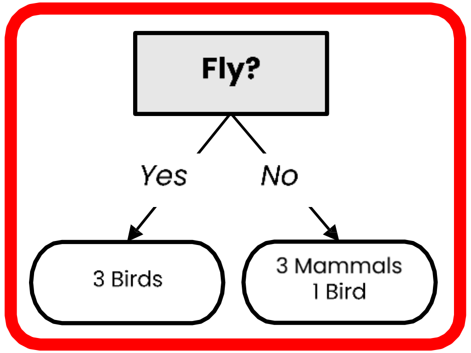
$$entropy\ (X_{fly=yes})\ = 0 \qquad entropy\ (X_{fly=no})\ = -\frac{3}{4}\ \log_2 \frac{3}{4} - \frac{1}{4}\ \log_2 \frac{1}{4} \approx\ 0.811$$

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**





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

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

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

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

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

ID3 Algorithm (Python)

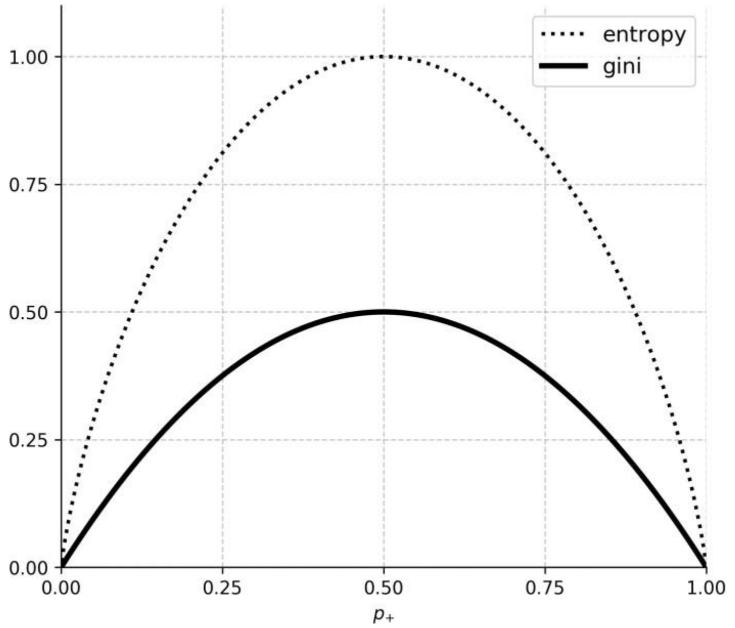
```
# ID = Iterative Dichotomiser
def ID3(X):
    node = TreeNode(X)
    if all_points_have_same_class(X):
        node.label = majority_label(X)
    else:
        a = select_attribute_with_highest_information_gain(X)
        if gain(X, a) == 0:
            node.label = majority_label(X)
        else:
            for v in values(a):
                $X_v = \{x \in X \mid x[a] == v\}$
                node.children.append(ID3($X_v$))
    return node
```

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:

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

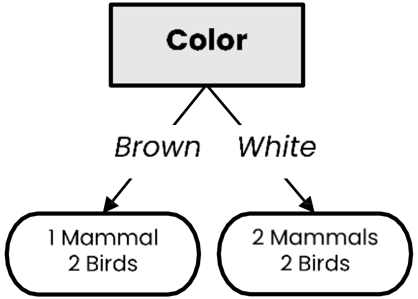
where p_i is the proportion of elements of class i

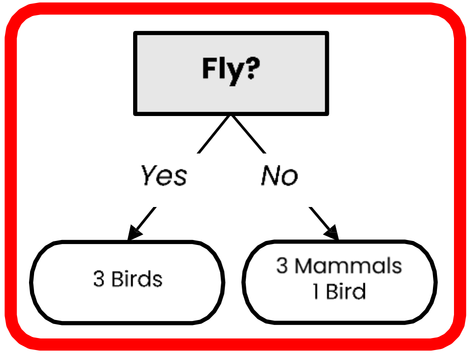
- Can be used as an alternative to entropy for selecting attributes!

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**





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

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

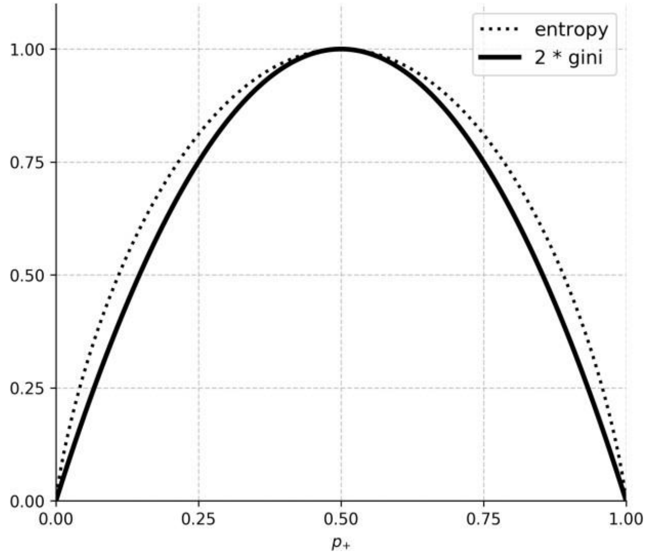
$$\boldsymbol{\triangle\, gini\ (X, color) = 0.489 - \frac{3}{7} \cdot 0.444 - \frac{4}{7} \cdot 0.5 \approx 0.013}$$

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

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

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



Pruning

Pruning

- Pruning is a technique that reduces the size of a decision tree by removing branches of the tree which provide little predictive power
- It is a **regularization** method that reduces the complexity of the final model, thus reducing overfitting
  - ➢ Decision trees are prone to overfitting!

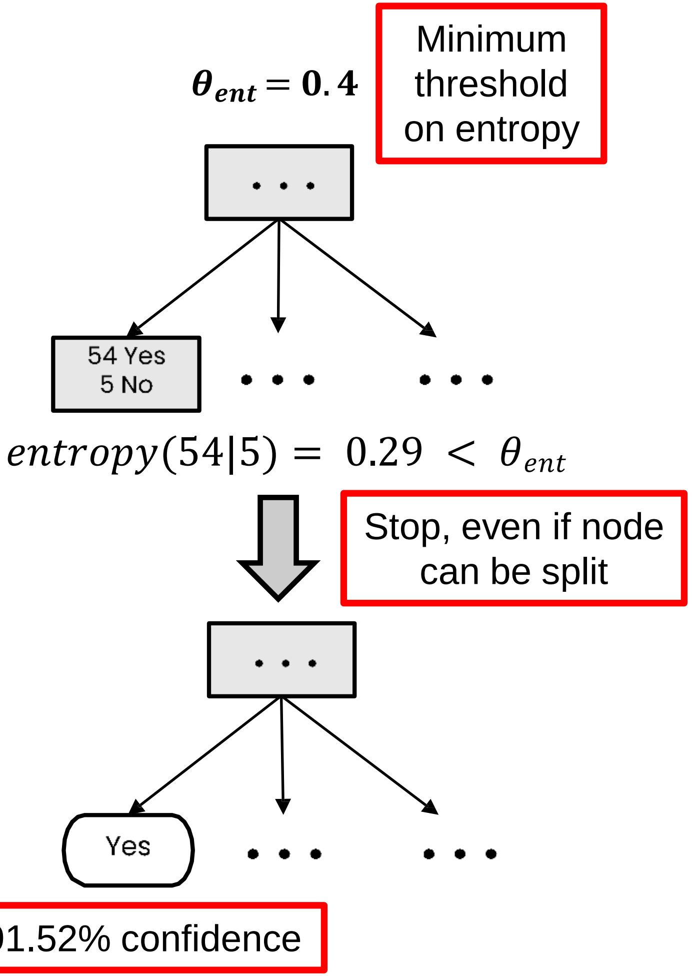
- Pruning methods:
  - ➢ Pre-pruning: Stop the tree building algorithm before it fully classifies the data
  - ➢ Post-pruning: Build the complete tree, then replace some non-leaf nodes with leaf nodes if this improves validation error

Pre-pruning

- Pre-pruning implies early stopping:
  - ➢ If some condition is met, the current node will not be split, even if it is not 100% pure
  - ➢ It will become a leaf node with the label of the majority class in the current set

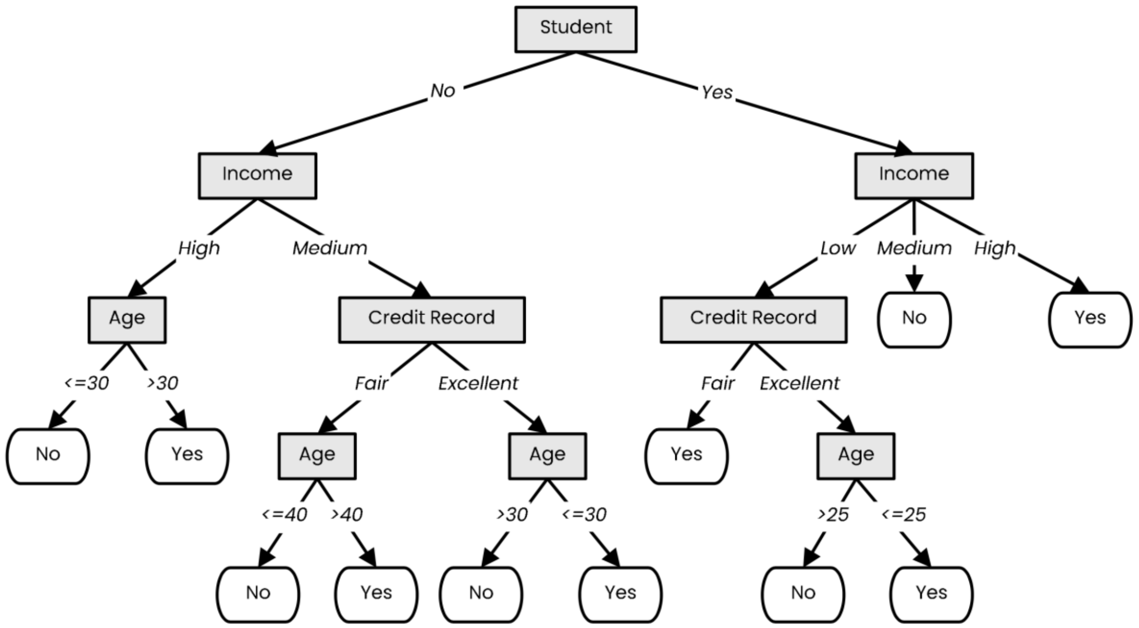
(the class distribution could be used as prediction confidence)

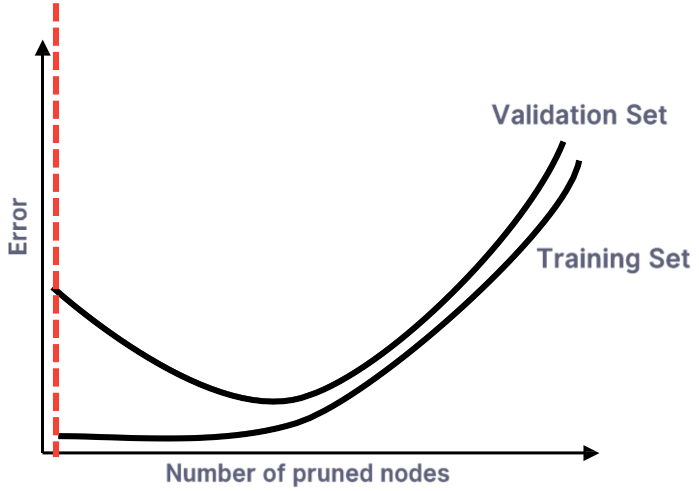
- Common stopping criteria include setting a threshold on:
  - ➢ Entropy (or Gini Impurity) of the current set
  - ➢ Number of samples in the current set
  - ➢ Gain of the best-splitting attribute
  - ➢ Depth of the tree



$\boldsymbol{\theta_{ent} = 0.4}$
Minimum threshold on entropy
. . .
54 Yes
5 No
. . .
. . .
$entropy(54|5) = 0.29 < \theta_{ent}$
Stop, even if node can be split
. . .
Yes
. . .
. . .
91.52% confidence

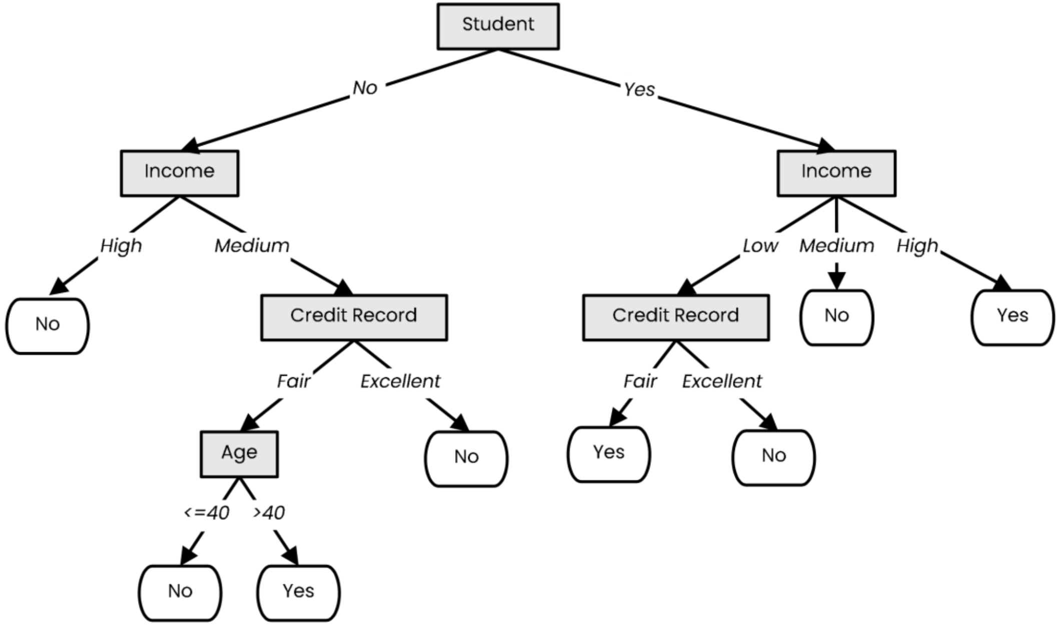
Post-pruning

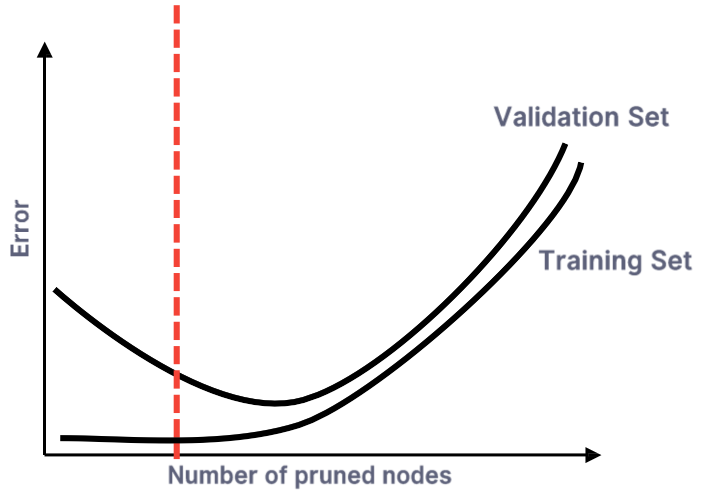




- Prune nodes in a bottom-up manner, if it decreases validation error

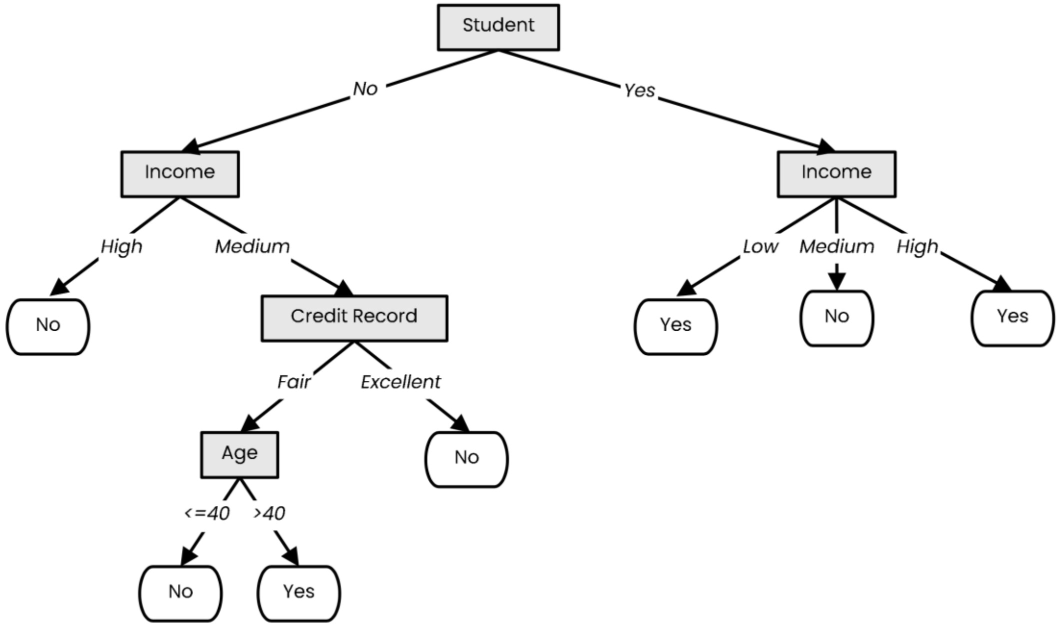
Post-pruning

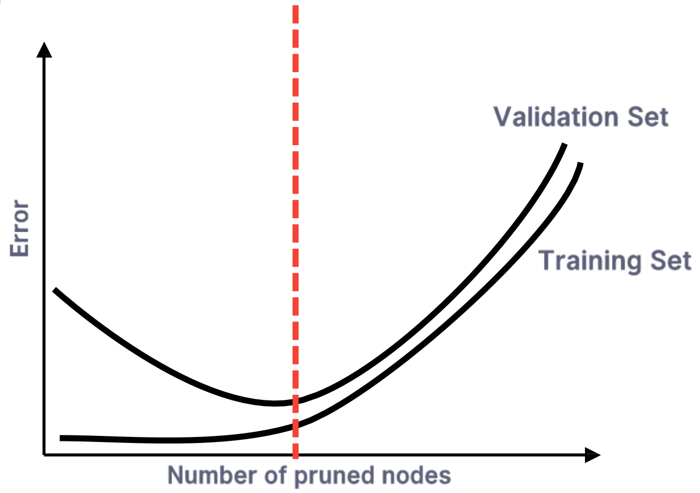




- Prune nodes in a bottom-up manner, if it decreases validation error

Post-pruning





- Prune nodes in a bottom-up manner, if it decreases validation error

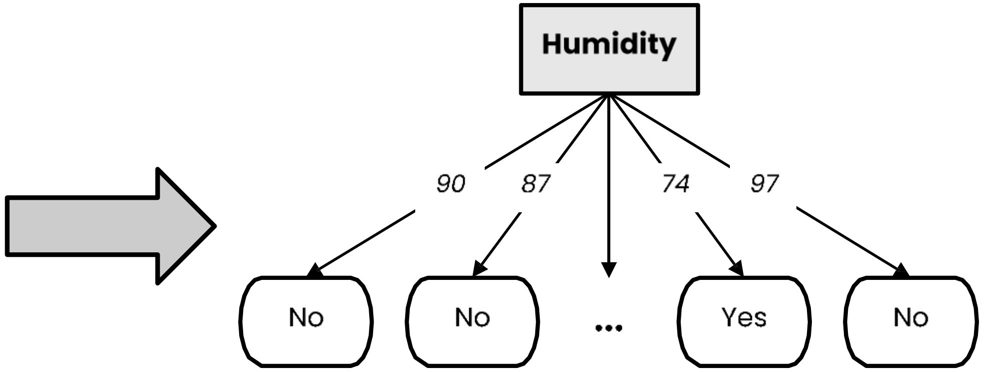
Handling Numerical Attributes

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 of numerical attribute a if we split at value t

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

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

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

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

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

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

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

$$gain\,(X, humidity, 83.5) = 0.94 - \frac{7}{14} \cdot 0.59$$

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

$$gain\,(X, humidity, 83.5) = 0.94 - \frac{7}{14} \cdot 0.59 - \frac{7}{14} \cdot 0.98$$

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

$$gain\,(X, humidity, 83.5) = 0.94 - \frac{7}{14} \cdot 0.59 - \frac{7}{14} \cdot 0.98 \approx \mathbf{0.152}$$

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

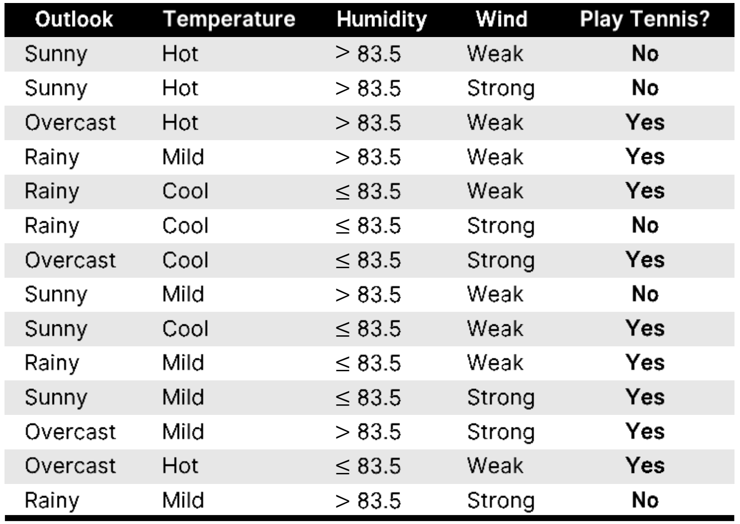
Gain for every candidate

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 numerical attributes

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

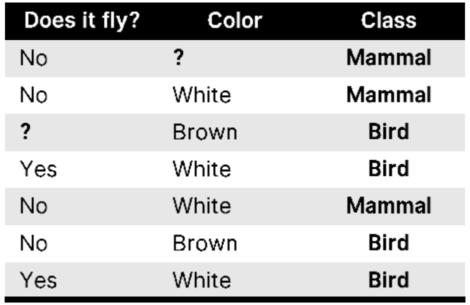


Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	> 83.5	Weak	**No**
Sunny	Hot	> 83.5	Strong	**No**
Overcast	Hot	> 83.5	Weak	**Yes**
Rainy	Mild	> 83.5	Weak	**Yes**
Rainy	Cool	≤ 83.5	Weak	**Yes**
Rainy	Cool	≤ 83.5	Strong	**No**
Overcast	Cool	≤ 83.5	Strong	**Yes**
Sunny	Mild	> 83.5	Weak	**No**
Sunny	Cool	≤ 83.5	Weak	**Yes**
Rainy	Mild	≤ 83.5	Weak	**Yes**
Sunny	Mild	≤ 83.5	Strong	**Yes**
Overcast	Mild	> 83.5	Strong	**Yes**
Overcast	Hot	≤ 83.5	Weak	**Yes**
Rainy	Mild	> 83.5	Strong	**No**

- 83.5 is the best splitting value for **Humidity** with an information gain of 0.152
- **Humidity** is now treated as a categorical attribute with two possible values
- A new optimal split is computed at every level of the tree
- A numerical attribute can be used several times in the tree, with different split values

Handling Missing Values

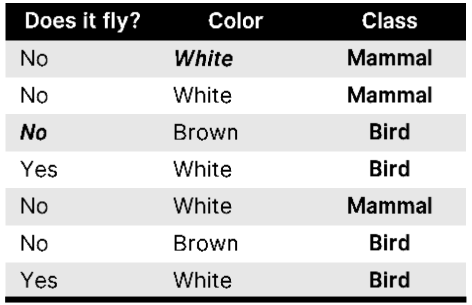
Handling missing values at training time



Does it fly?	Color	Class
No	**?**	**Mammal**
No	White	**Mammal**
?	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:

Handling missing values at training time



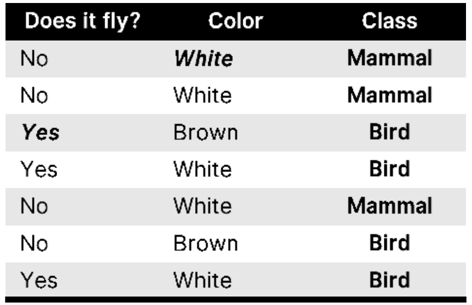
Does it fly?	Color	Class
No	***White***	**Mammal**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**

4 No	2 Brown
2 Yes	**4 White**

- 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

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**

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

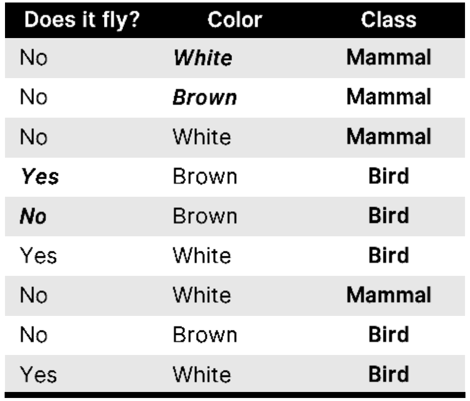
$$P(No|Bird) = \frac{1}{3} = 0.33$$

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

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

- 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

Handling missing values at training time

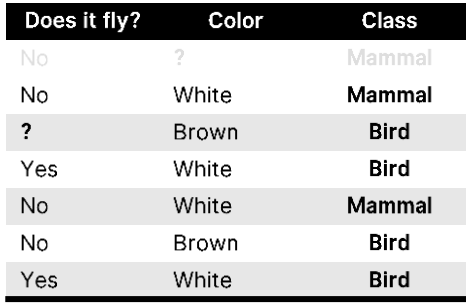


Does it fly?	Color	Class
No	***White***	**Mammal**
No	***Brown***	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
No	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**

- 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

Handling missing values at training time



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

$$\text{entropy}(X_{color=brown}) = 0$$

$$\text{entropy}(X_{color=white}) = 1$$

$$\text{gain}(X|color) = 0.985 - \frac{2}{6} \cdot 0 - \frac{4}{6} \cdot 1$$
$$= 0.318$$

- 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

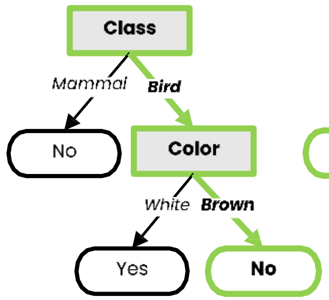
➢ Leave them unknown, but discard the sample when evaluating the gain of that attribute

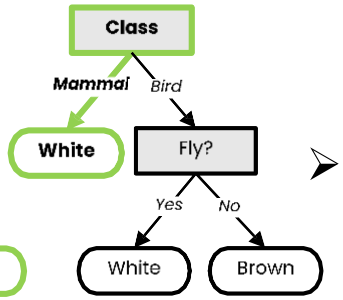
(if the attribute is chosen for splitting, send the instances with unknown values to all children)

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**

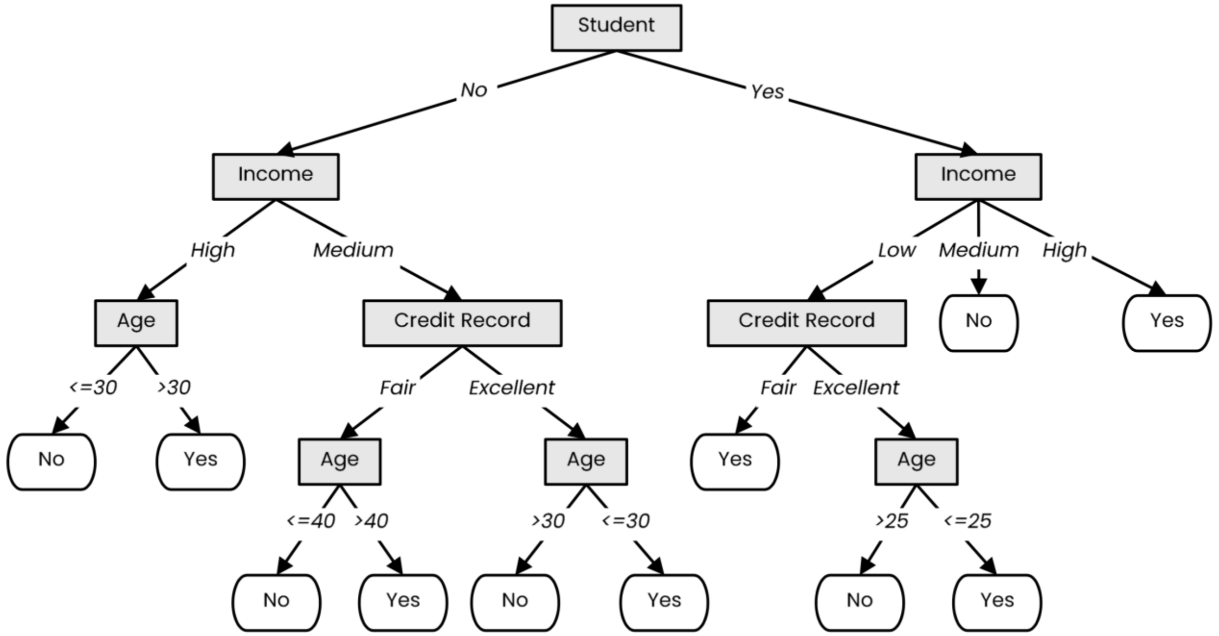
- Data sets might have samples with missing values for some attributes
- Simply ignoring them would mean throwing away a lot of information
- There are better ways of handling missing values:
- ➢ Set them to the most common value
- ➢ Set them to the most probable value given the label
- ➢ Add a new instance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





Handling missing values at inference time

- When we encounter a node that checks an attribute with a missing value, we explore all possibilities

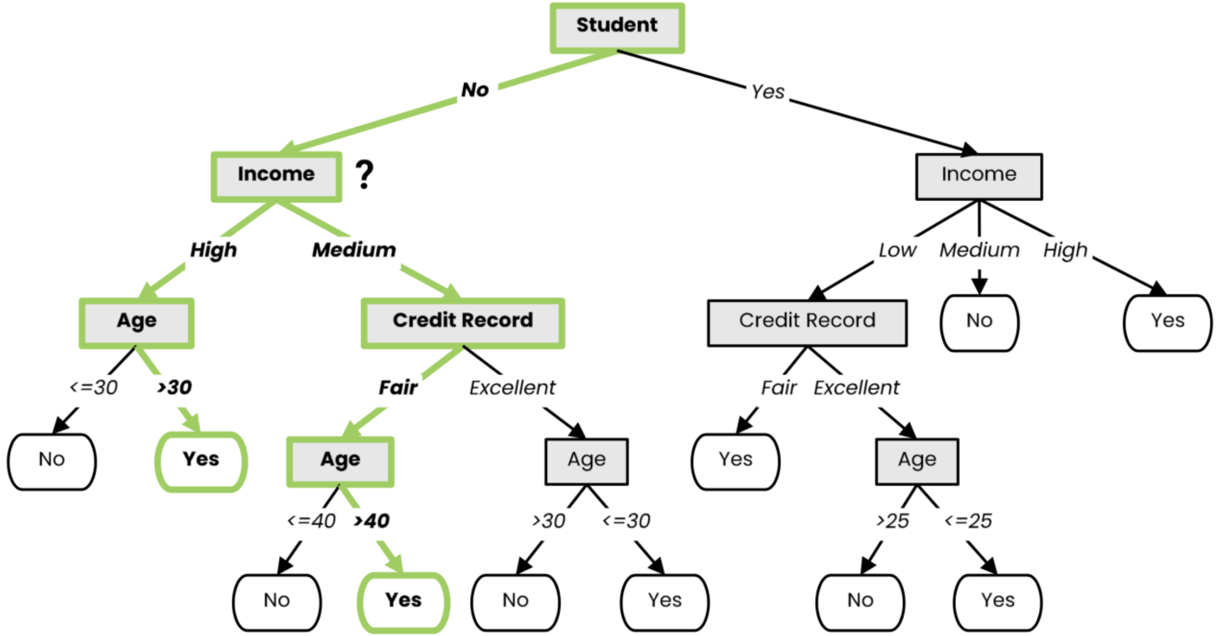


Loan?

- Not a student
- 49 years old
- Unknown income
- Fair credit record

Handling missing values at inference time

- When we encounter a node that checks an attribute with a missing value, we explore all possibilities
- We explore all branches and take the final prediction based on a (weighted) vote of the corresponding leaf nodes



Loan?

- Not a student
- 49 years old
- Unknown income
- Fair credit record
- ➢ Yes

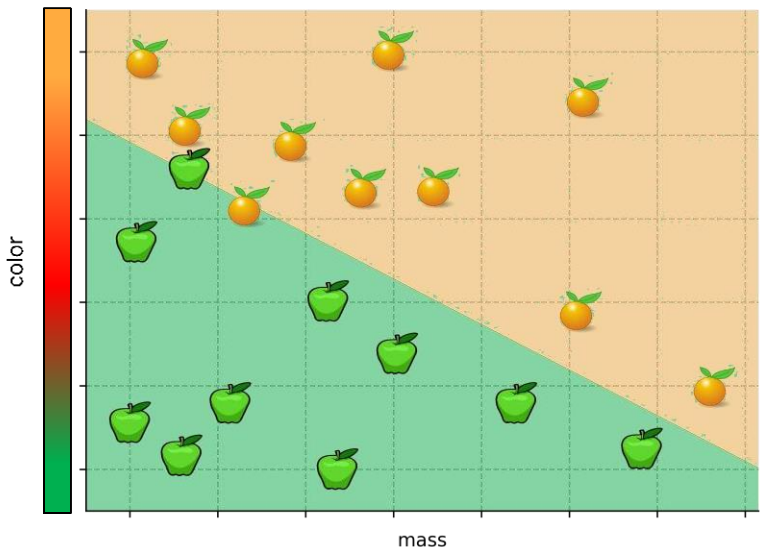
C4.5 Algorithm

- C4.5 algorithm is an extension of ID3 algorithm that brings several improvements:
  - ➢ Ability to handle both categorical (discrete) and numerical (continuous) attributes
    (continuous attributes are split by finding a best-splitting threshold)
  - ➢ Ability to handle missing values both at training and inference time (missing values at training are not used when information gain is computed; missing values at inference time are handled by exploring all corresponding branches)
  - ➢ Ability to handle attributes with different costs
  - ➢ Post-pruning in a bottom-up manner for removing branches that decrease validation error (i.e., that increase generalization capacity)

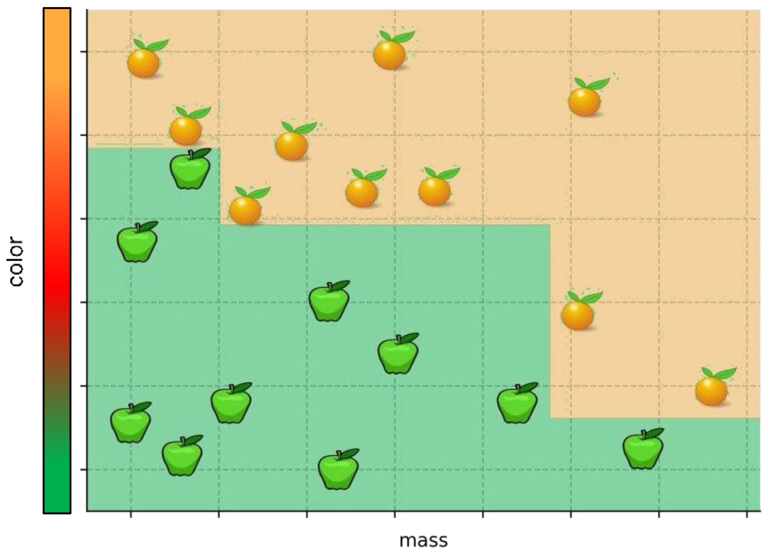
Decision Boundaries

- Decision trees produce non-linear decision boundaries

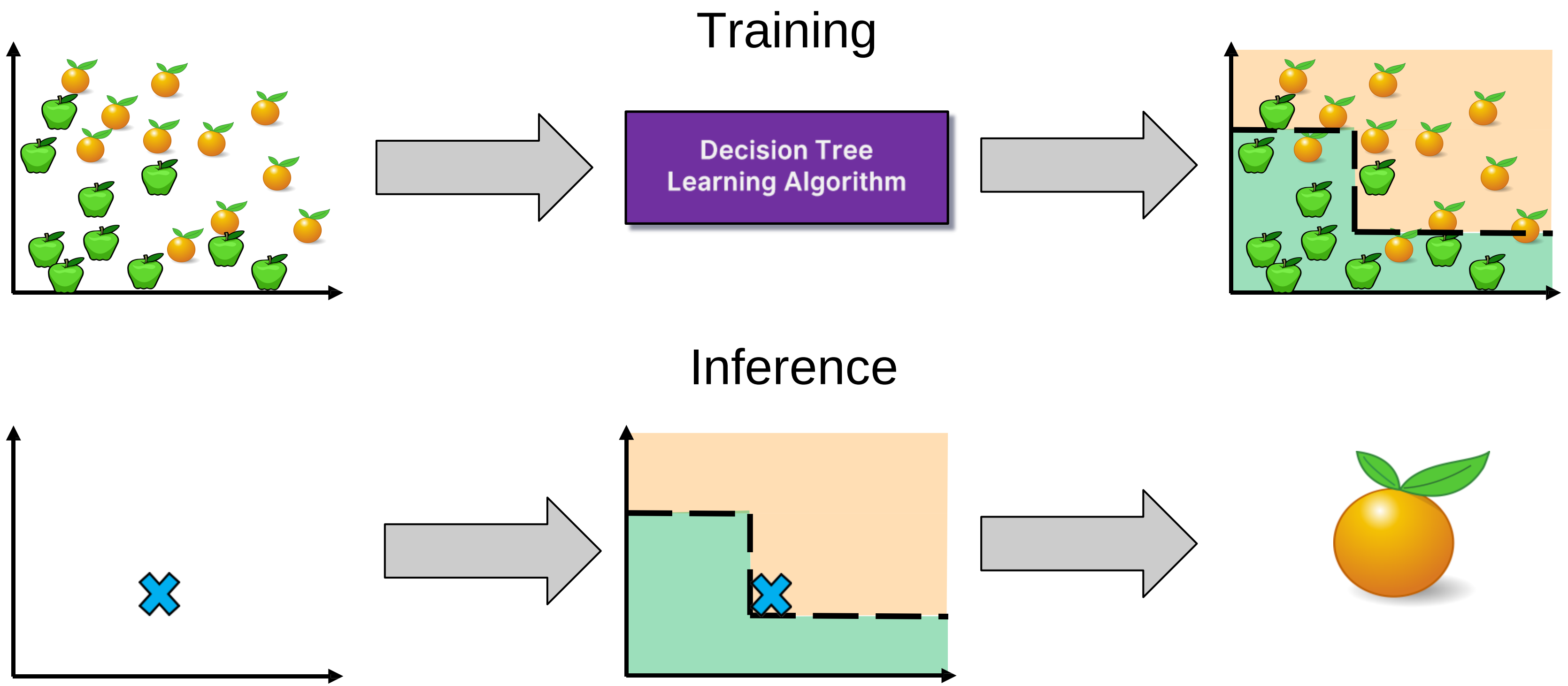
Support Vector Machines



Decision Tree



Decision Trees: Training and Inference



History of Decision Trees

- The first regression tree algorithm
  - ➢ “Automatic Interaction Detection (AID)” [Morgan & Sonquist, 1963]
- The first classification tree algorithm
  - ➢ “Theta Automatic Interaction Detection (THAID)” [Messenger & Mandel, 1972]
- Decision trees become popular
  - ➢ “Classification and regression trees (CART)” [Breiman et al., 1984]
- Introduction of the ID3 algorithm
  - ➢ “Induction of Decision Trees” [Quinlan, 1986]
- Introduction of the C4.5 algorithm
  - ➢ “C4.5: Programs for Machine Learning” [Quinlan, 1993]

Summary

- Decision trees represent a tool based on a tree-like graph of decisions and their possible outcomes
- Decision tree learning is a machine learning method that employs a decision tree as a predictive model
- ID3 builds a decision tree by iteratively splitting the data based on the values of an attribute with the largest information gain (decrease in entropy)
    - ➢ Using the decrease of Gini Impurity is also a commonly-used option in practice
- C4.5 is an extension of ID3 that handles attributes with continuous values, missing values and adds regularization by pruning branches likely to overfit

Random Forests

(Ensemble learning with decision trees)

Random Forests

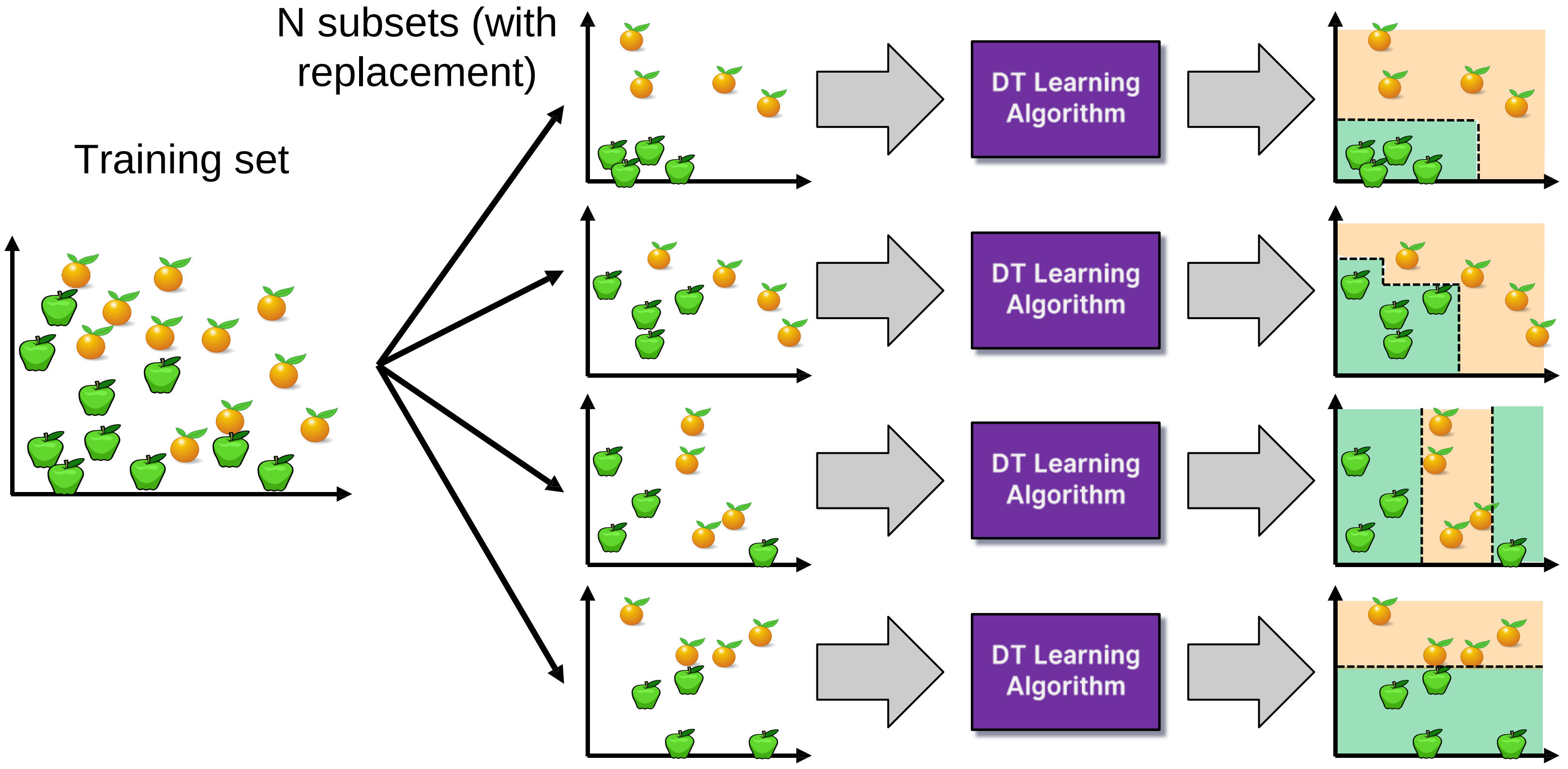
- Random Forests:
  - ➢ Instead of building a single decision tree and use it to make predictions, build many slightly different trees and combine their predictions

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

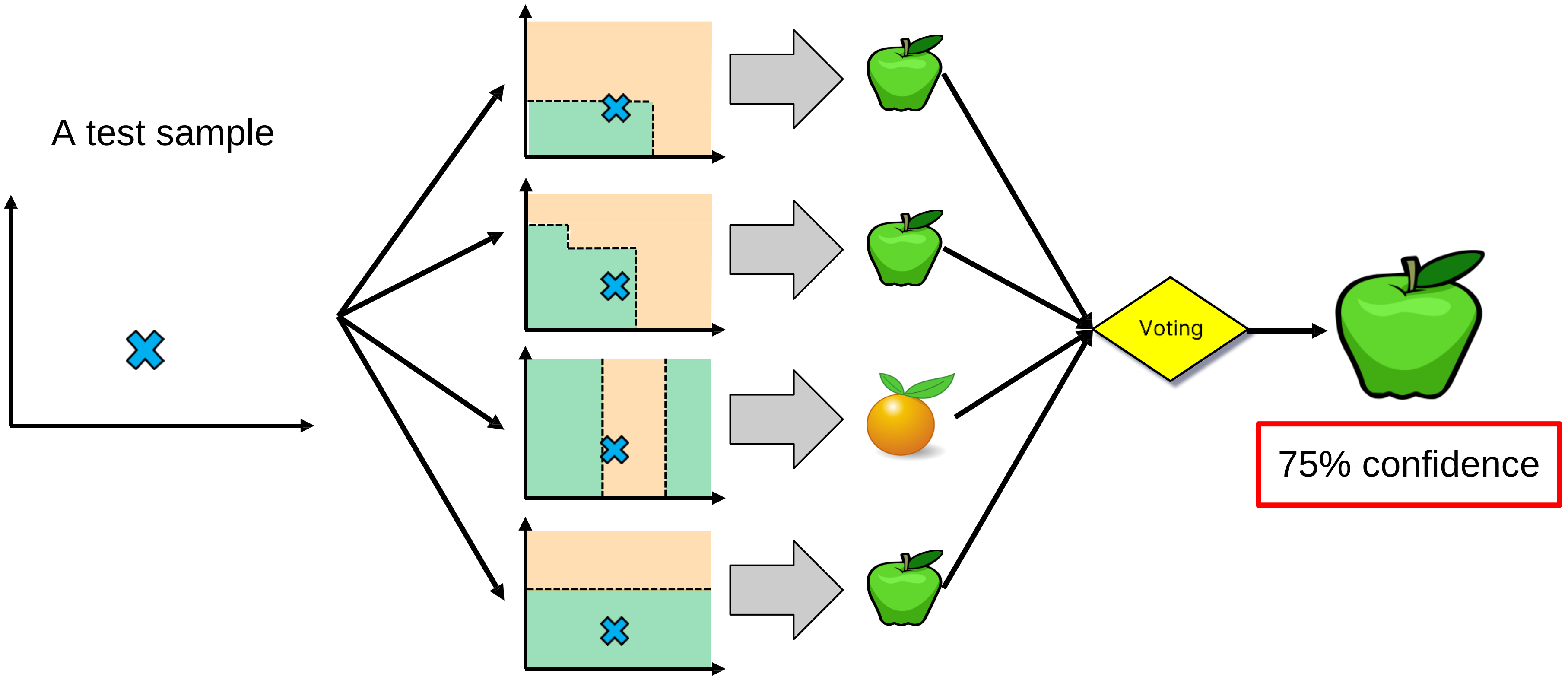
1. Bagging (**B**ootstrap **Agg**regat**ing**):
   - ➢ Take random subsets of data points from the training set to create N smaller data sets
   - ➢ Fit a decision tree on each subset

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



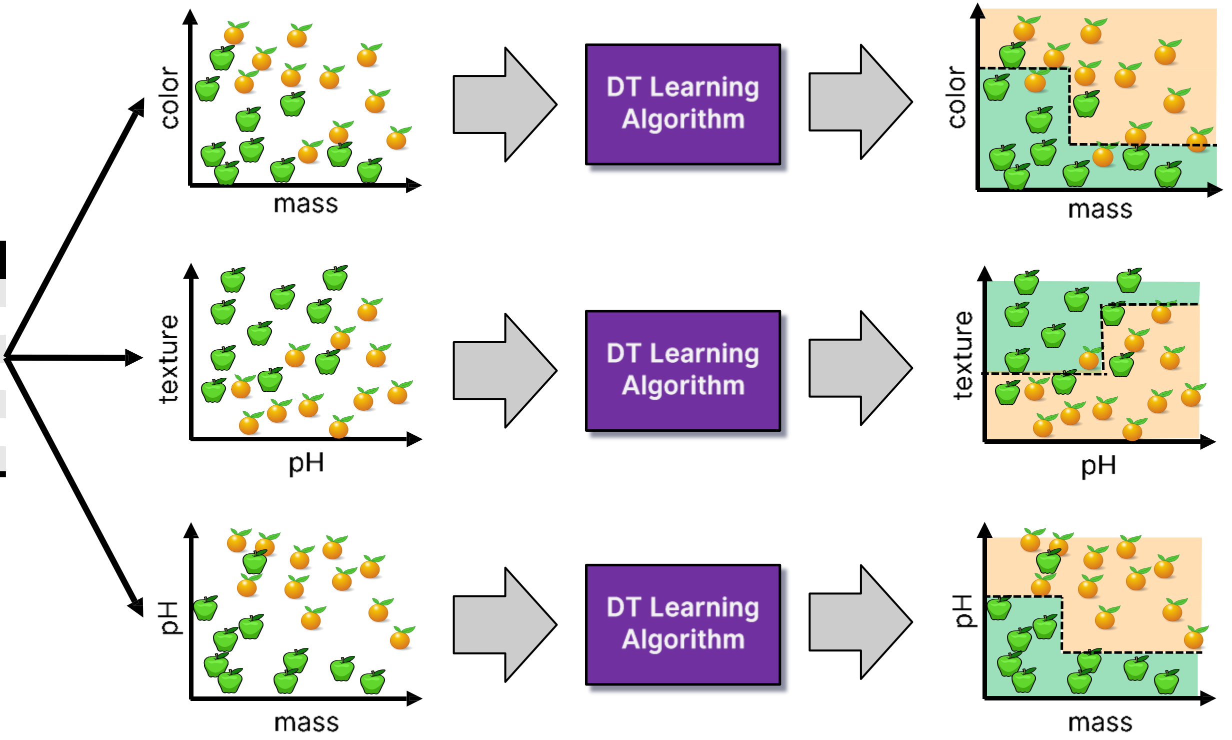
Bagging at inference time



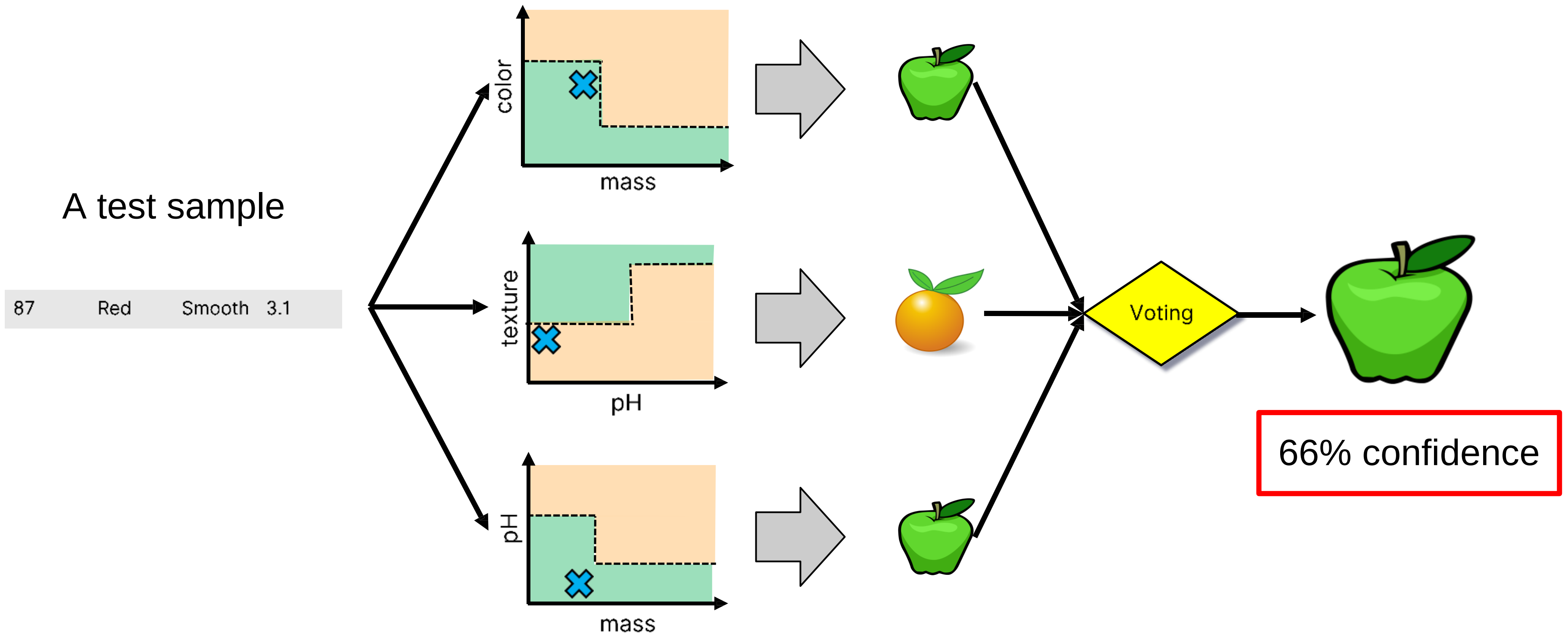
Random Subspace Method at training time

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**

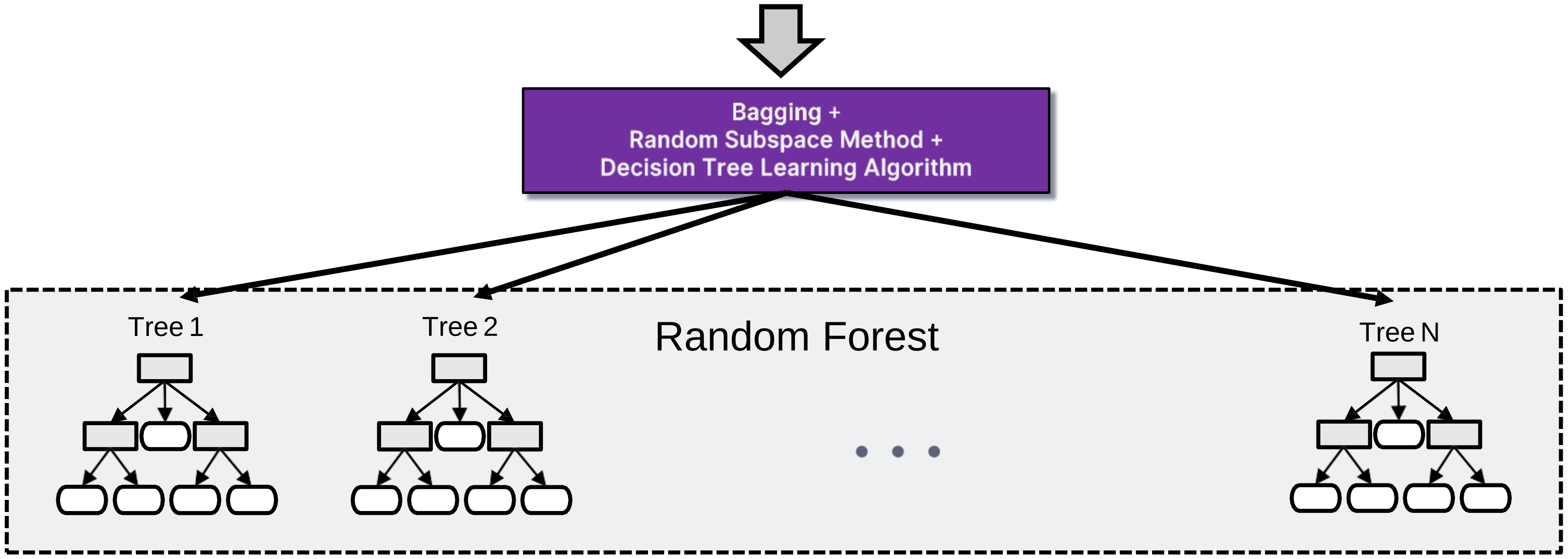


Random Subspace Method at inference time



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**



History of Random Forests

- Introduction of the Random Subspace Method
  - “Random Decision Forests” [Ho, 1995] and “The Random Subspace Method for Constructing Decision Forests” [Ho, 1998]

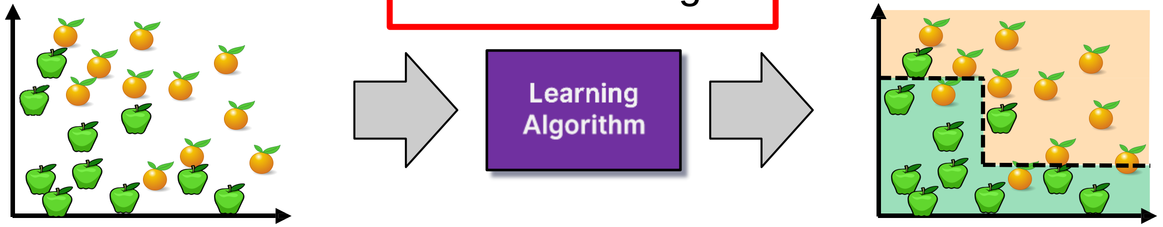
- Combined the Random Subspace Method with Bagging. Introduce the term Random Forest (a trademark of Leo Breiman and Adele Cutler)
  - “Random Forests” [Breiman, 2001]

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

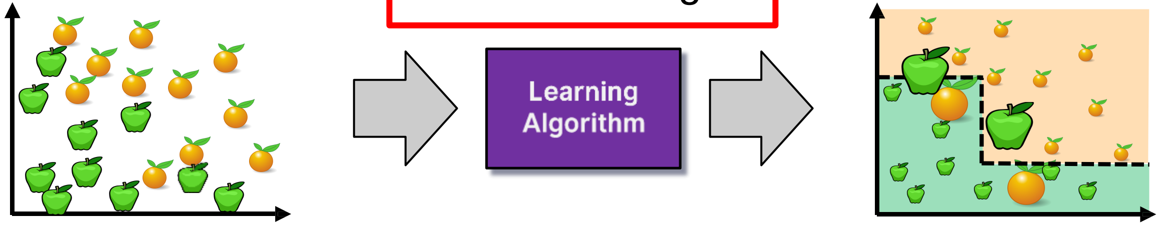
Boosting

All samples have the same weight



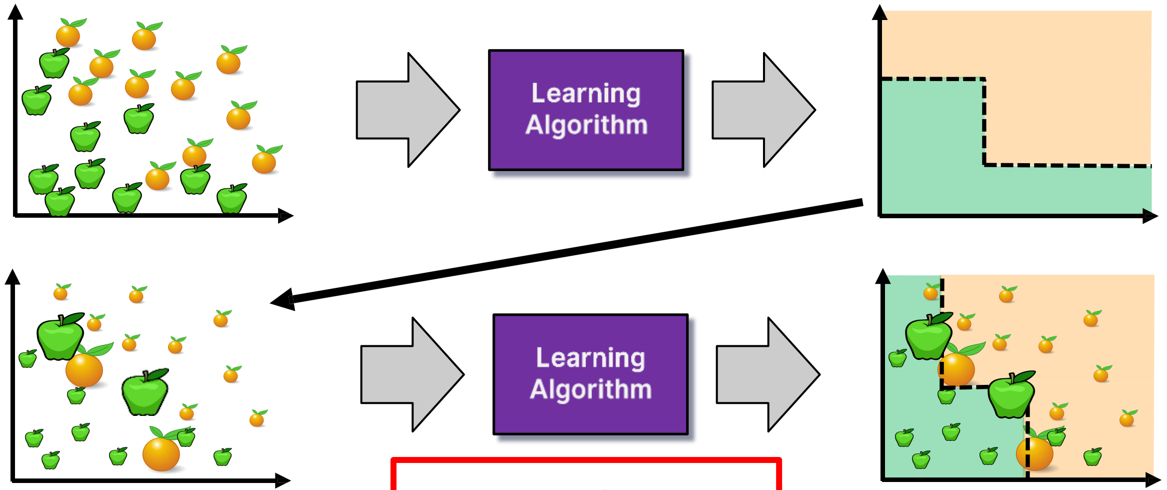
Boosting

All samples have the same weight



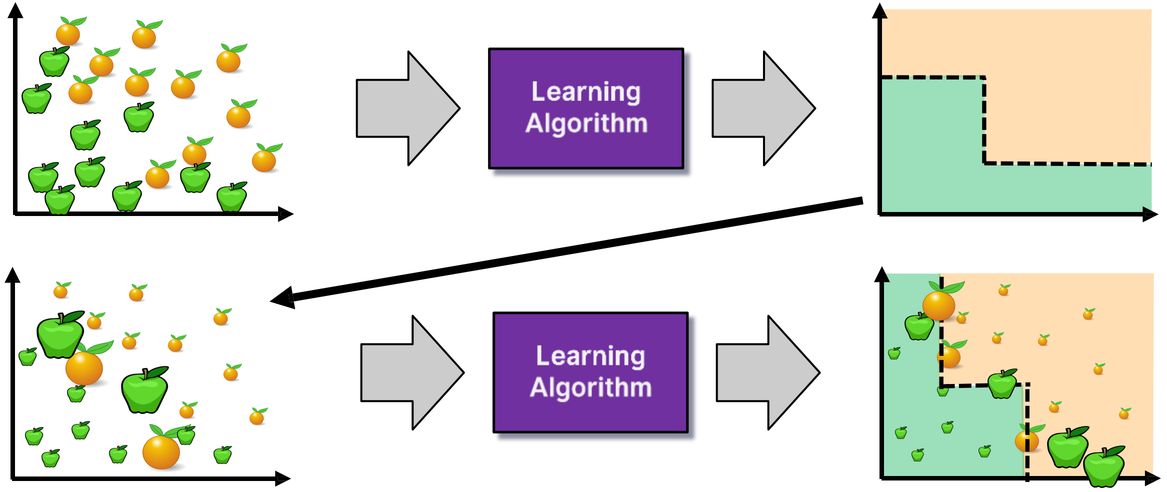
Reweight based on model’s mistakes

Boosting



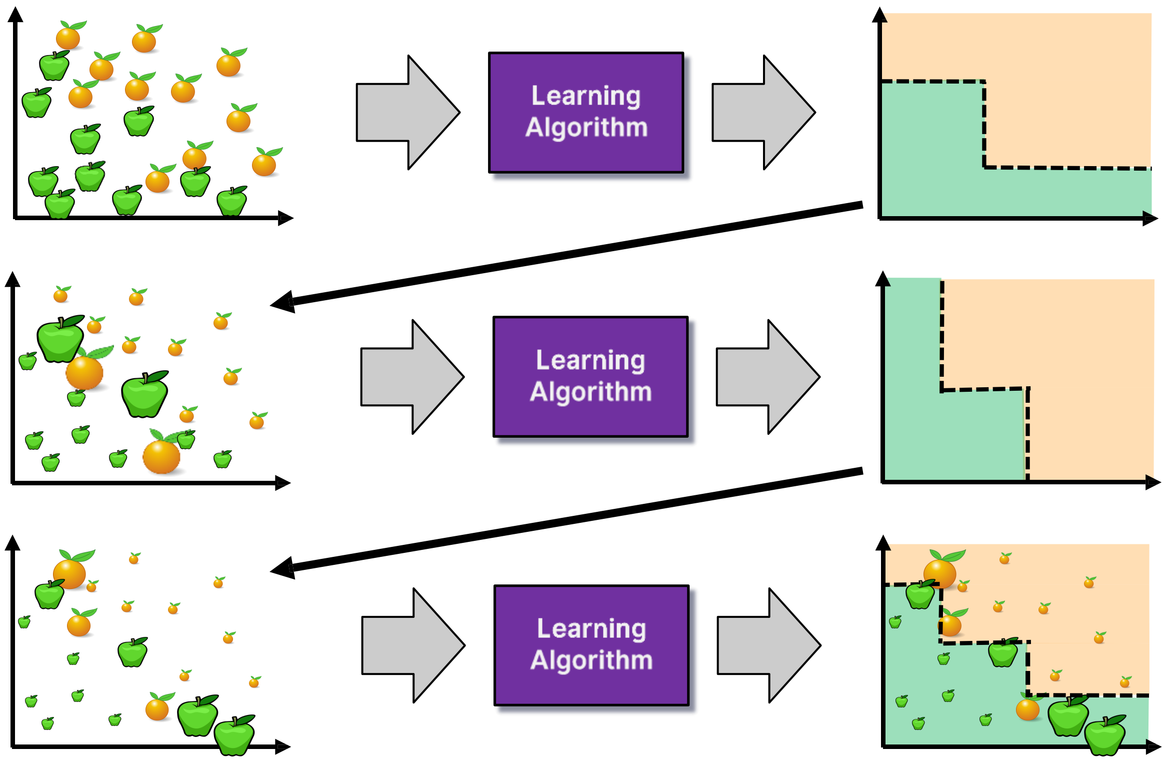
Next model sees weighted samples

Boosting

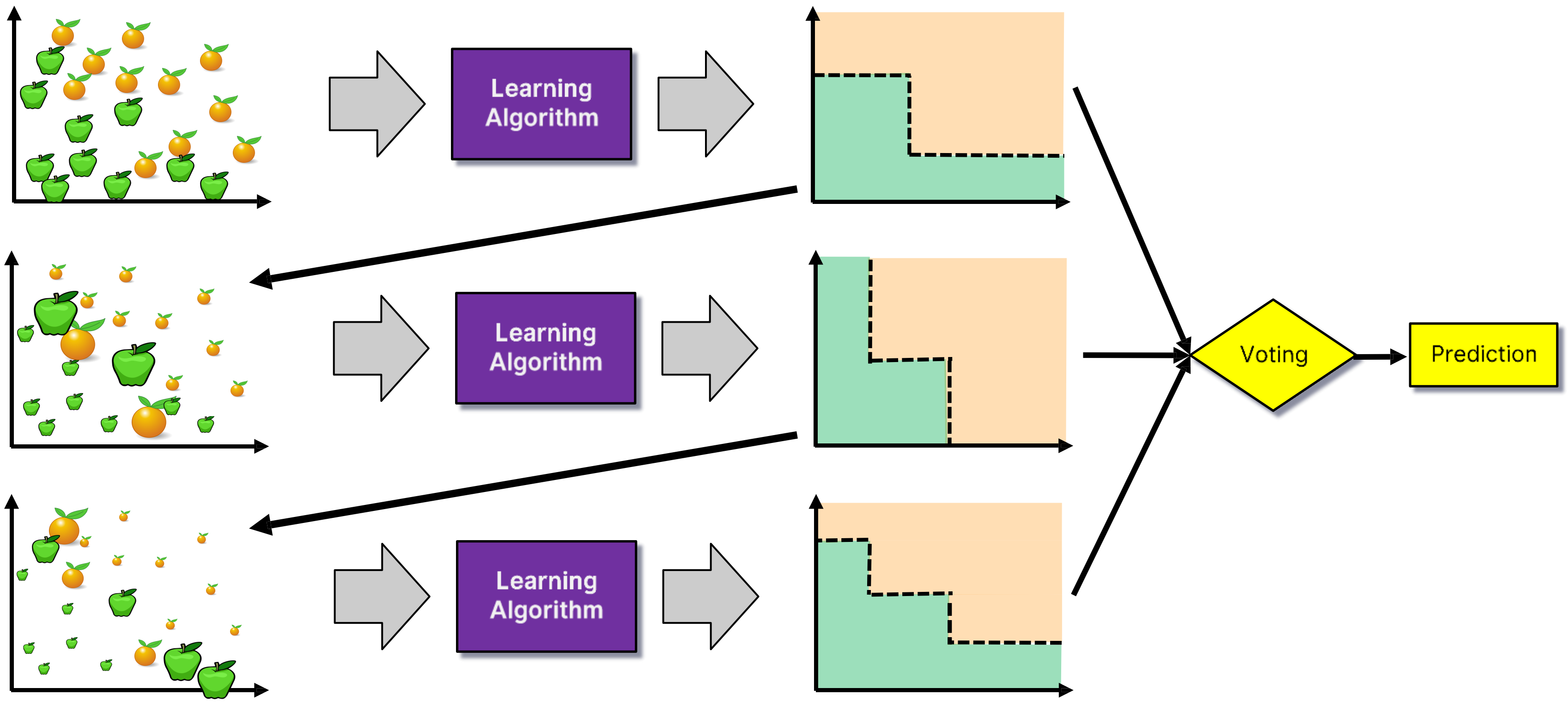


Reweight based on current model's mistakes

Boosting


Learning Algorithm
Learning Algorithm
Learning Algorithm

Boosting


Learning Algorithm
Learning Algorithm
Learning Algorithm
Voting
Prediction

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!

Decision Trees and Random Forest (Python)

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

clf = DecisionTreeClassifier(criterion = "entropy", min_samples_leaf = 3)
# Lots of parameters: criterion = "gini" / "entropy";
#                     max_depth;
#                     min_impurity_split;

clf.fit(X, y) # It can only handle numerical attributes!
# Categorical attributes need to be encoded, see LabelEncoder and OneHotEncoder

clf.predict([x]) # Predict class for x

clf.feature_importances_ # Importance of each feature
clf.tree_ # The underlying tree object

clf = RandomForestClassifier(n_estimators = 20) # Random Forest with 20 trees
```