

Decision Trees

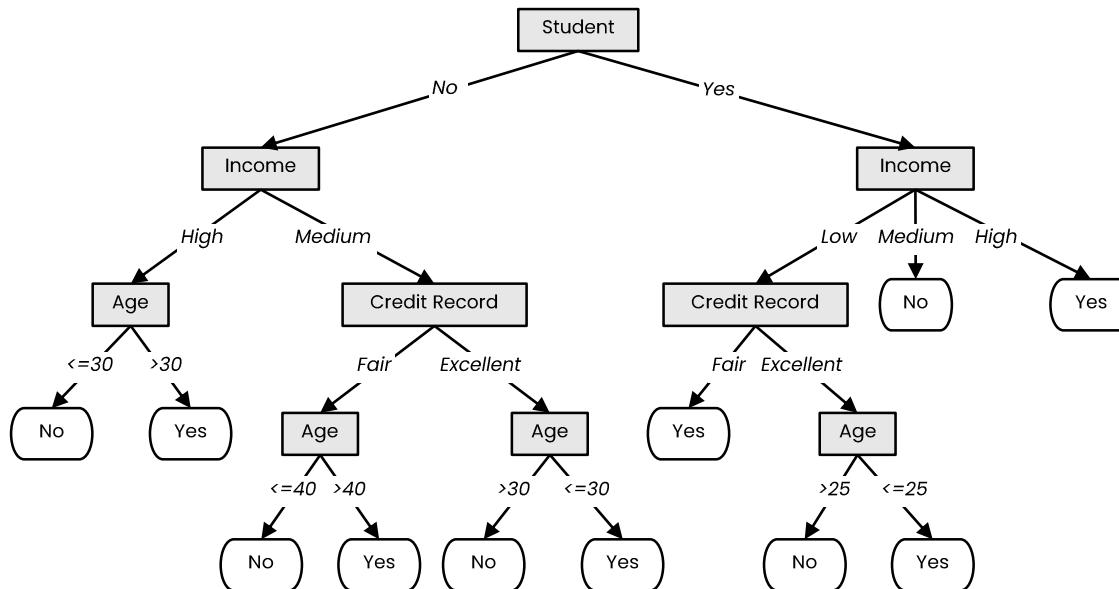
Making predictions with a **tree-like structure**

Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

Academic Year 2018/2019, 1st Semester

Definition

- A **tree-like model** which illustrates series of events which lead to certain *decisions*.
- Each node of the tree represents a “test” on an attribute and each branch, an outcome of that test.



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

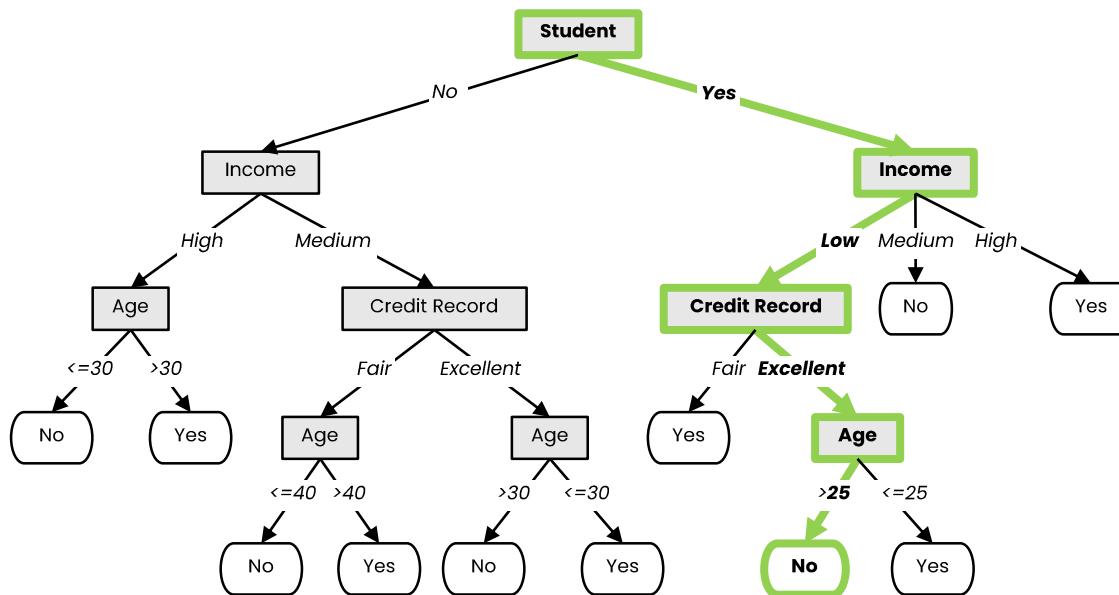
Loan default?



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

Definition

- A **tree-like model** which illustrates series of events which lead to certain *decisions*.
- Each node of the tree represents a “test” on an attribute and each branch, an outcome of that test.



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

No

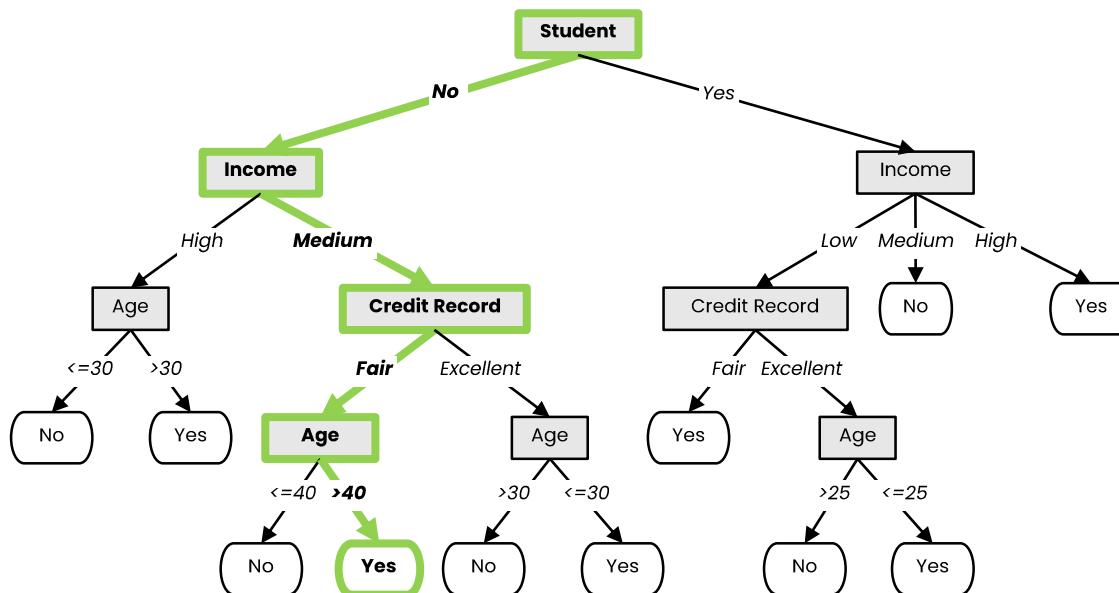
Loan default?



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

Definition

- A **tree-like model** which illustrates series of events which lead to certain *decisions*.
- Each node of the tree represents a “test” on an attribute and each branch, an outcome of that test.



No

Loan default?



Yes

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

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

Decision Tree Learning

Decision Tree Learning

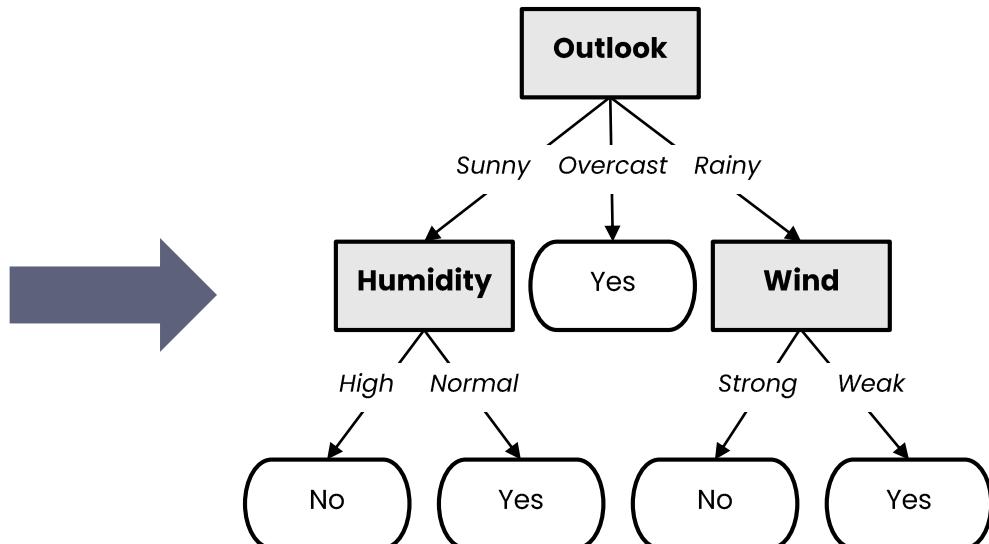
- In ML, we want to use labeled data to obtain a *suitable* decision tree for future predictions.
 - We want a tree which works well on unseen data, while asking as few questions as possible.

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

- In ML, we want to use labeled data to obtain a *suitable* decision tree for future predictions.
 - We want a tree which 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



How do we learn a decision tree?

- The basic idea is to *choose an attribute* and, based on its values, split the data into smaller sets.
- Recursively do this until we are *sure of 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

How do we learn a decision tree?

- The basic idea is to *choose an attribute* and, based on its values, split the data into smaller sets.
- Recursively do this until we are *sure of 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

Outlook

How do we learn a decision tree?

- The basic idea is to *choose an attribute* and, based on its values, split the data into smaller sets.
- Recursively do this until we are *sure of the label*.

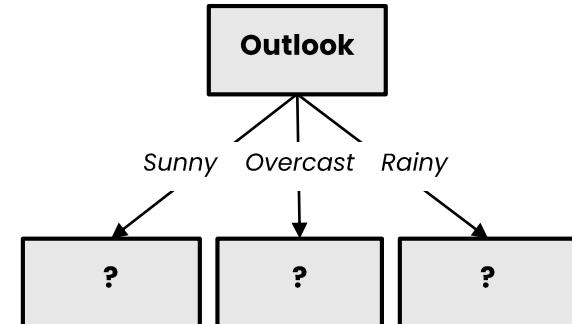
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

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



How do we learn a decision tree?

- The basic idea is to *choose an attribute* and, based on its values, split the data into smaller sets.
- Recursively do this until we are *sure of the label*.

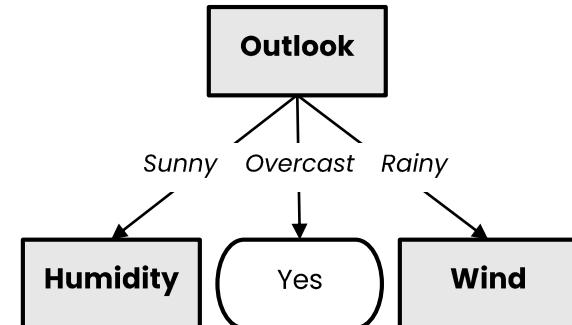
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

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



How do we learn a decision tree?

- The basic idea is to *choose an attribute* and, based on its values, split the data into smaller sets.
- Recursively do this until we are *sure of 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

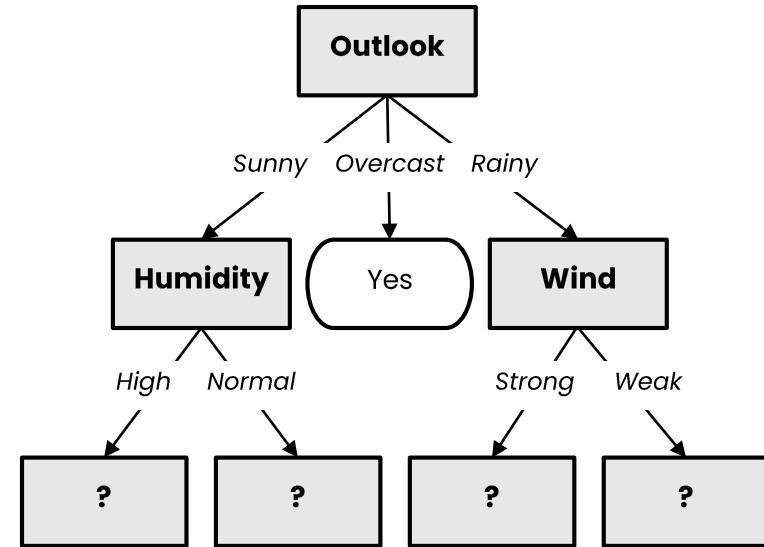
Outlook = Overcast

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



How do we learn a decision tree?

- The basic idea is to *choose an attribute* and, based on its values, split the data into smaller sets.
- Recursively do this until we are *sure of 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

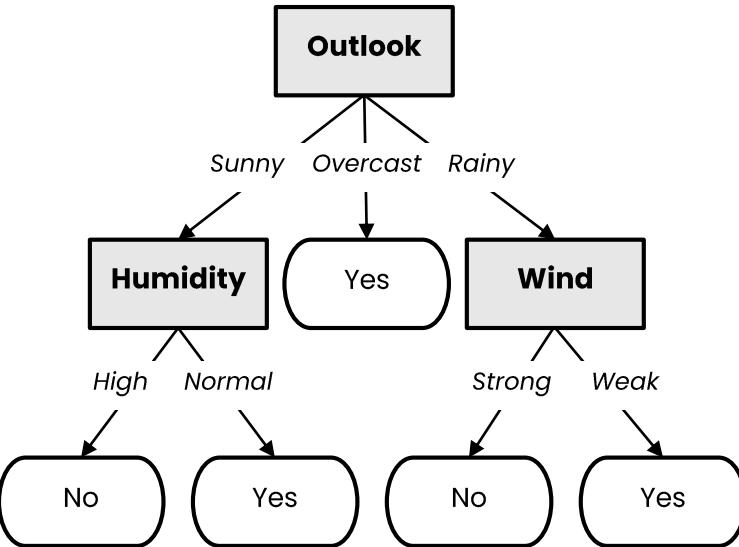
Outlook = Overcast

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 Algorithm

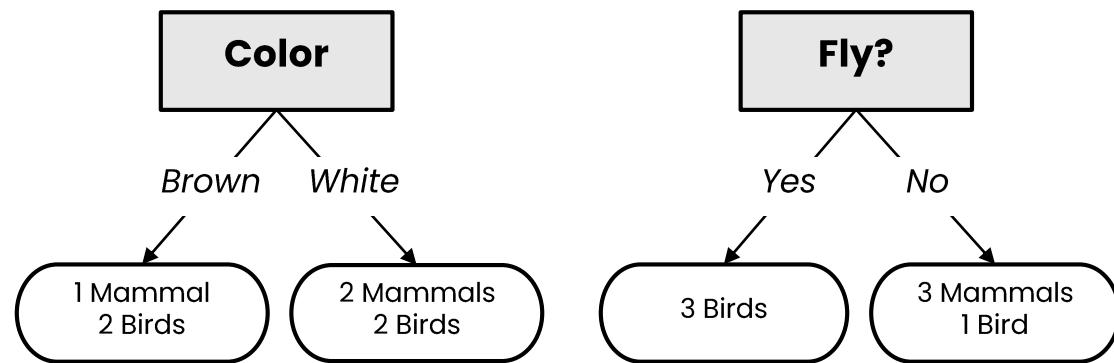
```
1 def extract_node(X):
2     node = TreeNode(X)
3     if should_be_leaf_node(X):
4         node.label = majority_label(X)
5     else:
6         a = select_best_splitting_attribute(X)
7         for v in values(a):
8              $X_v = \{x \in X \mid x[a] == v\}$ 
9             node.children.append(extract_node( $X_v$ ))
10    return node
```

What makes 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

What makes a “good” attribute?

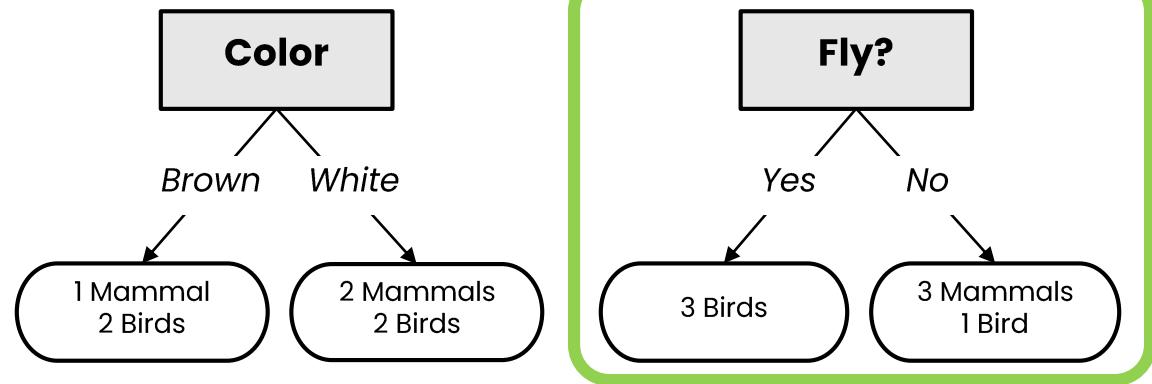
Does it fly?	Color	Class
No	Brown	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird



- Which one is a *better* attribute for splitting?

What makes a “good” attribute?

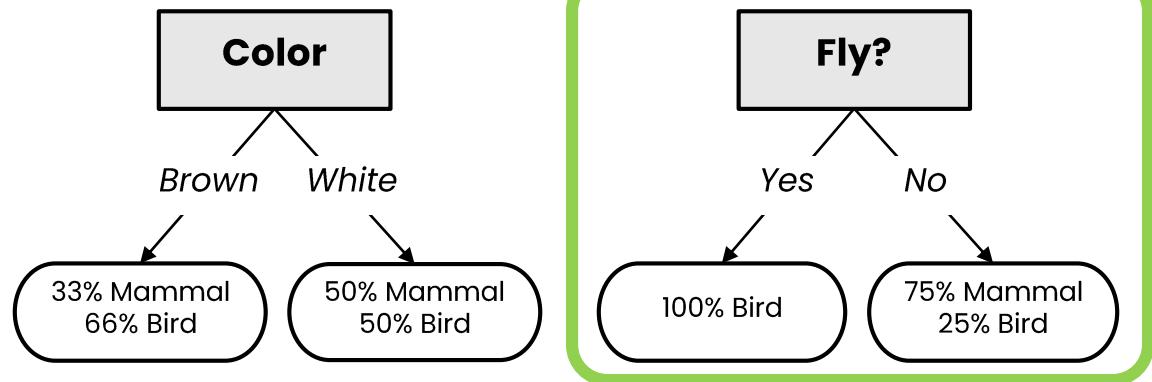
Does it fly?	Color	Class
No	Brown	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird



- Which one is a *better* attribute for splitting?
- Why?

What makes 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 one is a *better* attribute for splitting?
- Why?
 - Because the resulting subsets are more “pure”.
 - Knowing the value of this attribute gives us “*more information*” about the label.
 - i.e. The **entropy** of the subsets is lower.

Information Gain

Entropy

- Entropy measures the *degree of randomness* in a dataset, with lower entropy implying greater *predictability*.

Low entropy



High entropy



Entropy

- Entropy measures the *degree of randomness* in a dataset, with lower entropy implying greater *predictability*.

Low entropy



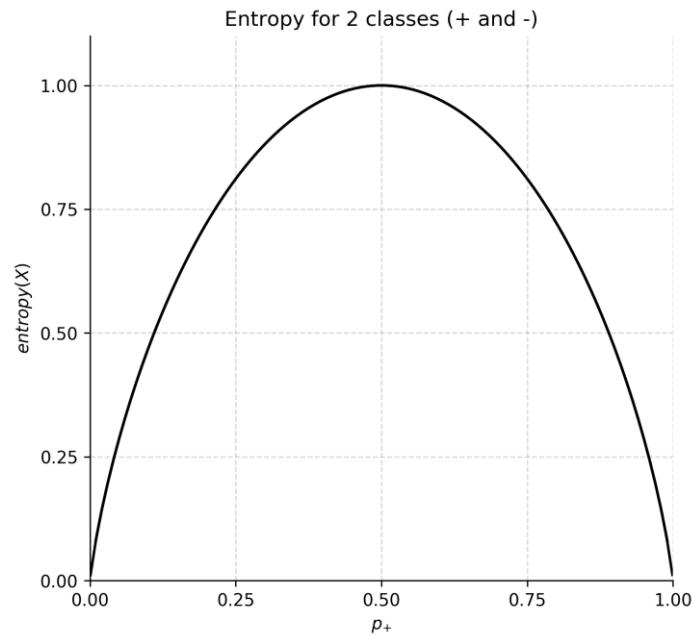
High entropy



- For a set of samples X with k classes:

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

where p_i is the proportion of elements with class i



Information Gain

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

Information Gain

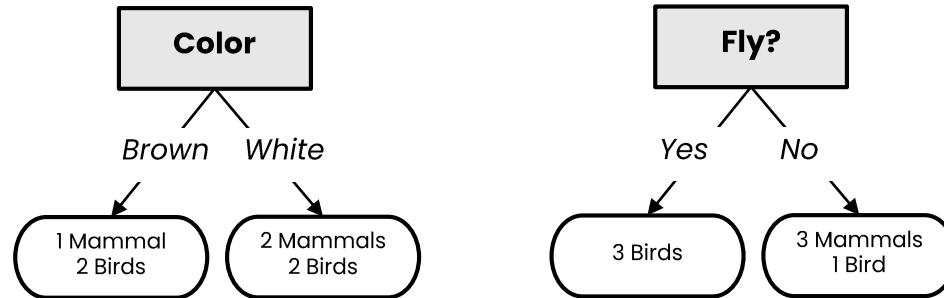
- The **information gain** of an attribute a is the expected *reduction in entropy* due to splitting on values of a :

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

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

Best attribute = highest information gain

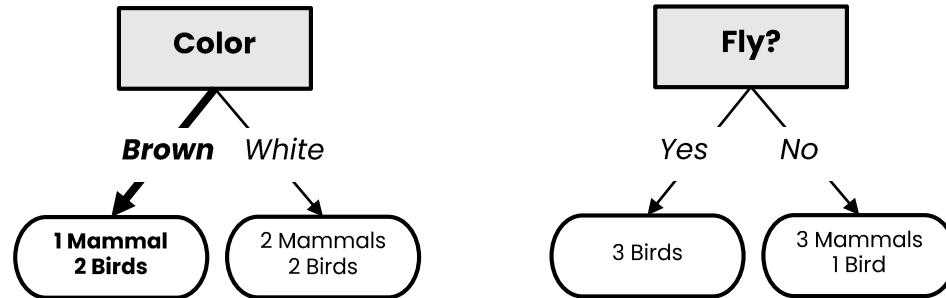
Does it fly?	Color	Class
No	Brown	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird



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

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

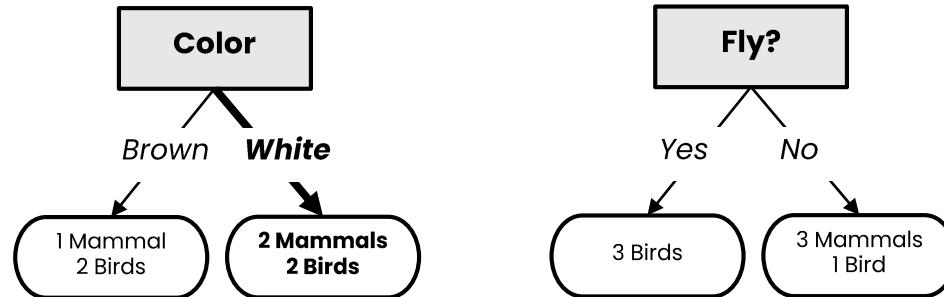


$$\text{entropy}(X) = -p_{\text{mammal}} \log_2 p_{\text{mammal}} - p_{\text{bird}} \log_2 p_{\text{bird}} = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \approx 0.985$$

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

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



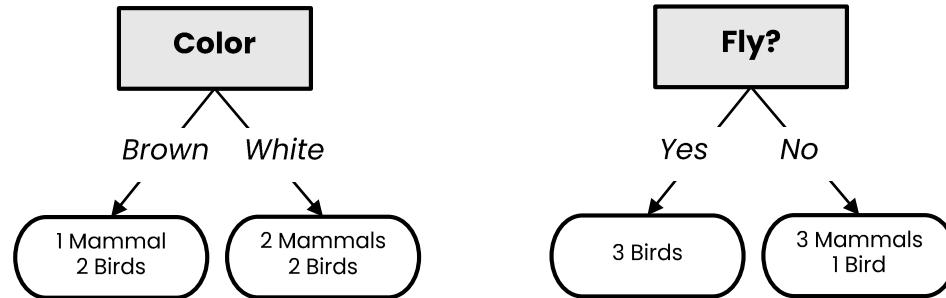
$$\text{entropy}(X) = -p_{\text{mammal}} \log_2 p_{\text{mammal}} - p_{\text{bird}} \log_2 p_{\text{bird}} = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \approx 0.985$$

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

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

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



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

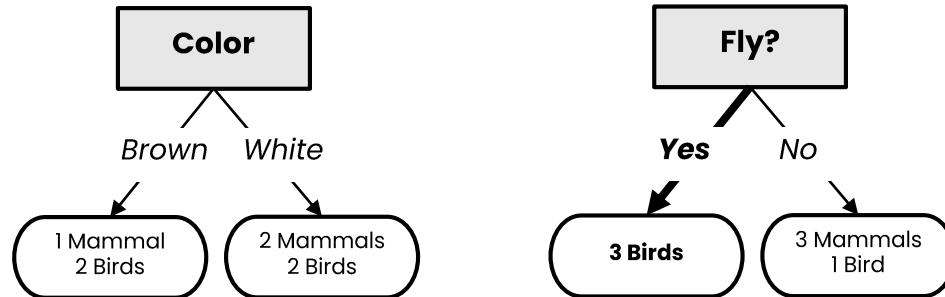
$$\text{entropy}(X_{\text{color=brown}}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} \approx 0.918$$

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

$$\text{gain}(X, \text{color}) = 0.985 - \frac{3}{7} * 0.918 - \frac{4}{7} * 1 = 0.02$$

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



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

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

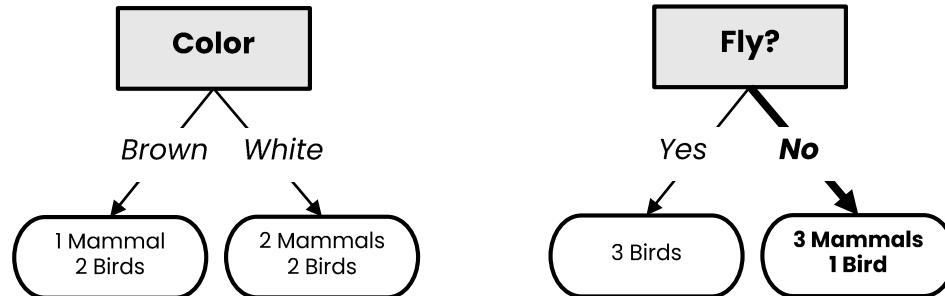
$$\text{entropy}(X_{\text{color=white}}) = 1$$

$$\text{gain}(X, \text{color}) = 0.985 - \frac{3}{7} * 0.918 - \frac{4}{7} * 1 = 0.02$$

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

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



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

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

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

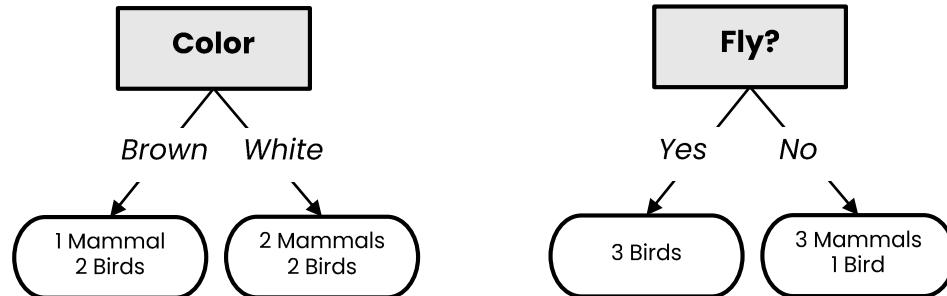
$$\text{gain}(X, \text{color}) = 0.985 - \frac{3}{7} * 0.918 - \frac{4}{7} * 1 = 0.02$$

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

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

Best attribute = highest information gain

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



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

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

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

$$\text{gain}(X, \text{color}) = 0.985 - \frac{3}{7} * 0.918 - \frac{4}{7} * 1 = 0.02$$

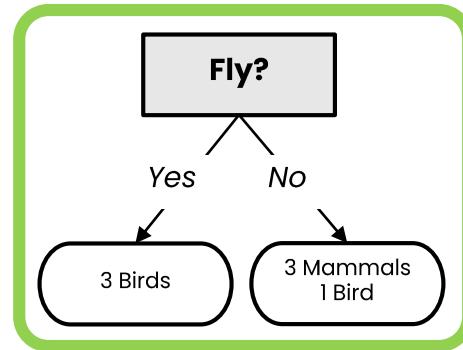
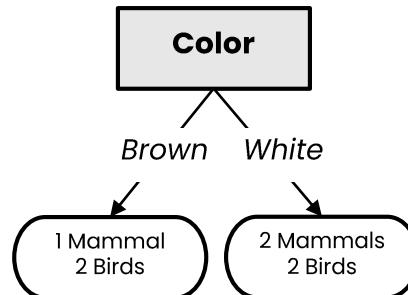
$$\text{entropy}(X_{\text{fly=yes}}) = 0$$

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

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

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



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

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

$$\text{gain}(X, \text{color}) = 0.985 - \frac{3}{7} * 0.918 - \frac{4}{7} * 1 = 0.02$$

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

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

In practice we don't need to compute $\text{entropy}(X)$ since it is the same for all attributes.

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

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

ID3 Algorithm

```
1 def ID3(X):
2     node = TreeNode(X)
3     if all_points_have_same_class(X):
4         node.label = majority_label(X)
5     else:
6         a = select_attribute_with_highest_information_gain(X)
7         if gain(X, a) == 0:
8             node.label = majority_label(X)
9         else:
10            for v in values(a):
11                X_v = {x ∈ X | x[a] == v}
12                node.children.append(ID3(X_v))
13
return node
```

ID stands for "Iterative Dichotomiser"

Gini Impurity

Gini impurity

- Measures how often a randomly chosen element would be incorrectly labeled if it was randomly labeled according to the distribution of labels.



Error of classifying
randomly picked
fruit with randomly
picked label



Gini impurity

- Measures how often a randomly chosen element would be incorrectly labeled if it was randomly labeled according to the distribution of labels.

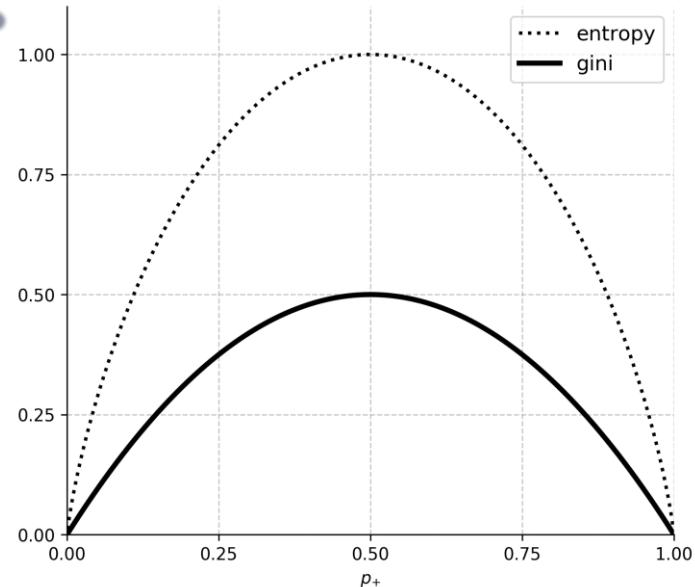


Error of classifying
randomly picked
fruit with randomly
picked label



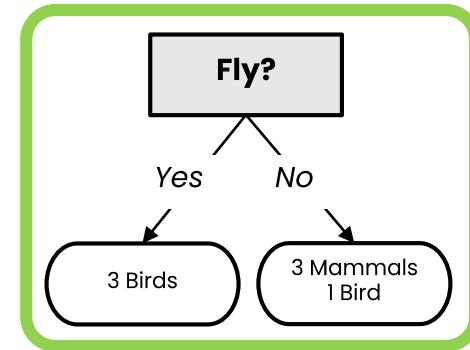
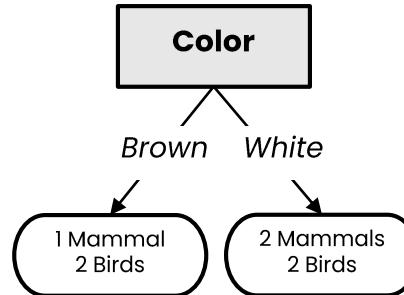
$$\text{gini}(X) = 1 - \sum_{i=1}^k p_i^2$$

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



Best attribute = highest impurity decrease

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



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

$$\text{gini}(X_{\text{color}=\text{brown}}) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 \approx 0.44$$

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

$$\Delta\text{gini}(X, \text{color}) = 0.489 - \frac{3}{7} * 0.44 - \frac{4}{7} * 0.5 \approx 0.01$$

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

$$\Delta\text{gini}(X, \text{fly}) = 0.489 - \frac{3}{7} * 0 - \frac{4}{7} * 0.375 = 0.274$$



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

In practice we don't need to compute $\text{gini}(X)$ since it is the same for all attributes.

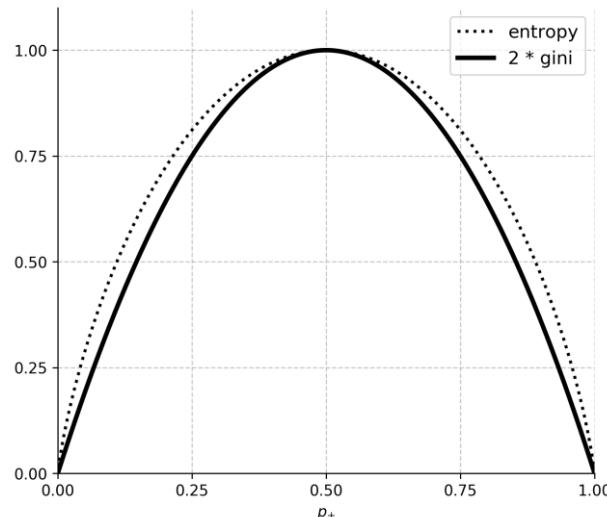
Entropy vs. Gini Impurity

- In practice, *entropy* and *gini impurity* are almost the same.
- They only disagree in about 2% of cases:

- “Theoretical Comparison between the Gini Index and Information Gain Criteria”

Laura Elena Raileanu, Kilian Stoffel, 2004

- Entropy might be slower to compute, because of the use of logarithms



Sklearn uses
gini by default.

Pruning

Pruning

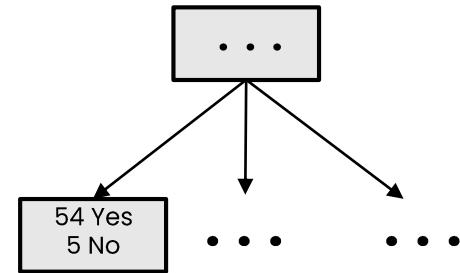
- **Pruning** is a technique which reduces the size of a decision tree by removing parts of the tree which provide little predictive power.
- Pruning is a *regularization* method which reduces the complexity of the final model, and hence reduce overfitting.
 - Decision trees are very prone to *overfitting*.
- *Pre-pruning:*
 - Stop the tree growing algorithm before it fully classifies the data
- *Post-pruning:*
 - Grow the tree completely, then replace some non-leaf nodes with leaf nodes if it improves validation error.

Pre-pruning

- Pre-pruning implies early stopping.
 - If *certain conditions* are met, the current node will *not be split*, even if it is impure and further splitting would be possible.
 - It will become a *leaf node* with the label of the majority class in the current set.
 - It is also possible to use the class distribution as the node's prediction confidence.
- Common stopping criteria include setting a threshold on:
 - Entropy (or impurity) of current set.
 - Number of samples in the current set.
 - Gain of the best-split attribute.
 - Depth of the tree

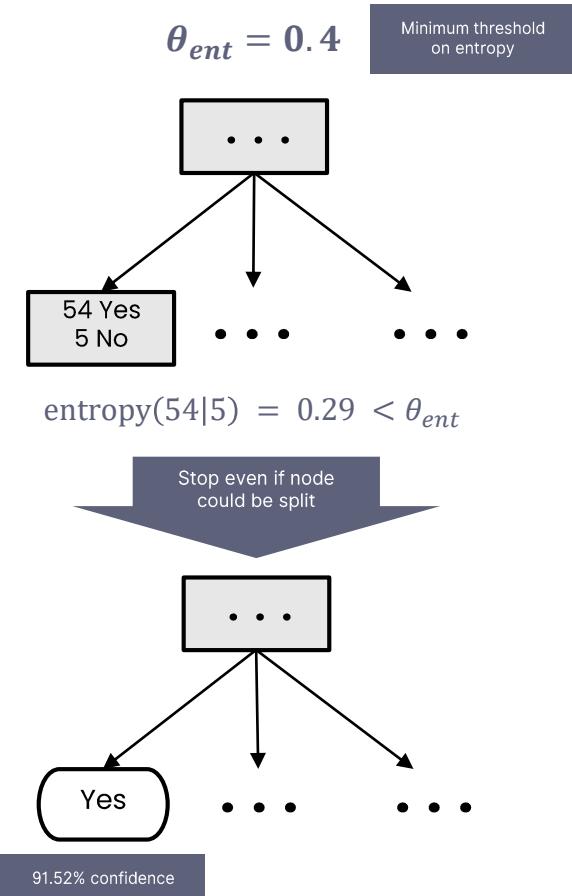
$$\theta_{ent} = 0.4$$

Minimum threshold
on entropy

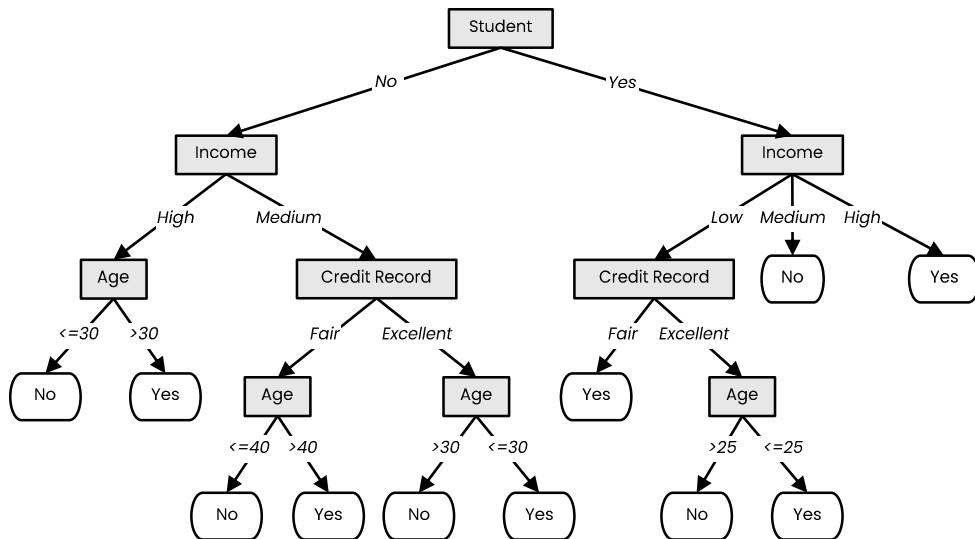


Pre-pruning

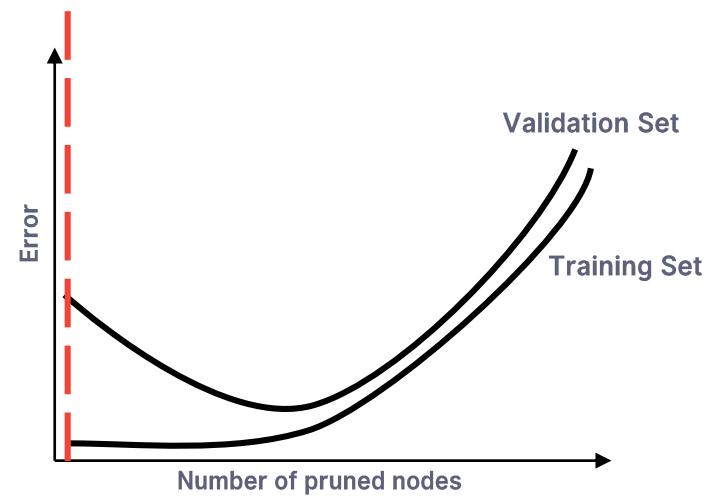
- Pre-pruning implies early stopping.
 - If certain conditions are met, the current node will *not be split*, even if it is impure and further splitting would be possible.
 - It will become a *leaf node* with the label of the majority class in the current set.
 - It is also possible to use the class distribution as the node's prediction confidence.
- Common stopping criteria include setting a threshold on:
 - Entropy (or impurity) of current set.
 - Number of samples in the current set.
 - Gain of the best-split attribute.
 - Depth of the tree



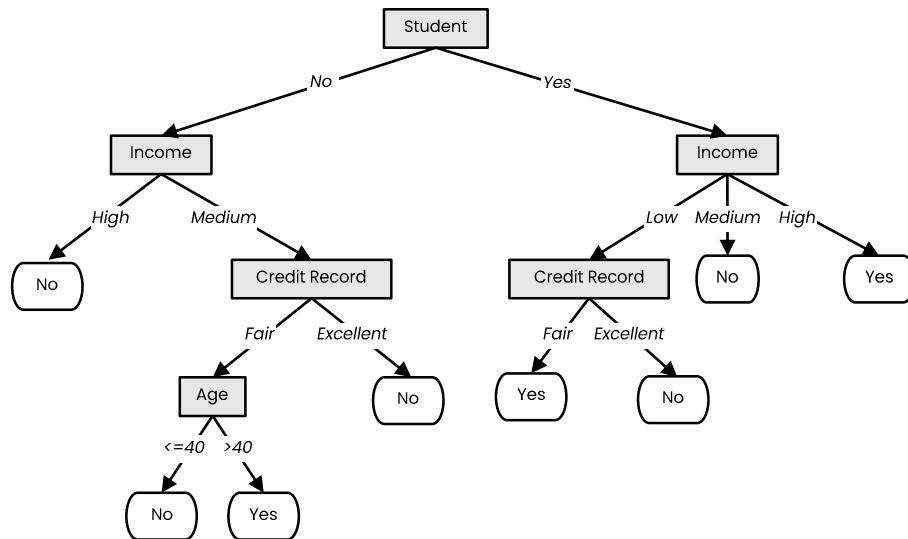
Post-pruning



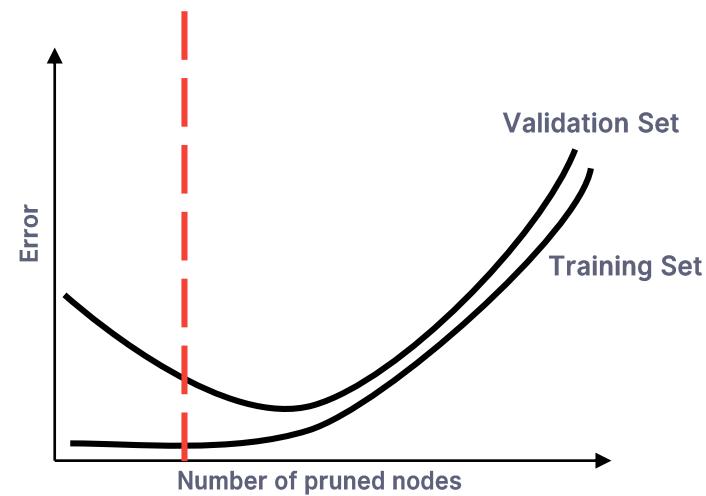
- Prune nodes in a bottom-up manner, if it decreases validation error.



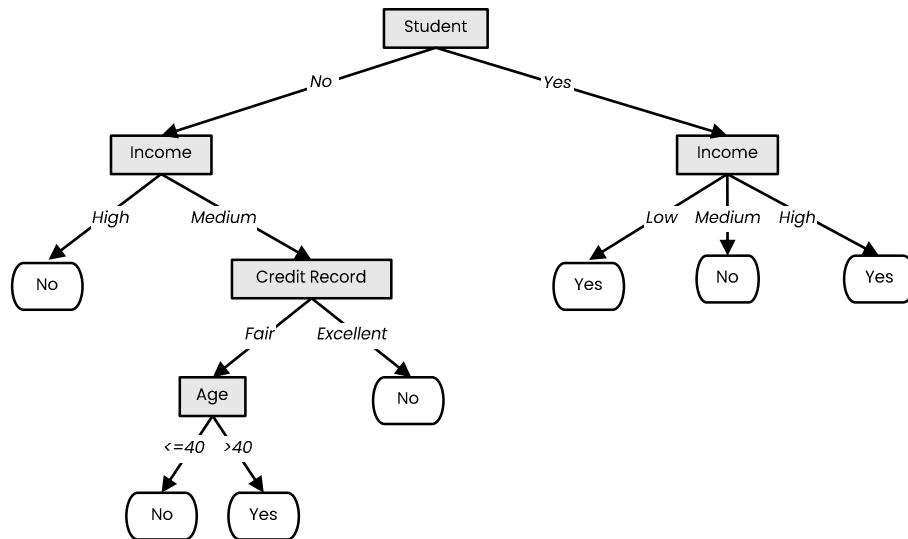
Post-pruning



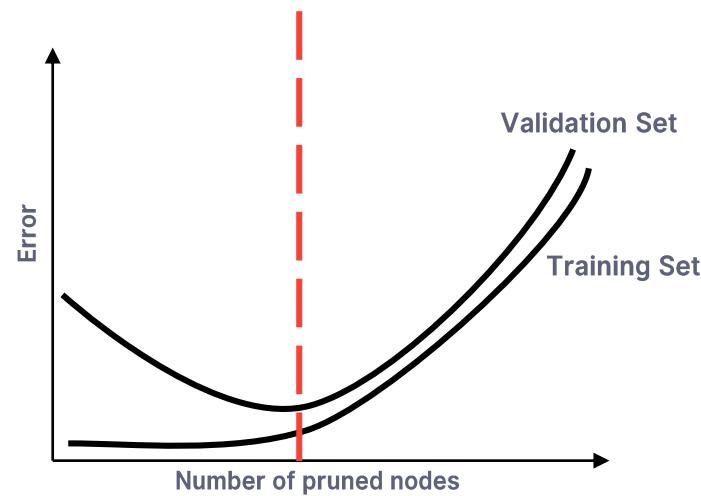
- Prune nodes in a bottom-up manner, if it decreases validation error.



Post-pruning



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



Handling Numerical Attributes

Handling numerical data

- How does the ID3 algorithm treat numerical attributes?

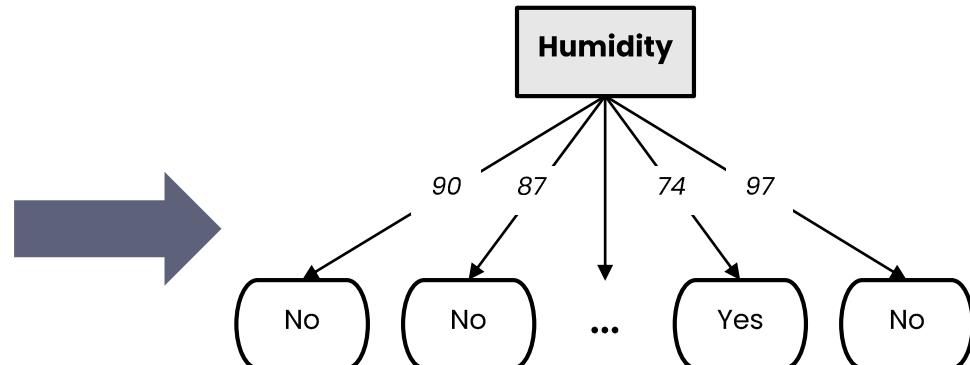
Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	90	Weak	No
Sunny	Hot	87	Strong	No
Overcast	Hot	93	Weak	Yes
Rainy	Mild	89	Weak	Yes
Rainy	Cool	79	Weak	Yes
Rainy	Cool	59	Strong	No
Overcast	Cool	77	Strong	Yes
Sunny	Mild	91	Weak	No
Sunny	Cool	68	Weak	Yes
Rainy	Mild	80	Weak	Yes
Sunny	Mild	72	Strong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

Now we have a numerical value for humidity

Handling numerical data

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

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



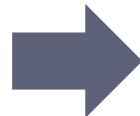
Now we have a numerical value for humidity

Handling numerical data

- Numerical attributes need 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

Handling numerical data

- Numerical attributes need 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

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

Sort

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 of numerical attribute a if it is split on value t .

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

Handling numerical data

- Numerical attributes need 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

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

gain(X , humidity, 83.5)

Handling numerical data

- Numerical attributes need 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

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

$$\text{gain}(X, \text{humidity}, 83.5) = 0.94$$

Handling numerical data

- Numerical attributes need 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

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

$$\text{gain}(X, \text{humidity}, 83.5) = 0.94 - \frac{7}{14} * 0.591$$

Handling numerical data

- Numerical attributes need 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

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

$$\text{gain}(X, \text{humidity}, 83.5) = 0.94 - \frac{7}{14} * 0.591 - \frac{7}{14} * 0.985$$

Handling numerical data

- Numerical attributes need 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

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

Sort

Mean of each consecutive pair

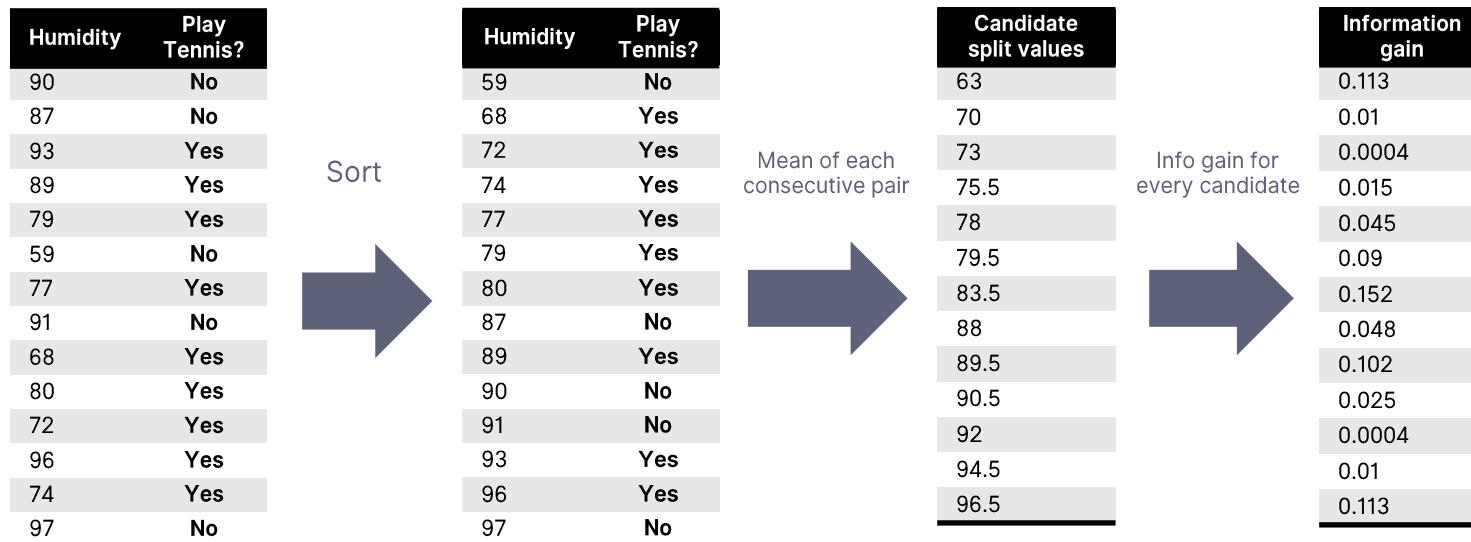
Candidate split values
63
70
73
75.5
78
79.5
83.5
88
89.5
90.5
92
94.5
96.5

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

$$\text{gain}(X, \text{humidity}, 83.5) = 0.94 - \frac{7}{14} * 0.591 - \frac{7}{14} * 0.985 = \\ \mathbf{0.152}$$

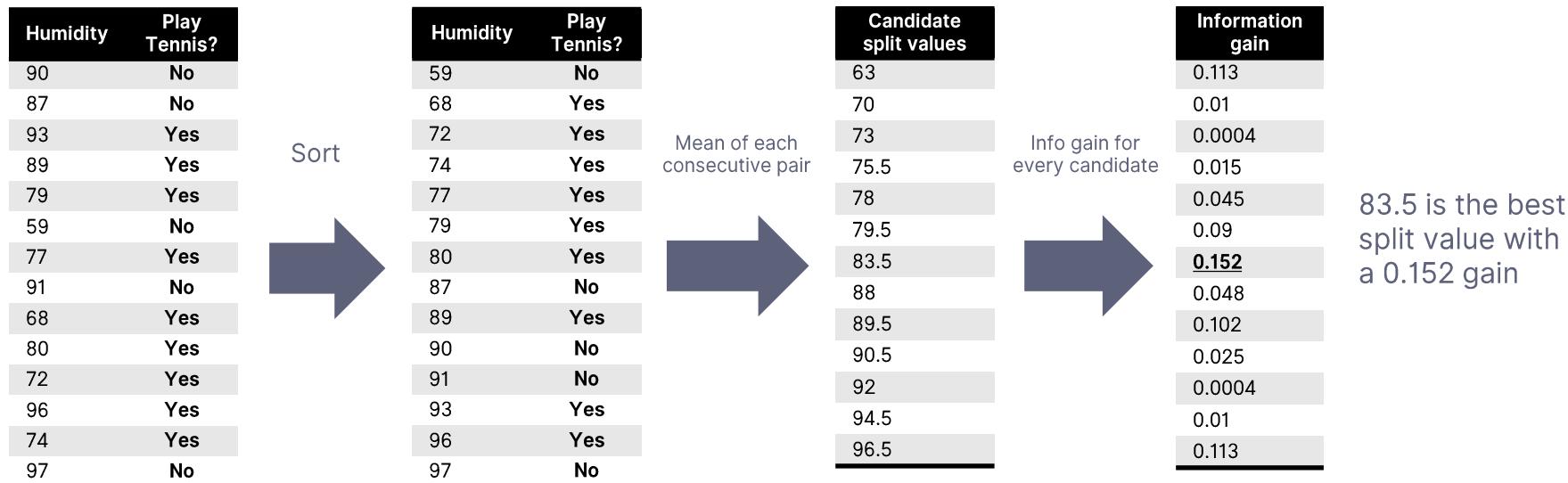
Handling numerical data

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



Handling numerical data

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



Handling numerical data

- Numerical attributes need 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 split value for *Humidity* with a 0.152 information gain.
- *Humidity* is now treated like a categorical attribute with two possible values.
- A new optimal split is computed at every level of the tree.
- Numerical attribute can be used more than once in the tree, with different split values

Handling Missing Attributes

Handling Missing Attributes 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

- A dataset might have instances where some attribute values are unknown.
- Simply ignoring them would mean that we throw a lot of information away.
- There are many better ways of handling missing values:

Handling Missing Attributes at Training Time

Does it fly?	Color	Class
No	<i>White</i>	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**

- A dataset might have instances where some attribute values are unknown.
- Simply ignoring them would mean that we throw a lot of information away.
- There are many better ways of handling missing values:
 - Set it as the most common value for that attribute.

Handling Missing Attributes at Training Time

Does it fly?	Color	Class
No	<i>White</i>	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird

- A dataset might have instances where some attribute values are unknown.
- Simply ignoring them would mean that we throw a lot of information away.
- There are many better ways of handling missing values:
 - Set it as the most common value for that attribute.
 - Set it as the most probable value given the label.

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

$$P(\text{No}|\text{Bird}) = 0.33$$

$$P(\text{White}|\text{Mammal}) = 1$$

$$P(\text{Brown}|\text{Mammal}) = 0$$

Handling Missing Attributes at Training Time

Does it fly?	Color	Class
No	<i>White</i>	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

- A dataset might have instances where some attribute values are unknown.
- Simply ignoring them would mean that we throw a lot of information away.
- There are many better ways of handling missing values:
 - Set it as the most common value for that attribute.
 - Set it as the most probable value given the label.
 - Add a new instance to the dataset for every possible attribute value.

Handling Missing Attributes 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

$$\text{entropy}(X_{\text{color=brown}}) = 0$$
$$\text{entropy}(X_{\text{color=white}}) = 1$$

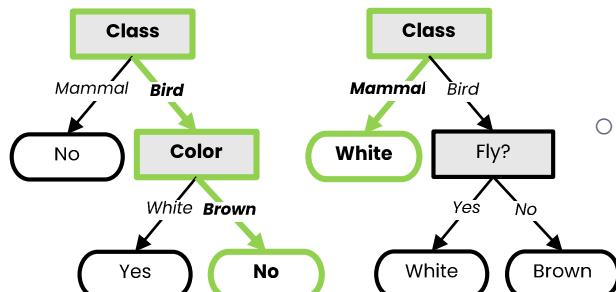
$$\text{gain}(X, \text{color}) = 0.985 - \frac{2}{6} * 0 - \frac{4}{6} * 1 = 0.318$$

- A dataset might have instances where some attribute values are unknown.
- Simply ignoring them would mean that we throw a lot of information away.
- There are many better ways of handling missing values:
 - Set it as the most common value for that attribute.
 - Set it as the most probable value given the label.
 - Add a new instance to the dataset for every possible attribute value.
 - Leave it unknown, but, when evaluating the gain for that attribute, consider only the instances where the attribute is defined.
 - If the attribute is chosen for splitting, send the instances with unknown values to all children.

Handling Missing Attributes at Training Time

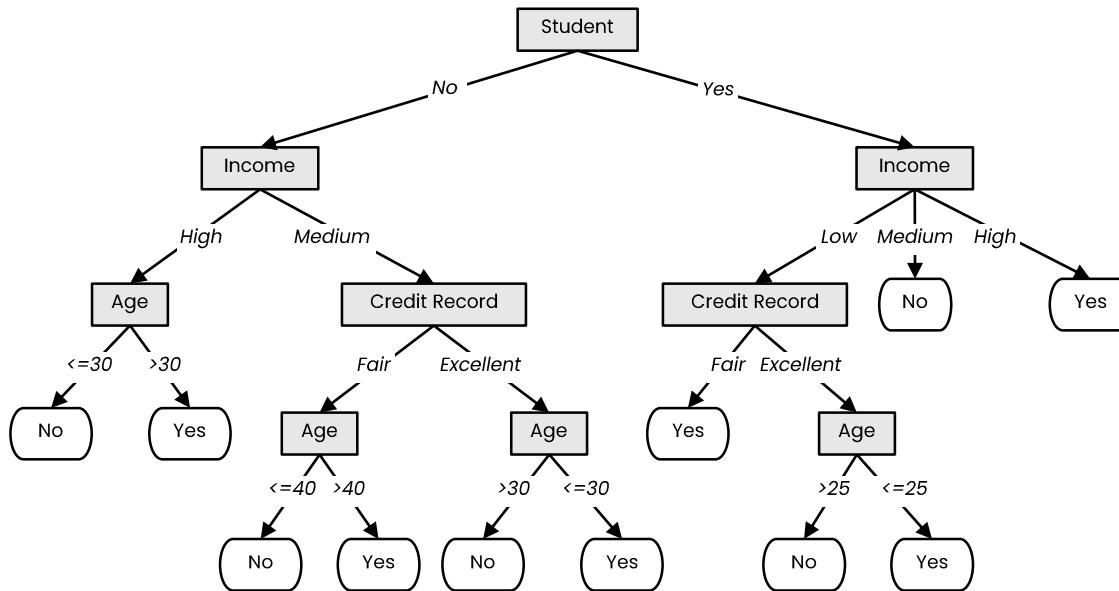
Does it fly?	Color	Class
No	White	Mammal
No	White	Mammal
No	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird

- A dataset might have instances where some attribute values are unknown.
- Simply ignoring them would mean that we throw a lot of information away.
- There are many better ways of handling missing values:
 - Set it as the most common value for that attribute.
 - Set it as the most probable value given the label.
 - Add a new instance to the dataset for every possible attribute value.
 - Leave it unknown, but, when evaluating the gain for that attribute, consider only the instances where the attribute is defined.
 - If the attribute is chosen for splitting, send the instances with unknown values to all children.
 - Grow a decision tree on all other attributes (including label) to predict the missing values.
 - Use instances where the attribute is defined as training data.



Handling Missing Attributes at Inference Time

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



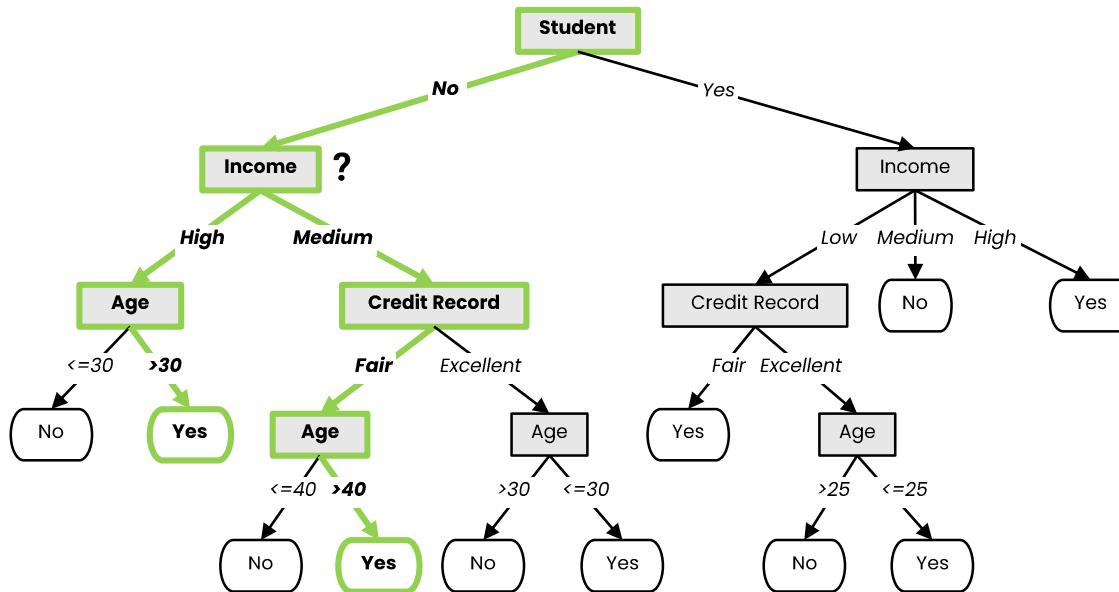
Loan default?



- Not a student
- 46 years old
- Unknown income*
- Fair credit record

Handling Missing Attributes at Inference Time

- When we encounter a node which checks an attribute with a missing value, we explore all possibilities.
- We go down all branches and make all leaf nodes take a (*weighted*) vote for the final prediction.



Loan default?



- Not a student
- 46 years old
- Unknown income*
- Fair credit record

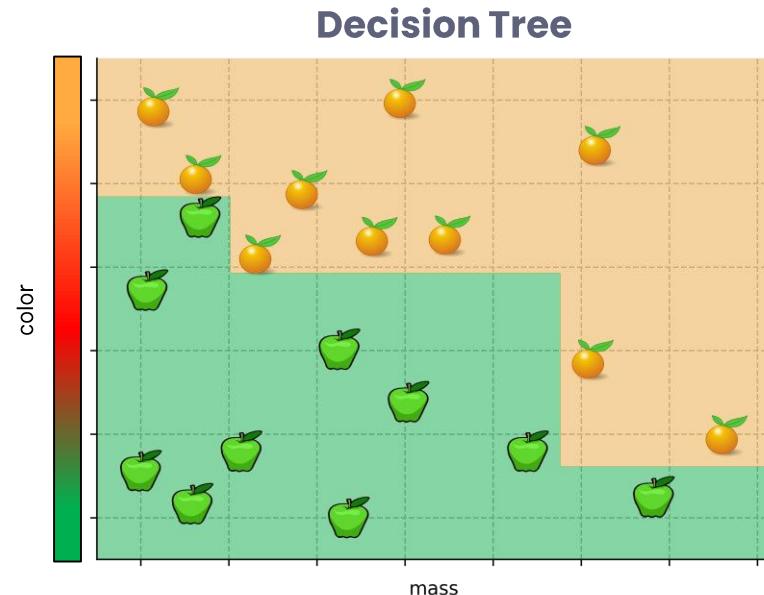
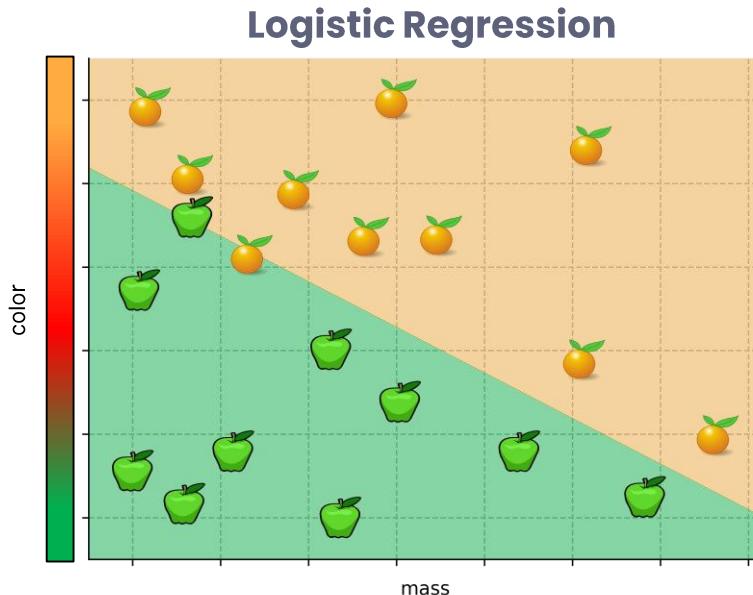
Yes

C4.5 Algorithm

- C4.5 is an extension to the ID3 algorithm which brings several improvements:
 - Ability to handle *both discrete and continuous attributes*. Continuous attributes are split by finding a best-splitting threshold.
 - Ability to handle **missing attributes** both at training and inference time. Training missing values are not used in information gain computation. Inference missing values are handled by considering all subsequent branches.
 - Ability to handle **attributes with different costs**.
 - Post-pruning in a bottom-up manner for removing branches which decrease validation error.

Decision Boundaries

- Different algorithms produce different decision boundaries.



History of Decision Trees

- “*Automatic Interaction Detection (AID)*”
 - The first regression tree algorithm.

Morgan and Sonquist, 1963
- “*Theta Automatic Interaction Detection (THAID)*”
 - The first classification tree algorithm.

Messenger and Mandell, 1972
- “*Classification and regression trees (CART)*”
 - Introduced the ID3 algorithm.

Breiman L., J. Friedman, R. Olshen, and C. Stone, 1984
- “*Induction of Decision Trees*”
 - Introduced the C4.5 algorithm.

J.R. Quinlan, 1986
- “*C4.5: Programs for Machine Learning*”
 - Introduced the C4.5 algorithm.

J. R. Quinlan, 1993

Summary

- A **decision tree** is a tool which uses a tree-like graph of decisions and their possible outcomes.
- **Decision tree learning** is a machine learning method which uses a *decision tree* as a predictive model.
- The **ID3** algorithm builds a decision tree by iteratively splitting the data on the values of the attribute with the largest **information gain** (decrease in **entropy**)
 - Using the decrease in **gini impurity** is also an option very common in practice.
- **C4.5** is an extension to *ID3* which handles *attributes with continuous values*, handles *missing attribute values* and adds regularization by *pruning* unnecessary branches.

Random Forests

Ensemble learning with decision trees

Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

Random Forests

- **Random Forests** basic idea:
 - Instead of *growing* a single decision tree and use it to make predictions, grow many slightly different trees and combine their predictions.

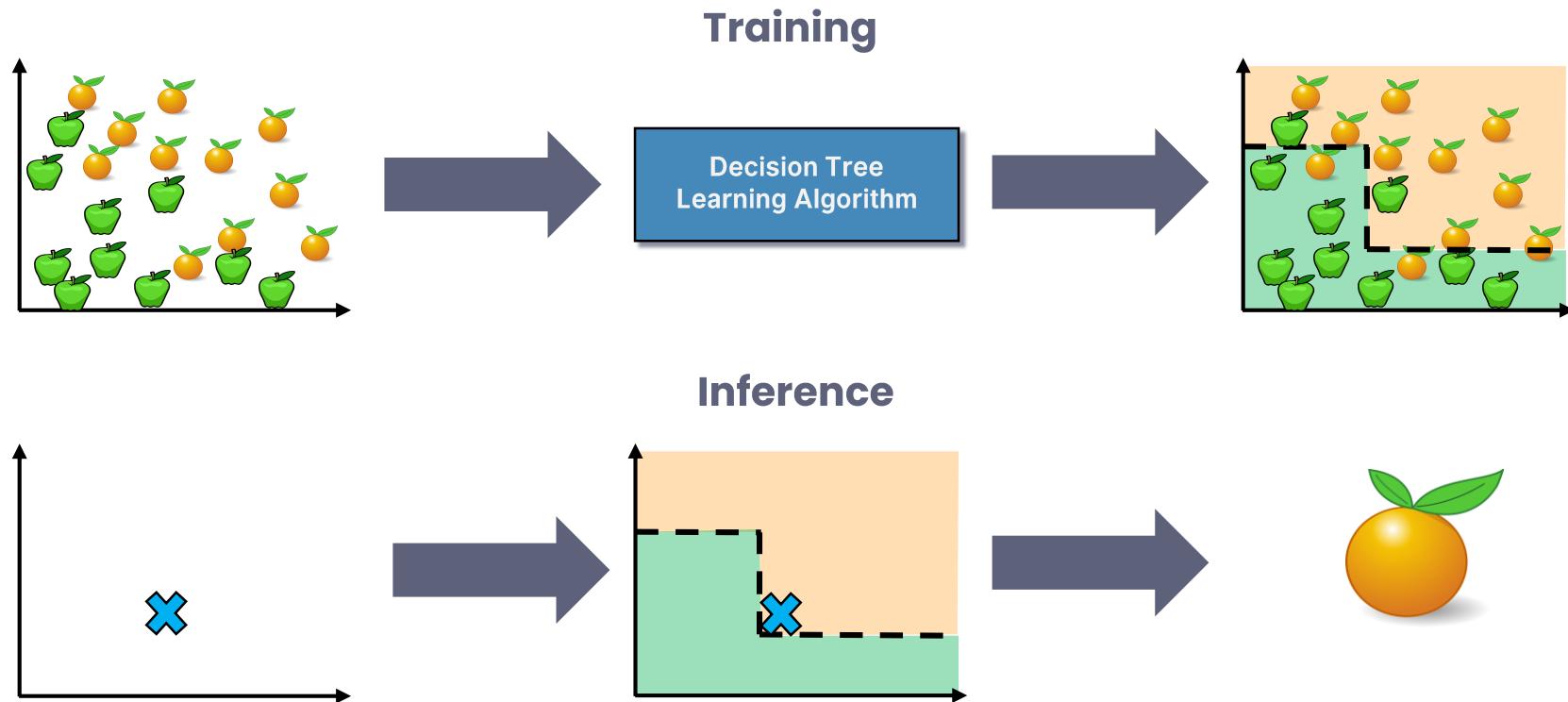
Random Forests

- **Random Forests** basic idea:
 - Instead of *growing* a single decision tree and use it to make predictions, grow many slightly different trees and combine their predictions.
- We have one dataset, so how do we obtain *slightly different* trees?

Random Forests

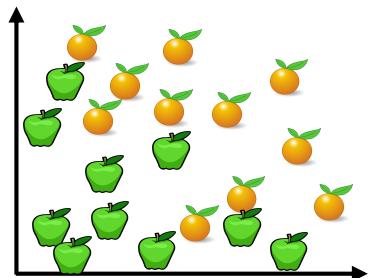
- **Random Forests** basic idea:
 - Instead of *growing* a single decision tree and use it to make predictions, grow many slightly different trees and combine their predictions.
- We have one dataset, so how do we obtain *slightly different* trees?
 - **Bagging (Bootstrap Aggregating)**
 - Take random subsamples of data points from the large dataset to create N smaller datasets
 - Fit a different model on each of small datasets.
 - **Random Subspace Method** (a.k.a. Feature Bagging)
 - Fit N different models, but allow each one to see a random subsample of features from the original dataset.

Reminder – Single Decision Tree Flow

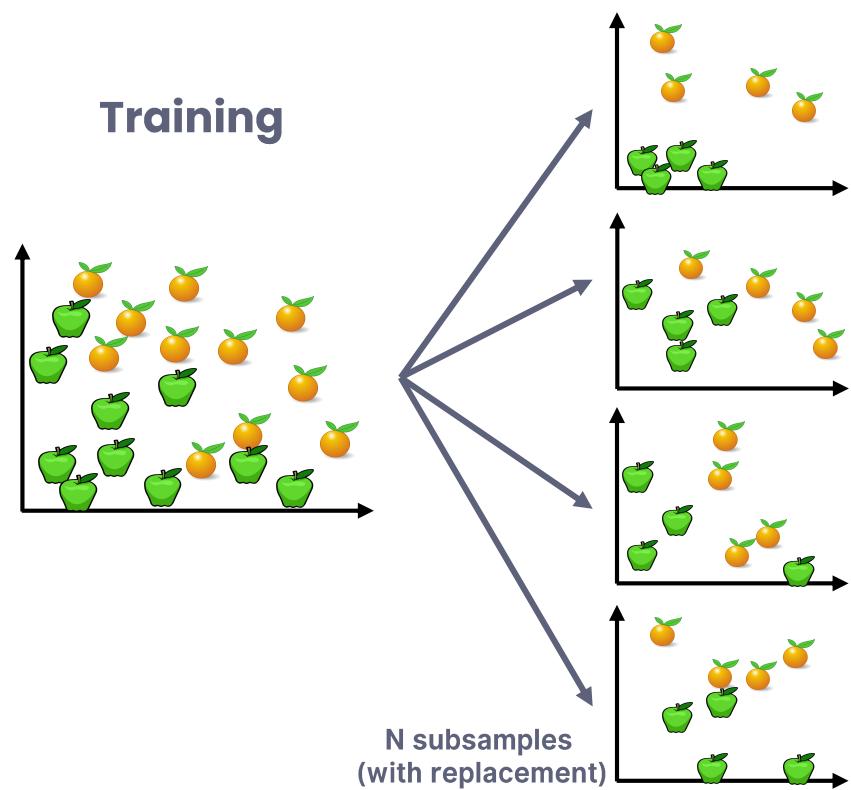


Bagging

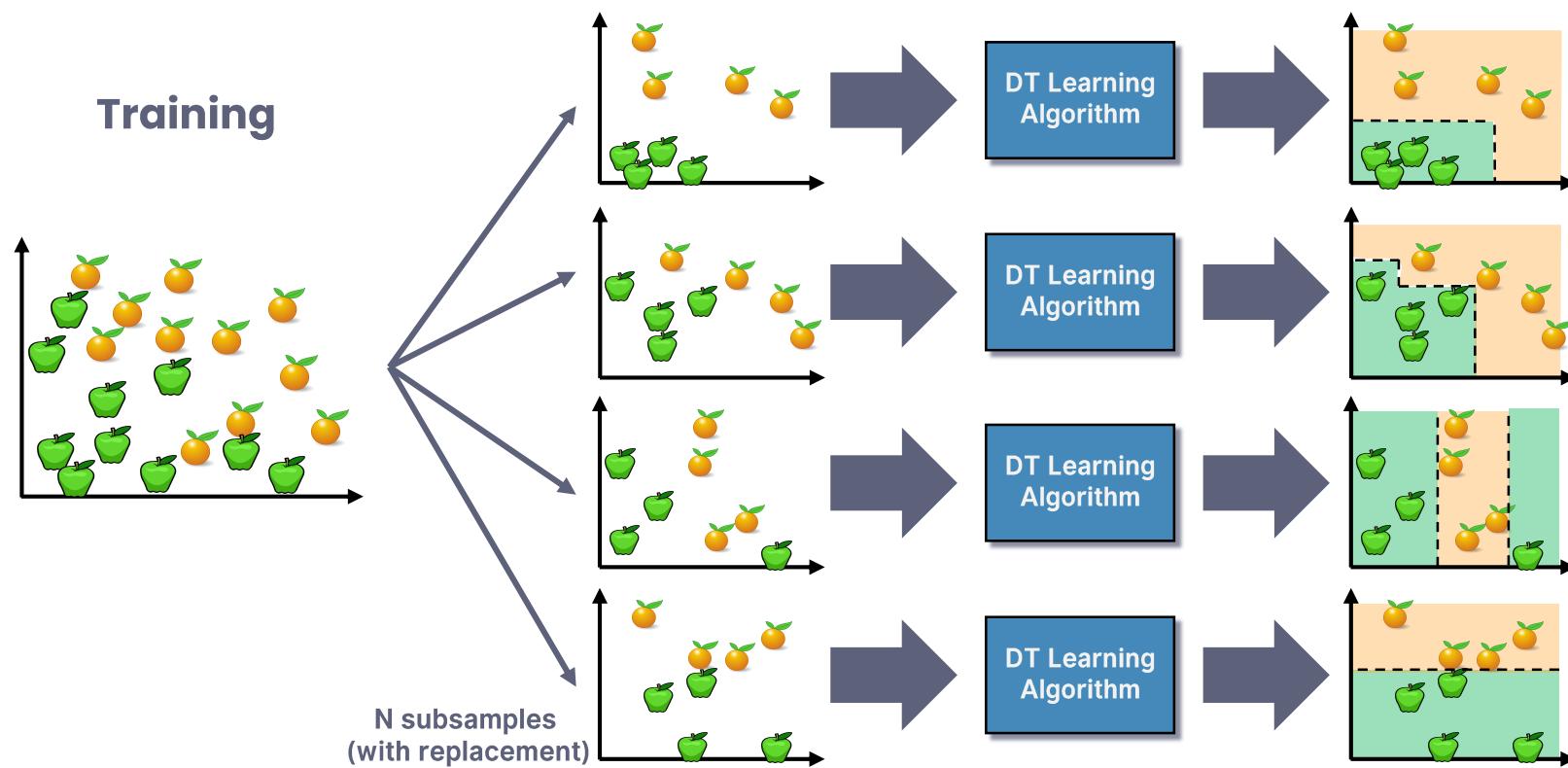
Training



Bagging

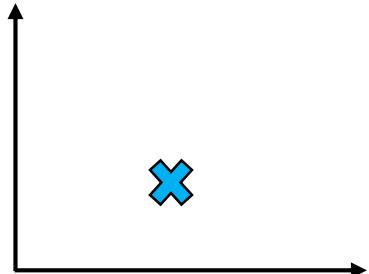


Bagging



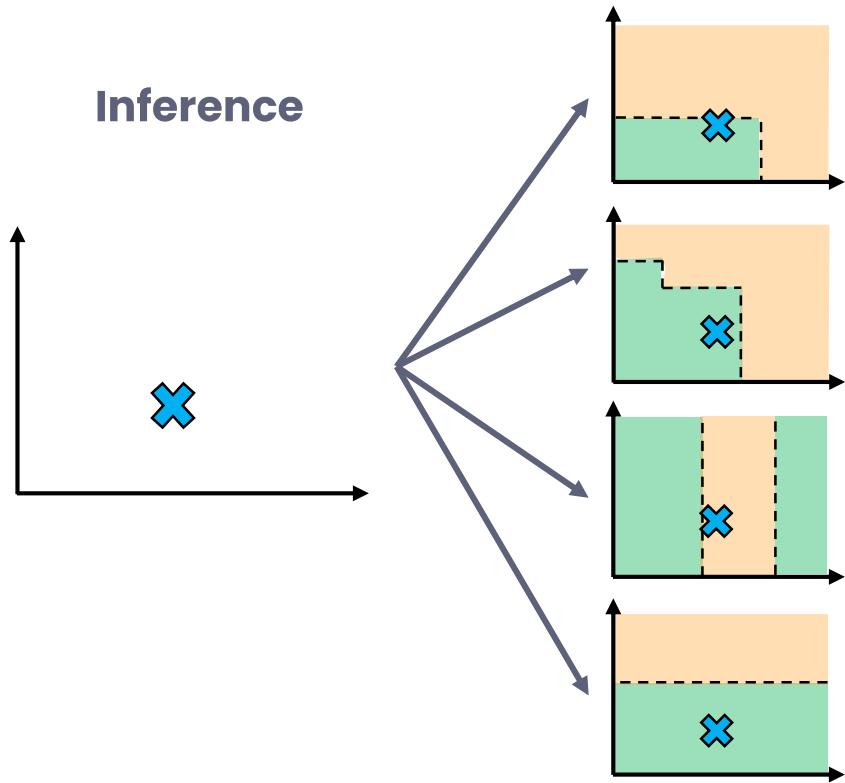
Bagging

Inference



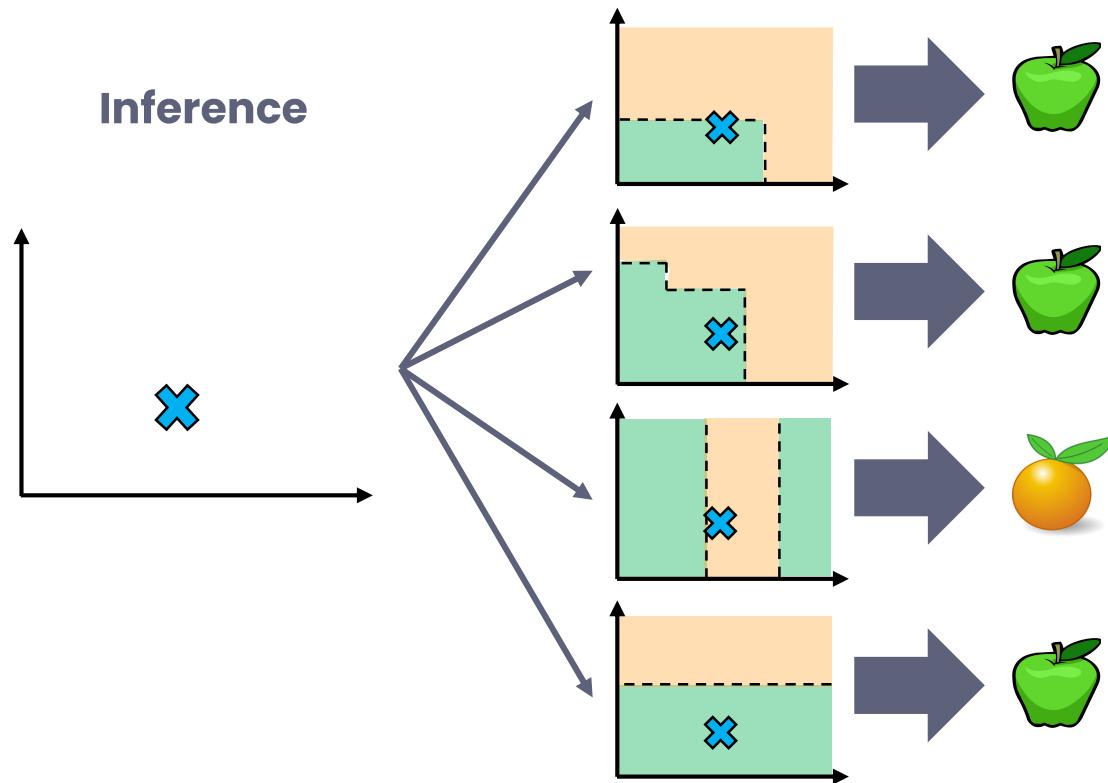
Bagging

Inference



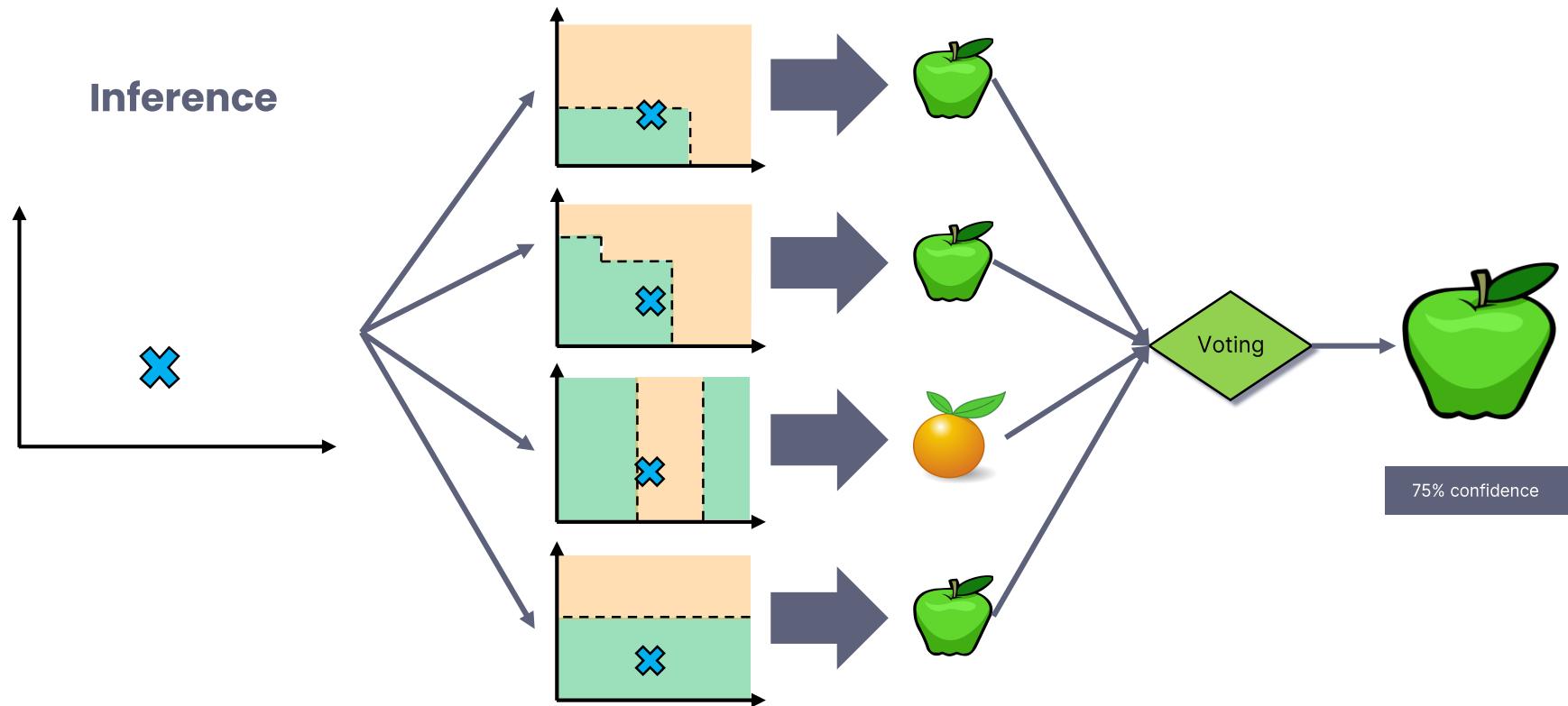
Bagging

Inference



Bagging

Inference



Random Subspace Method

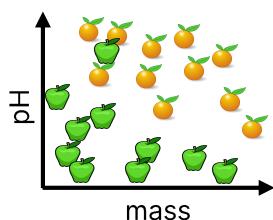
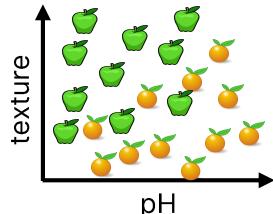
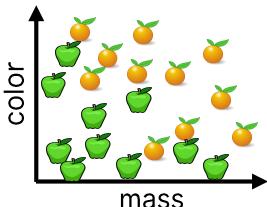
Training

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

Training

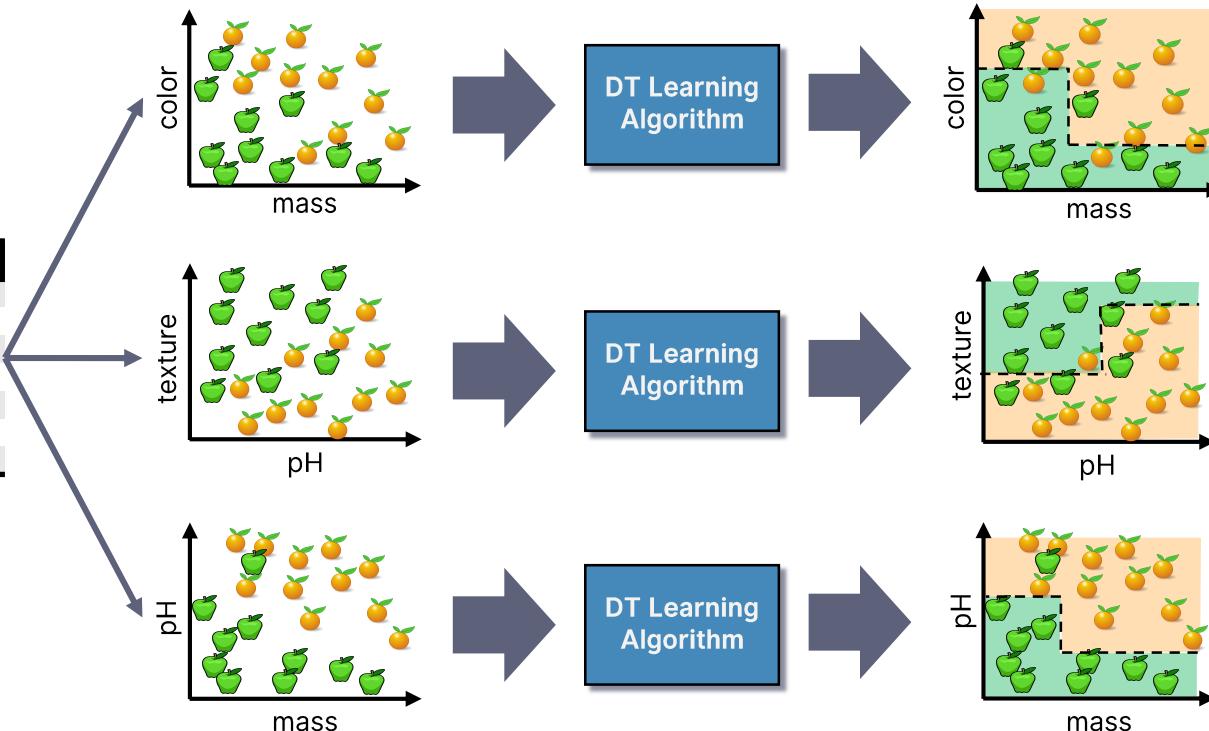
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	Apple
121	Orange	Rough	3.9	Orange
85	Red	Smooth	3.3	Apple
101	Orange	Smooth	3.7	Orange
111	Green	Rough	3.5	Apple
...				
117	Red	Rough	3.4	Orange



Random Subspace Method

Training

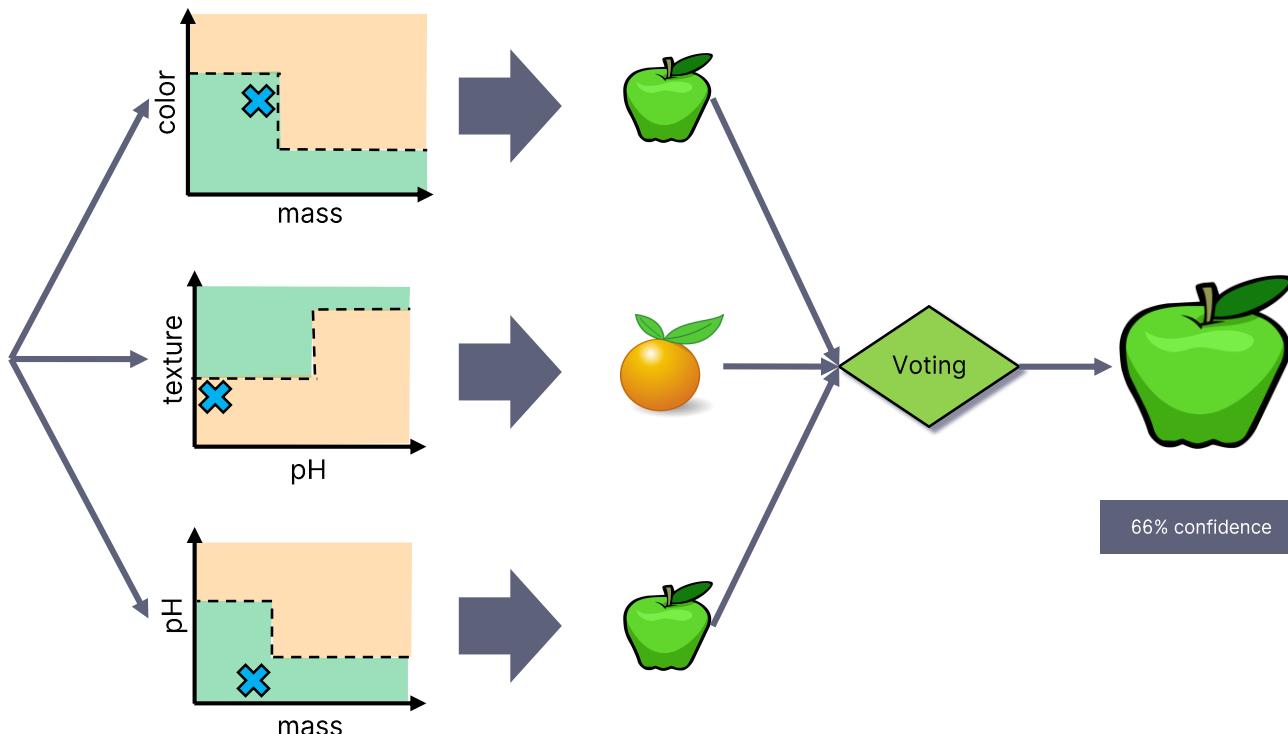
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	Apple
121	Orange	Rough	3.9	Orange
85	Red	Smooth	3.3	Apple
101	Orange	Smooth	3.7	Orange
111	Green	Rough	3.5	Apple
...				
117	Red	Rough	3.4	Orange



Random Subspace Method

Inference

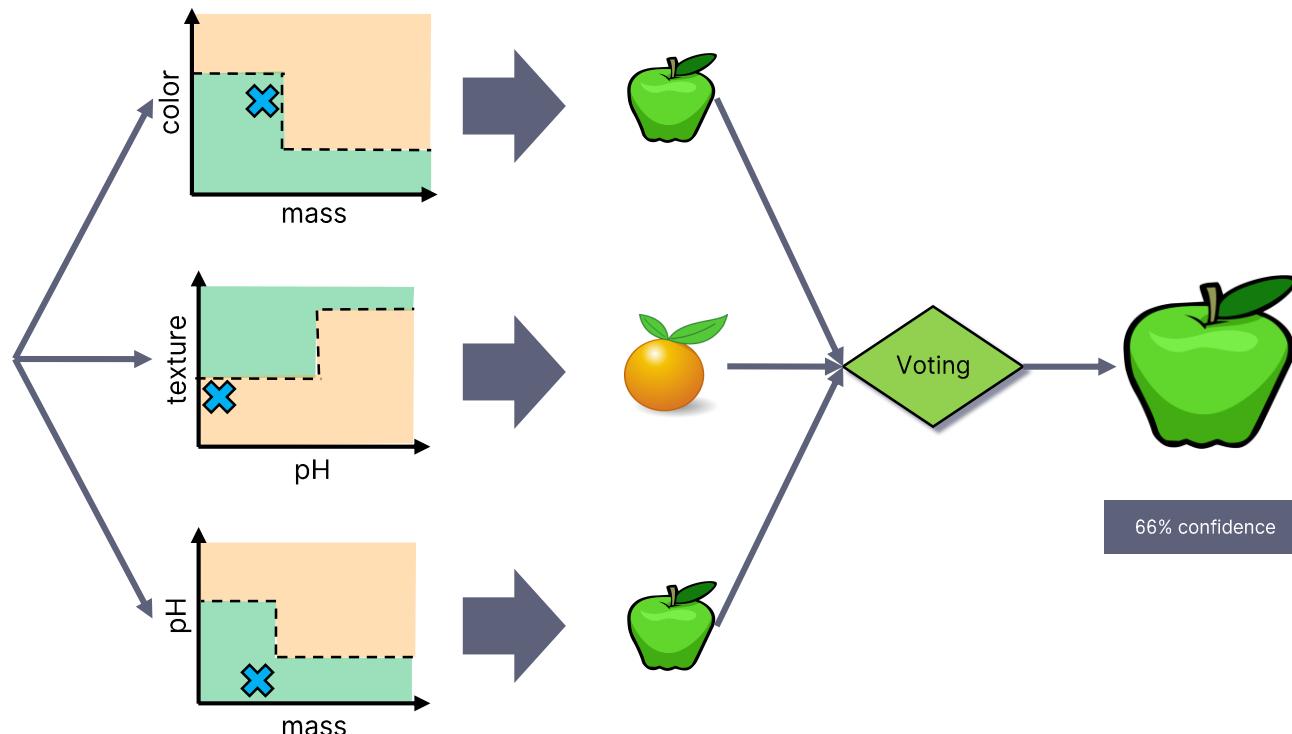
87 Red Smooth 3.1



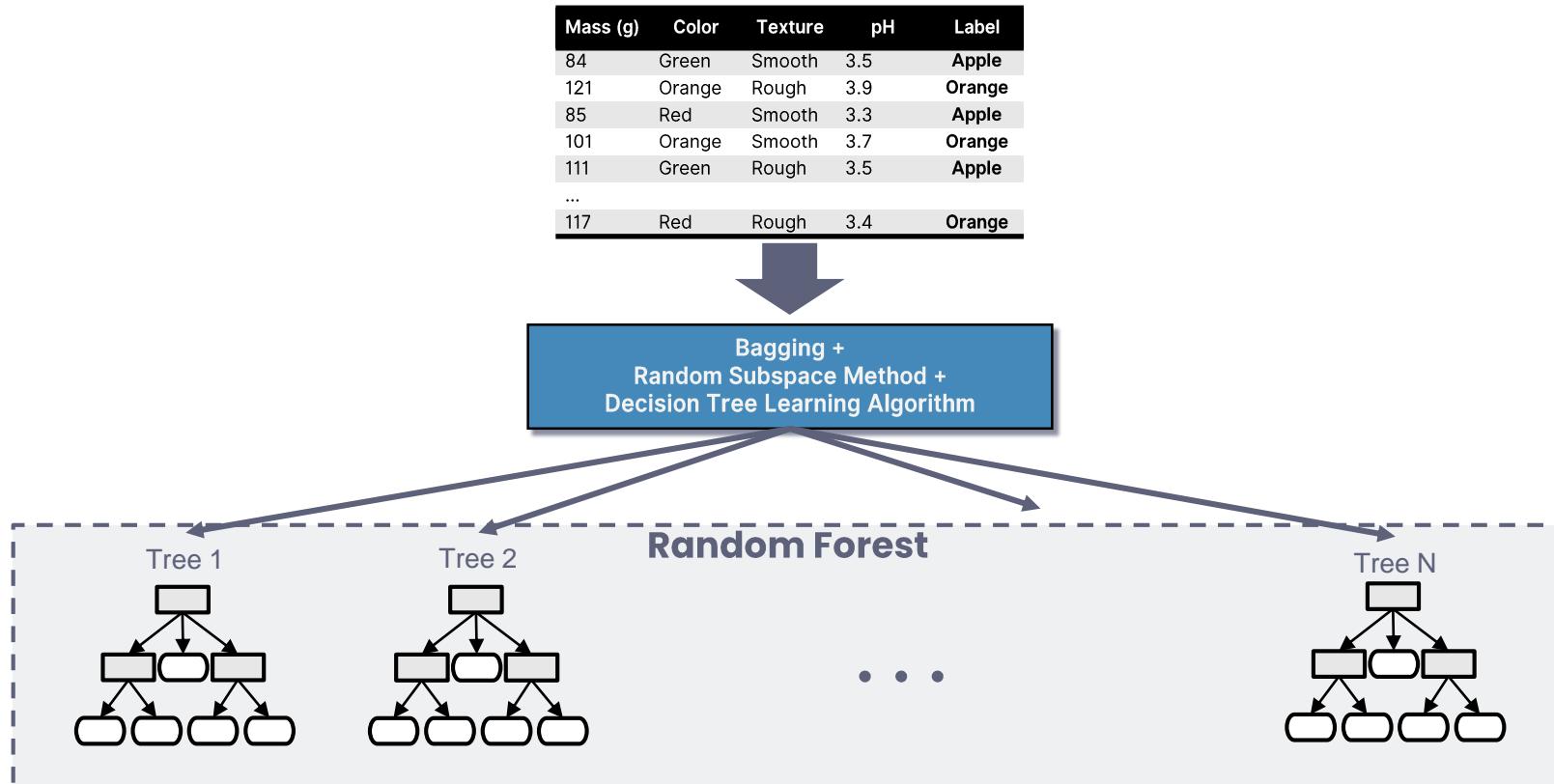
Random Subspace Method

Inference

87 Red Smooth 3.1



Random Forests



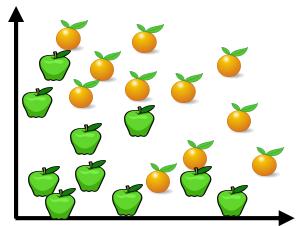
History of Random Forests

- “*Random Decision Forests*” and “*The Random Subspace Method for Constructing Decision Forests*”
 - Introduced the *Random Subspace Method* Tin Kam Ho, 1995 and 1998
- “*Random Forests*”
 - Combined the Random subspace method with *Bagging* Leo Breiman, 2001
 - Introduced the term *random forest™*, which is a trademark of Leo Breiman and Adele Cutler

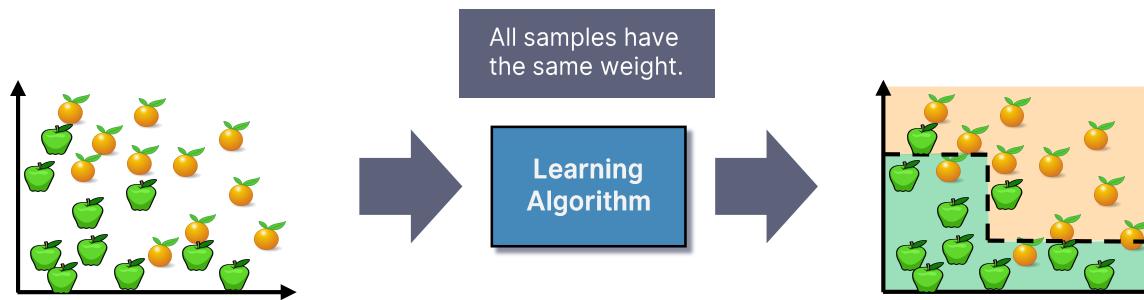
Ensemble Learning

- **Ensemble Learning** is a generic term for methods which combine *multiple learning algorithms* to obtain better performance than could be obtained from any constituent algorithm alone.
- *Random Forests* are one of the most common instances of ensemble learning.
- Common ensemble techniques:
 - **Bagging** – multiple models on random subsamples of data
 - **Random Subspace Method** – multiple models on random subsamples of features
 - **Boosting** – iteratively train models, while making the current model focus more on the mistakes of previous ones by increasing the weight of misclassified samples.

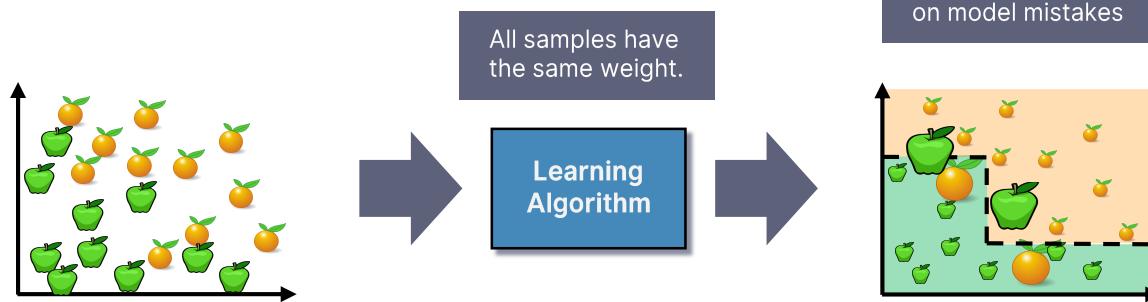
Boosting



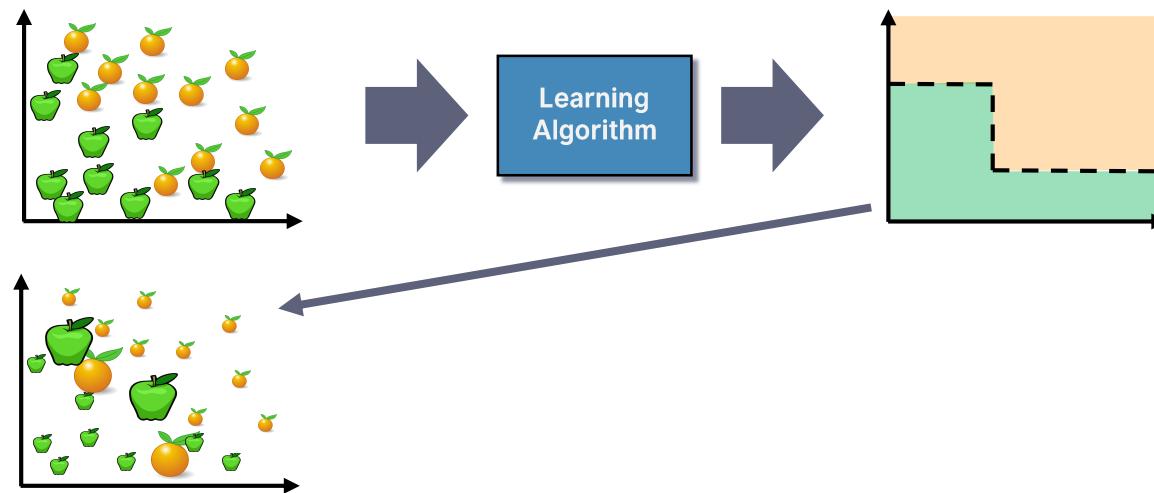
Boosting



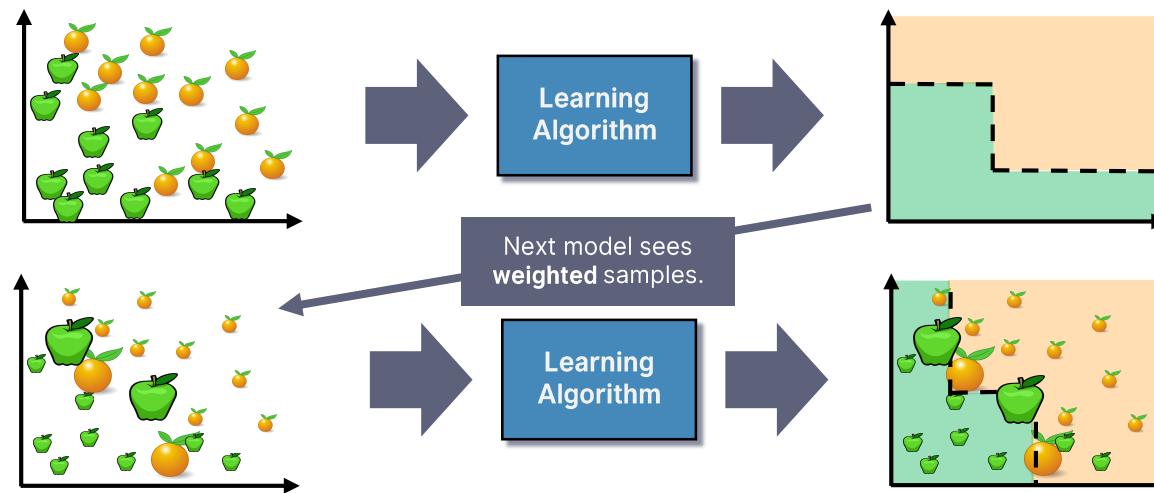
Boosting



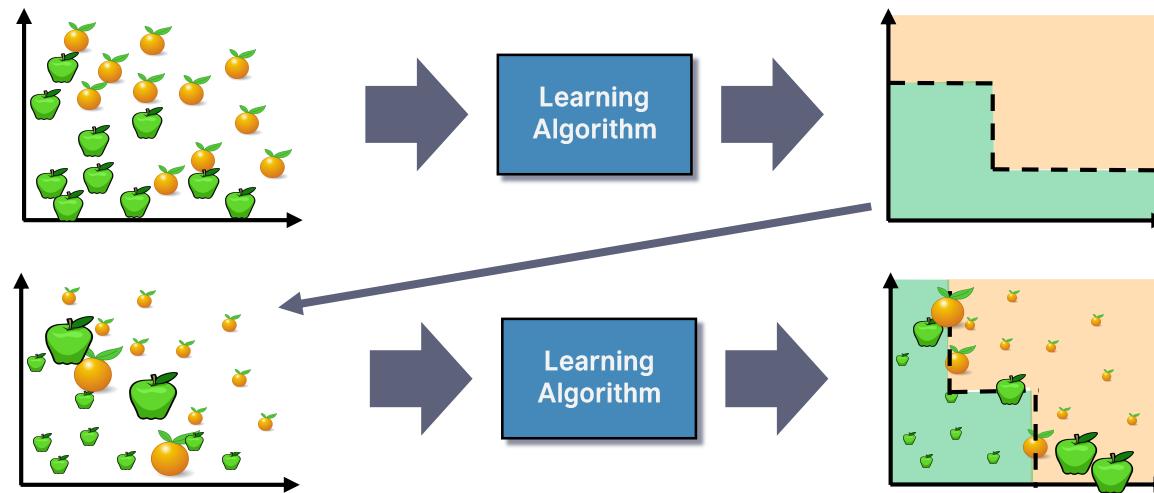
Boosting



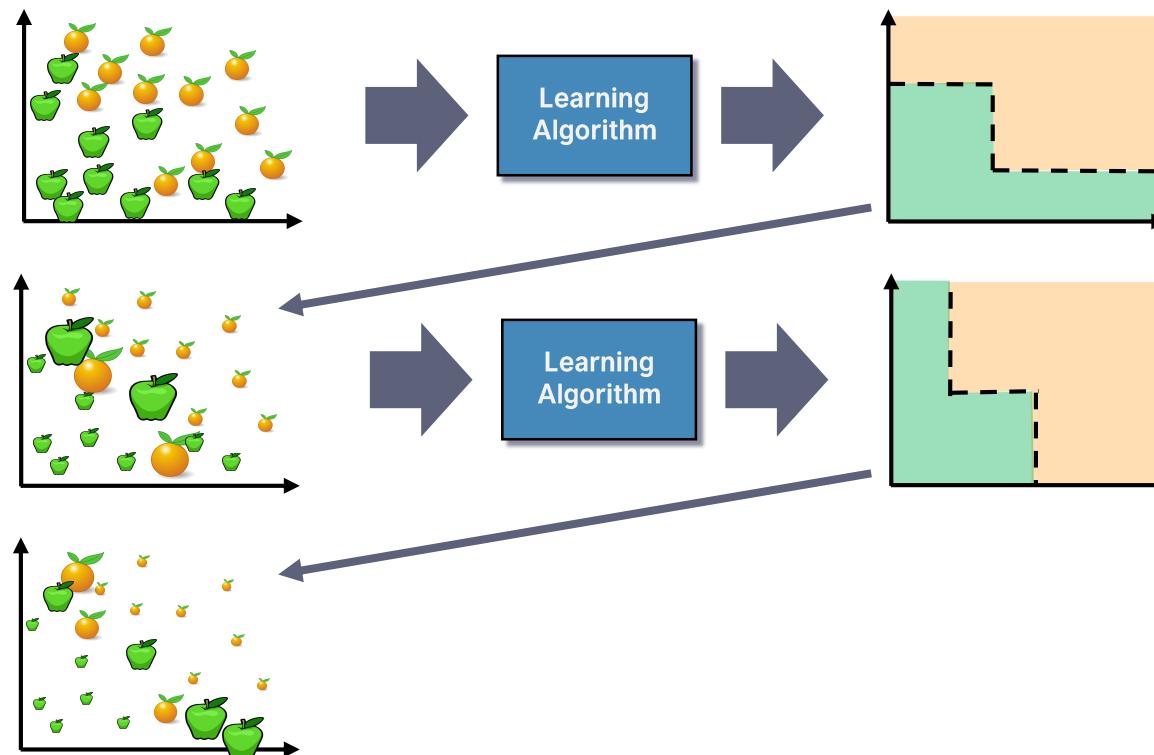
Boosting



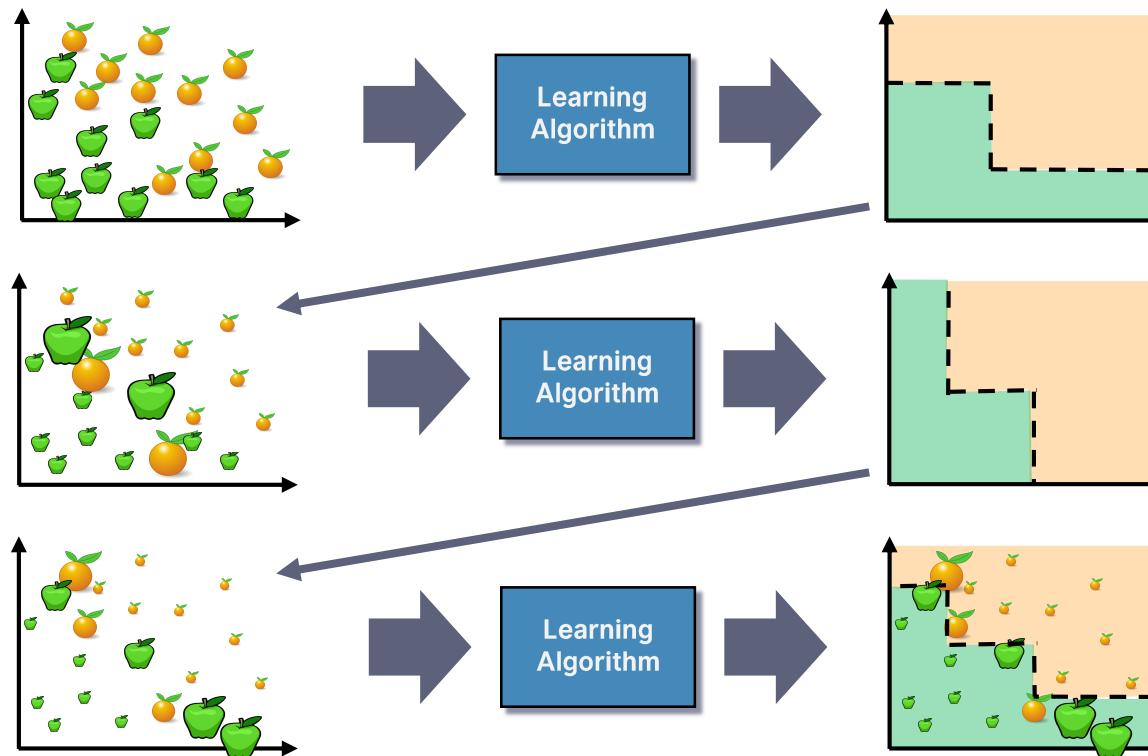
Boosting



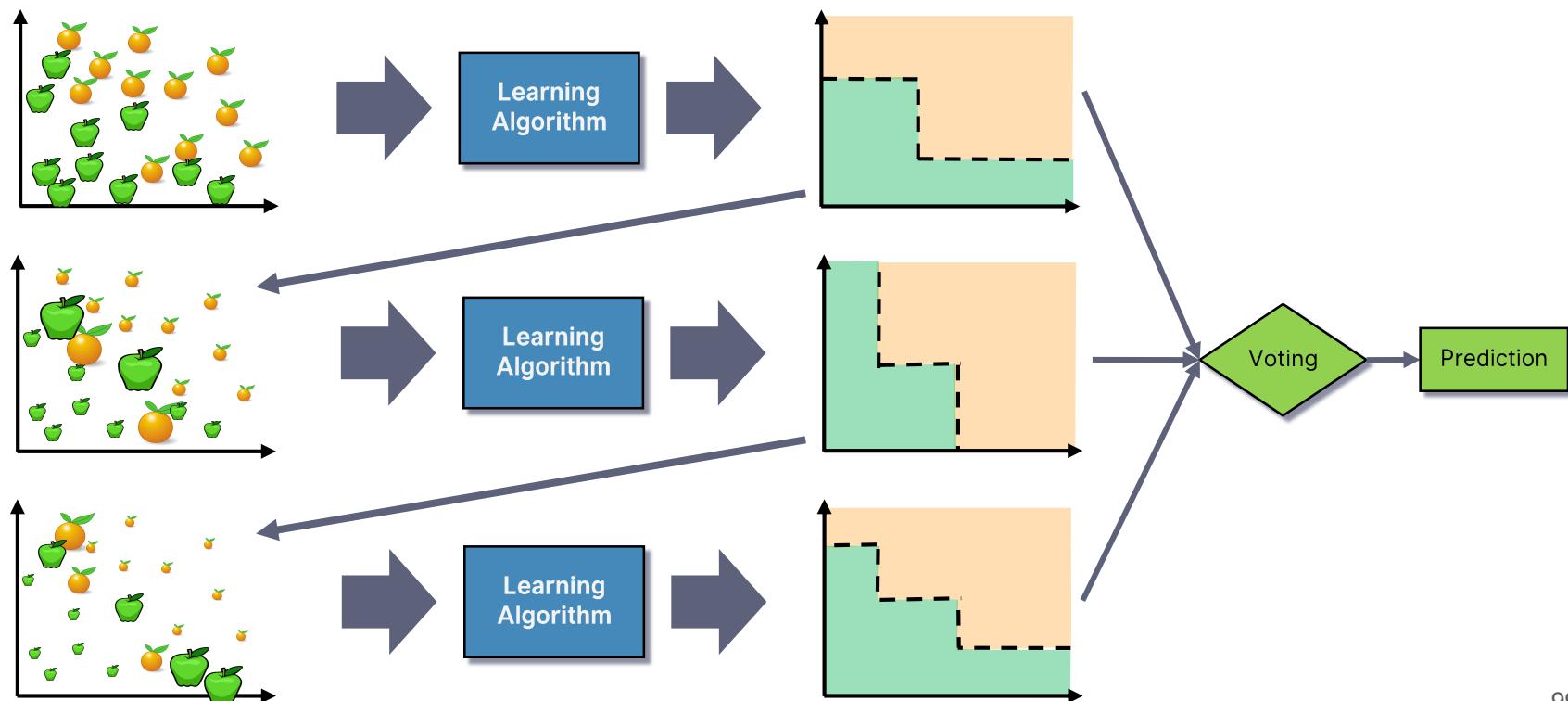
Boosting



Boosting



Boosting



Summary

- **Ensemble learning** methods combine multiple learning algorithms to obtain better performance that could be obtained from any of the constituent algorithms alone.
- Common ensemble techniques include **bagging** (obtaining multiple models from *random subsamples of data*), the **random subspace method** (multiple models on *random subsamples of features*) and **boosting** (iteratively train models, while making the current model *focus more on the mistakes of previous ones* by increasing the weight of misclassified samples)
- **Random Forests** are an ensemble learning method which uses a *decision tree learning* algorithm to obtain multiple trees by *bagging* and the *random subspace method*. They are used to rectify decision trees' habit of overfitting.

Decision Trees and Random Forests in Python

```
1  from sklearn.tree import DecisionTreeClassifier
2
3
4  clf = DecisionTreeClassifier(criterion = "entropy", min_samples_leaf = 3)
5      # lots of parameters: criterion = "gini" / "entropy"; max_depth; min_impurity_split; ...
6  clf.fit(X, y) # Note: It can only handle numerical attributes!
7      # Categorical attributes need to be encoded, see LabelEncoder and OneHotEncoder
8
9  clf.predict([x]) # predict class for x
10
11 clf.feature_importances_ # importance of each feature
12 clf.tree_ # the underlying tree object
13
14 clf = RandomForestClassifier(n_estimators = 20) # random forest with 20 trees
```

Keywords

Decision Tree

ID3

Entropy

Information Gain

Gini Impurity

C4.5

Pruning

Random Forests

Ensemble Learning

Bagging

Random Subspace Method

Boosting