UNIVERSITY OF ABERDEEN

CELEBRATING 525 YEARS 1495 – 2020

ABERDEEN 2040

Decision Trees in Practice

Data Mining & Visualisation

Lecture 12

2025

Today...

- Pruning
- Handling numerical attributes
- Handling missing values

Decision Tree Learning

As we discussed in the previous lecture, the process of training a full decision tree is iterative.

- For a given node (i.e. a parent), we calculate the information gain for every possible attribute.
- We select the attribute with the highest information gain to split on.
- For each child node of the parent, we iteratively repeat this process, until every child node ends up as a leaf (where a label is picked) **OR** until a point that we decide to stop training.

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.

This is particularly useful; **Decision trees are highly prone to overfitting!**

Pruning

There are two main methods of pruning:

- Pre-pruning: Where we stop the algorithm from continuing to build the tree before it fully classifies the data.

- Post-Pruning: Where we allow the algorithm to fully build the tree, before then replacing some non-leaf nodes with leaf nodes if it improves the validation error.

Pre-Pruning

Pre-pruning involves stopping the training process early:

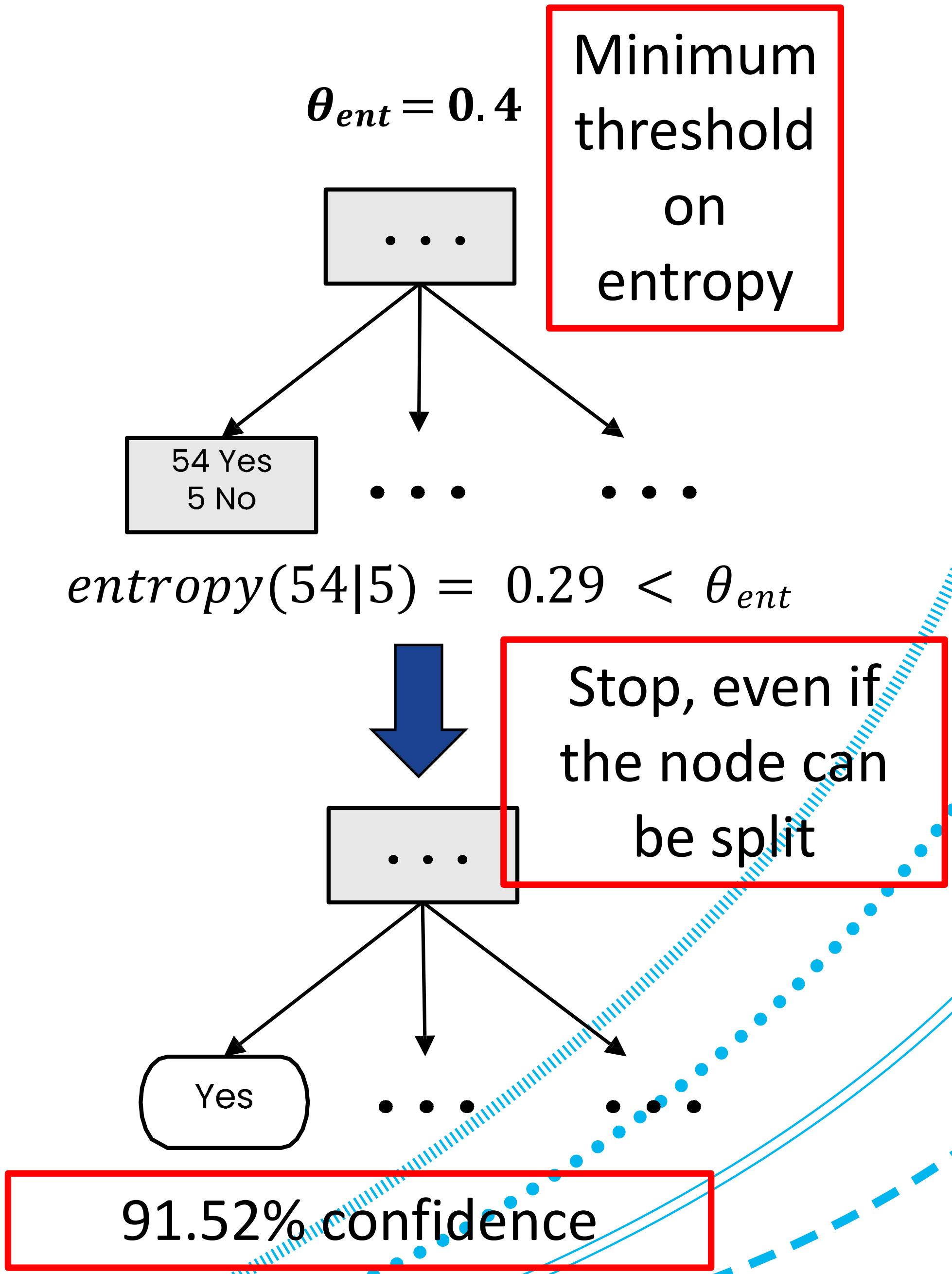
- If some condition is met, then the current node will not be split (even if the node is not 100% pure).
- That node will then become a leaf node, with the label of the majority class within the current set.

When turning impure nodes into leaf nodes, we can even use the class distribution as a prediction confidence value.

Pre-Pruning

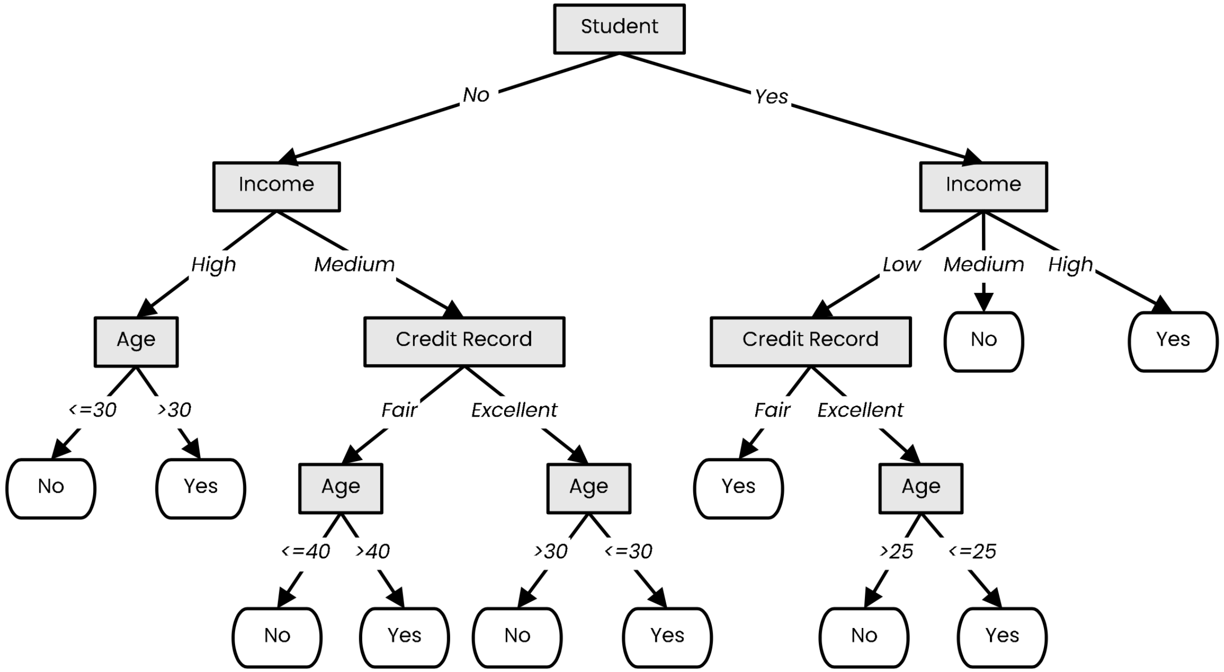
Common pre-pruning stopping criteria include setting a threshold on:

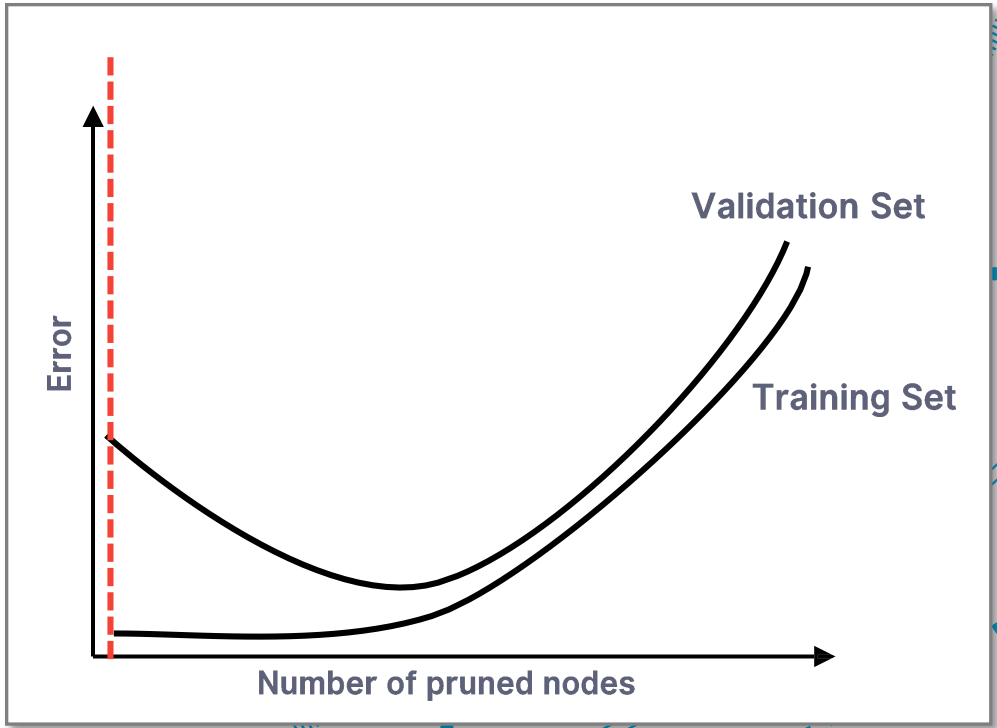
- The entropy of the current set
- The number of samples in the current set
- The information gain of the best attribute
- The depth of the tree



Post-Pruning

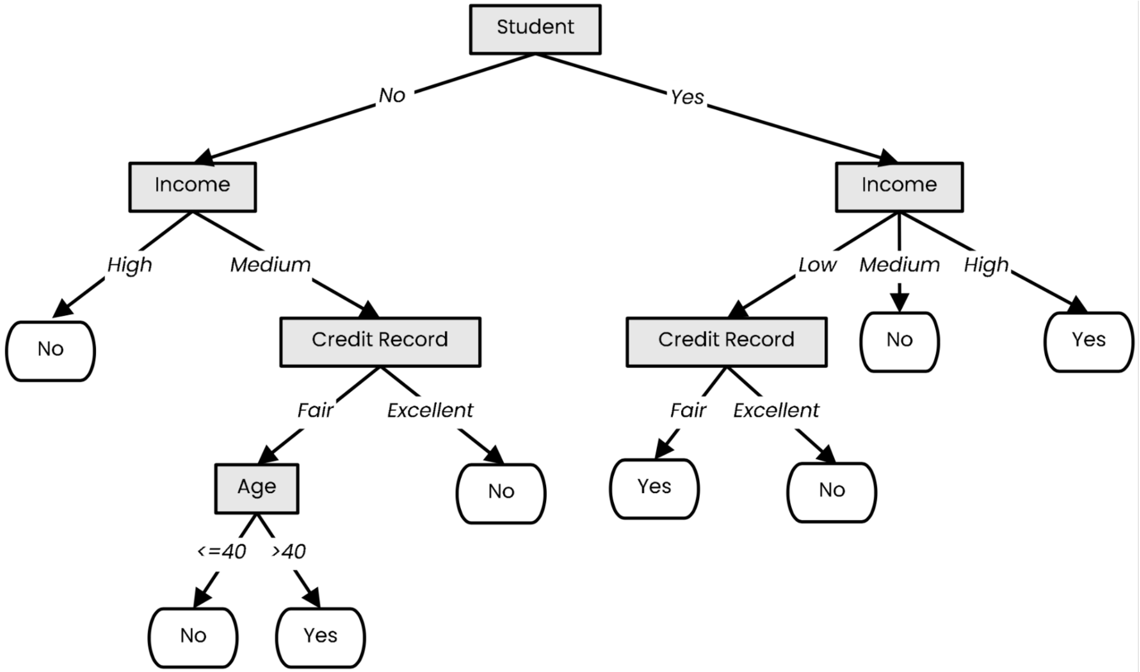
In **Post-Pruning**, we prune nodes in a bottom-up manner, if it decreases the validation error.

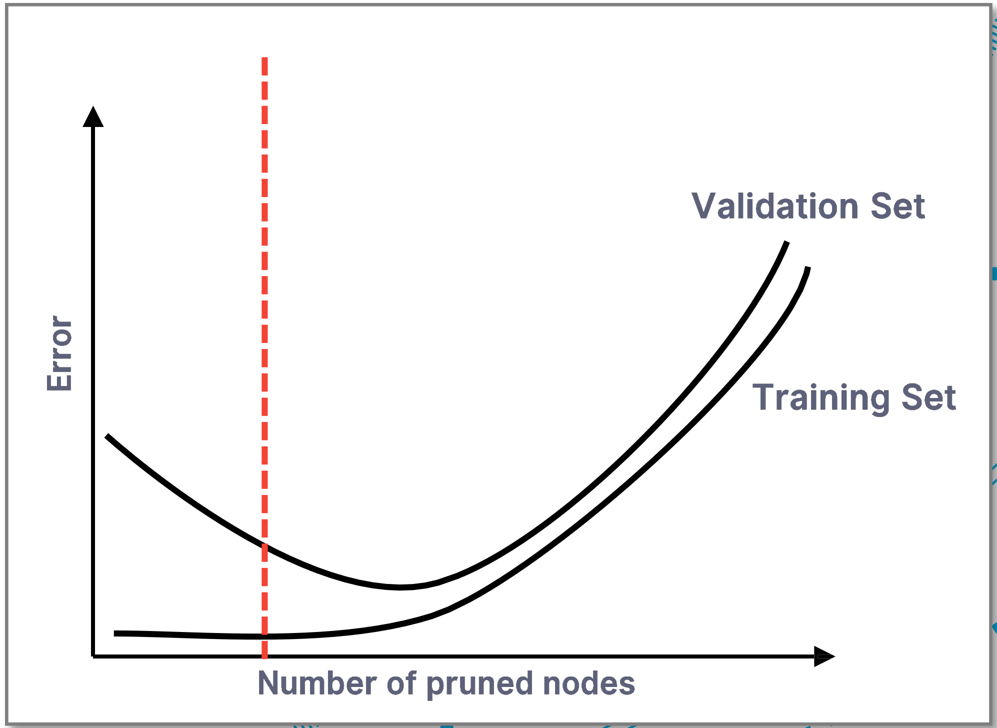




Post-Pruning

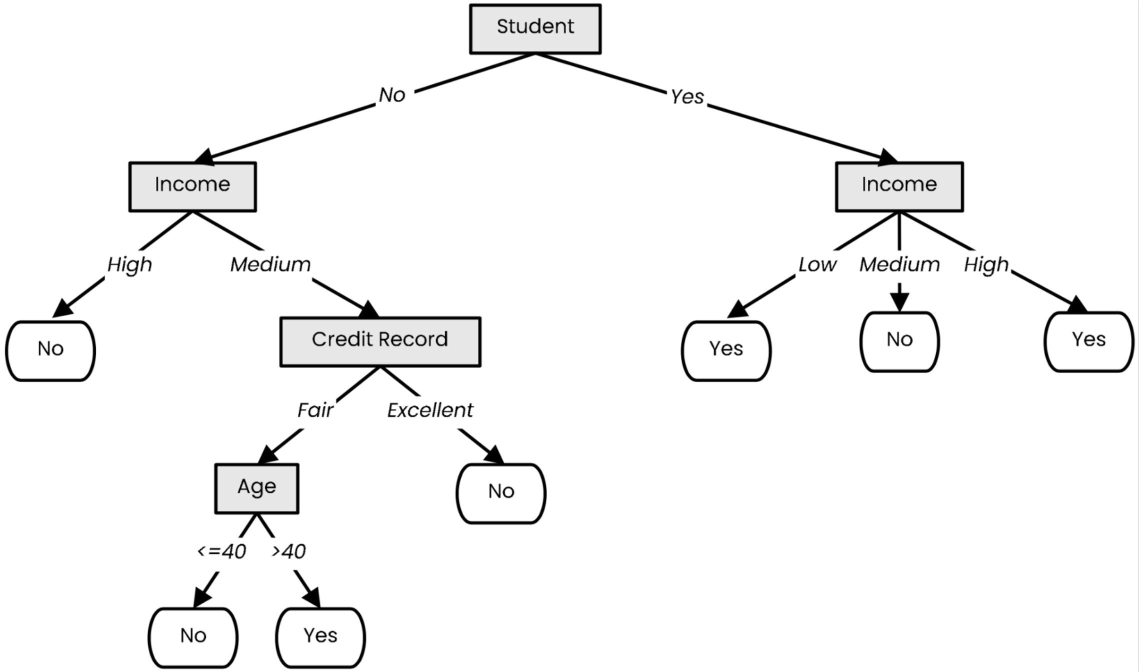
In **Post-Pruning**, we prune nodes in a bottom-up manner, if it decreases the validation error.

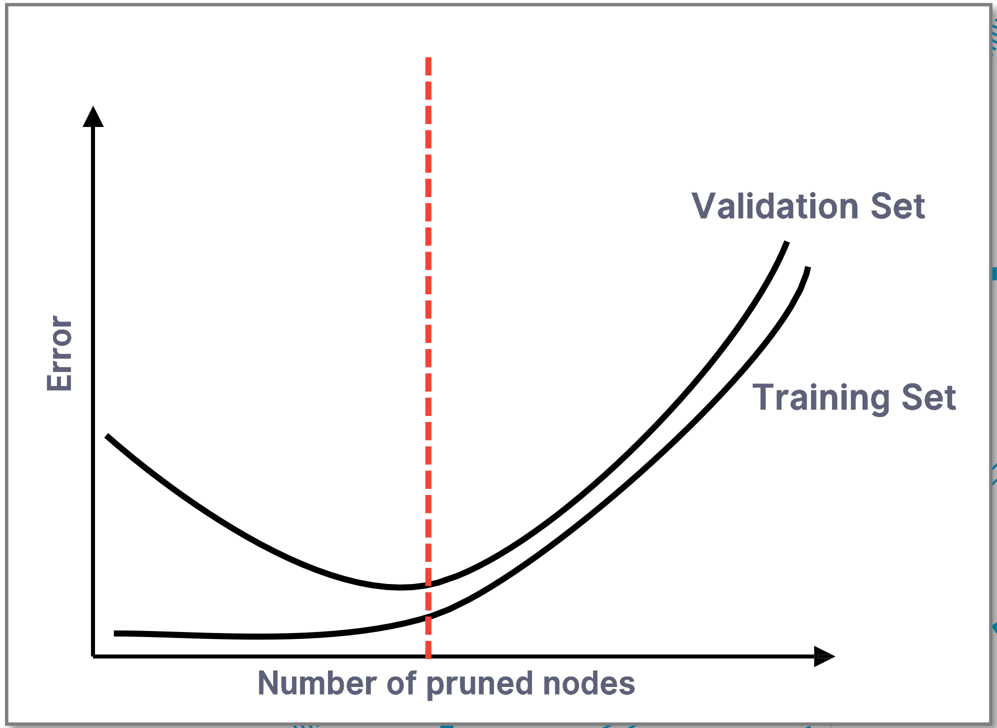




Post-Pruning

In **Post-Pruning**, we prune nodes in a bottom-up manner, if it decreases the validation error.





Decision Tree Algorithms

Decision Tree Algorithms

There are a few different variations of algorithms for constructing decision trees, each with different capabilities.

Two worth noting are:

- ID3
- C4.5

ID3 Algorithm

ID = Iterative Dichotomiser

1. Determine the entropy for the overall dataset using the class distribution.
2. For each feature:
   I. Calculate the entropy for categorical values.
   II. Assess the information gain for each unique categorical value of the feature.
3. Choose the feature that generates highest information gain.
4. Iteratively apply all above steps to build the decision tree structure.

C4.5 Algorithm

C4.5 is an extension of ID3, that brings several improvements:

- The ability to handle both categorical (discrete) **and** numerical (continuous) attributes.
    - Continuous attributes are split by finding a best-splitting threshold.
- The ability to handle missing values at both training and inference time.
    - During training, missing values are not used when information gain is computed.
    - During inference, missing values are handled by exploring all corresponding branches).

C4.5 Algorithm (Continued)

C4.5 is an extension of ID3, that brings several improvements:

- The ability to handle attributes with different costs.
- Post-pruning in a bottom-up manner, for removing branches that decrease the validation error (i.e. that increase generalization capacity).

Handling Numerical Attributes

Handling Numerical Attributes

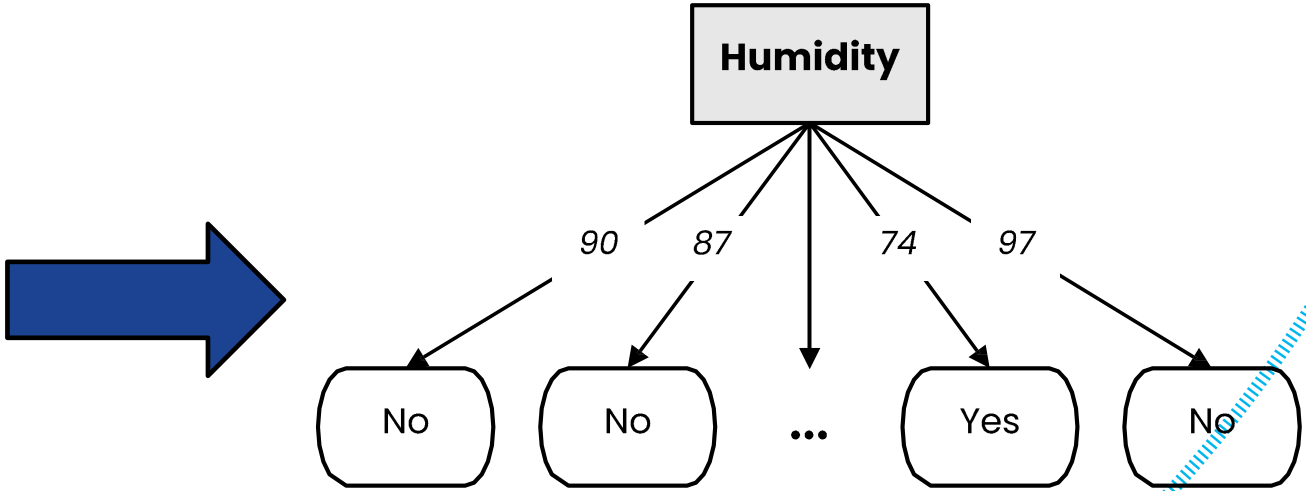
So how does C4.5 handle numerical attributes?

- Any numerical attribute would almost always bring entropy down to 0.
- In other words, it would completely overfit the training data.

Let's consider *Humidity* within our Play Tennis dataset…

Handling 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**



Handling Numerical Attributes

Numerical attributes have to be treated differently.

One way of doing this is to find the best splitting value:

- i.e. we calculate the Information Gain of numerical attribute a if we split at value t.

Handling Numerical Attributes

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

Calculate mean of each consecutive pair

Candidate split values
63.5
70
73
75.5
78
79.5
83.5
88
89.5
90.5
92
94.5
96.5

Calculate information 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

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**

So 83.5 is the best splitting value for *Humidity*, with an IG 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, and 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?	Colour	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 these 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?	Colour	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 Yes

2 Brown
4 White

1. Set them to the most common values.

Handling Missing Values at Training Time

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

1. Set them to the most common values.

2. Set them to the most probable value given the label.

$$P(Yes|Bird) = \frac{3}{4} = 0.75$$

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

$$P(No|Bird) = \frac{1}{4} = 0.25$$

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

Handling Missing Values at Training Time

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

1. Set them to the most common values.

2. Set them to the most probable value given the label.

3. Add a new instance for each possible value.

Handling Missing Values at Training Time

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

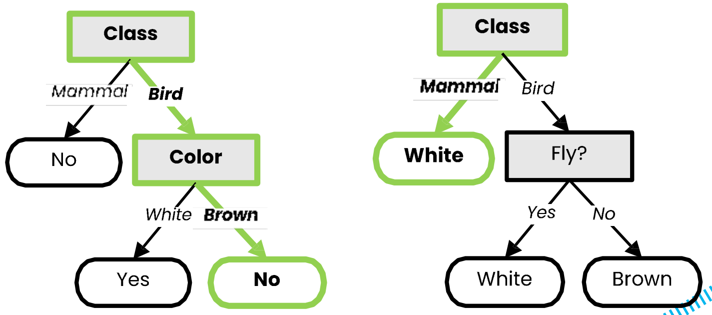
4. Leave them unknown, but discard the sample when evaluating the gain of that attribute.

Handling Missing Values at Training Time

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

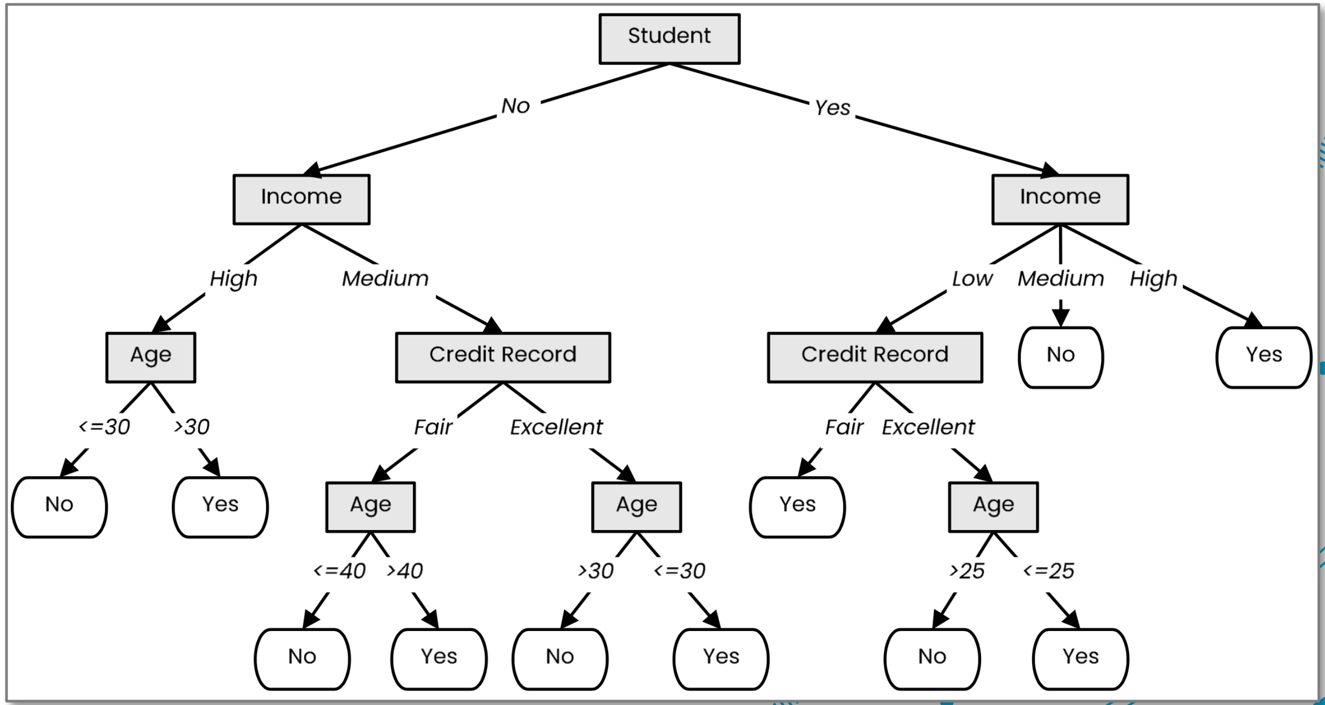
4. Leave them unknown, but discard the sample when evaluating the gain of that attribute.

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



Handling Missing Values at Inference Time

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



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.