

UNIVERSITY OF WARWICK

Lecture 6

Decision Trees

Gabriele Pergola and Amina Asif

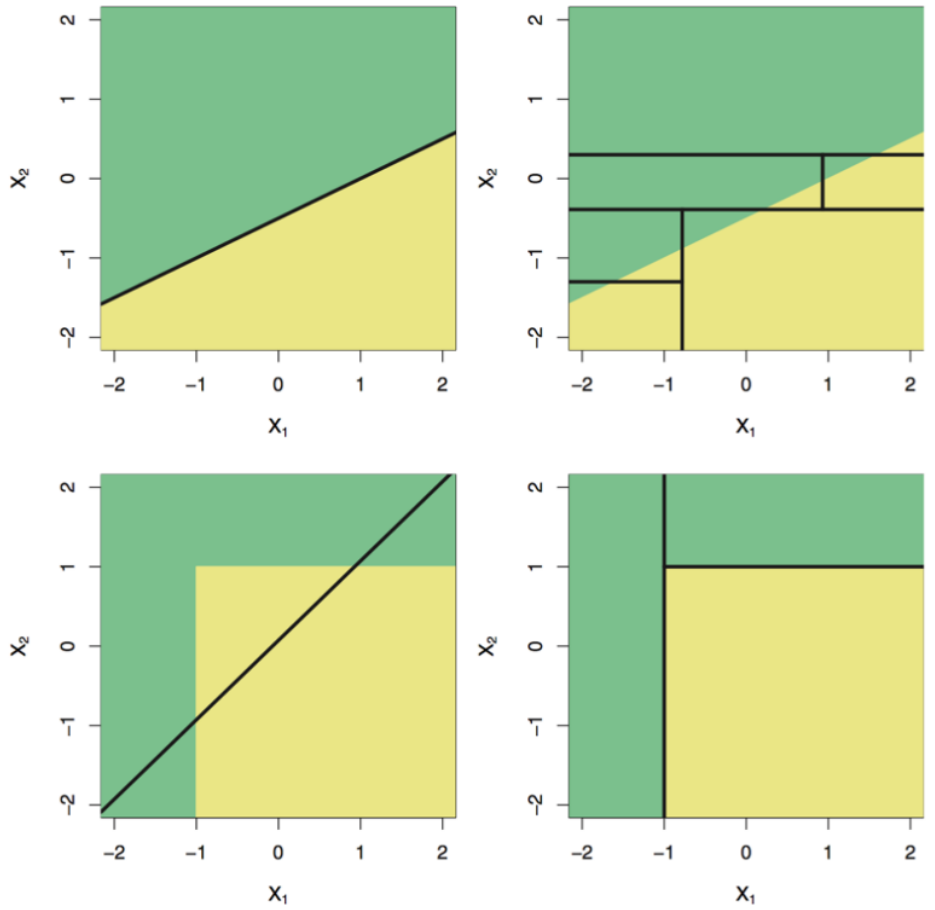
Machine Learning Algorithms and Practice
CS429

October 2026

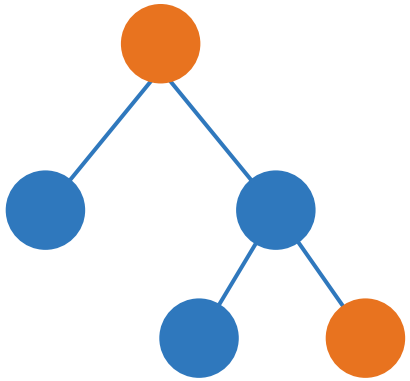
Tree vs. Linear Models

- Which one will perform better?
- Depends on your problem.

Consider the classification problem at the right, and the boundaries discovered by linear models versus a classification tree.



Classification: Decision Trees



Outline

- Decision Tree
- Avoid Overfitting
- Random Forest

Running Example: Predicting Tax Evasion

A tax return is the yearly form in which a person declares their income to the tax office.

Question: given a taxpayer's details, will they cheat, that is, evade tax by hiding income?

Input: attributes describing each taxpayer

Output: class label

Refund

Did the return end with money paid back?
Yes / No
Categorical

Marital Status

Their family status
Single, Married, Divorced
Categorical

Taxable Income

Yearly income, in thousands
60K, 95K, 220K, …
Continuous

Cheat

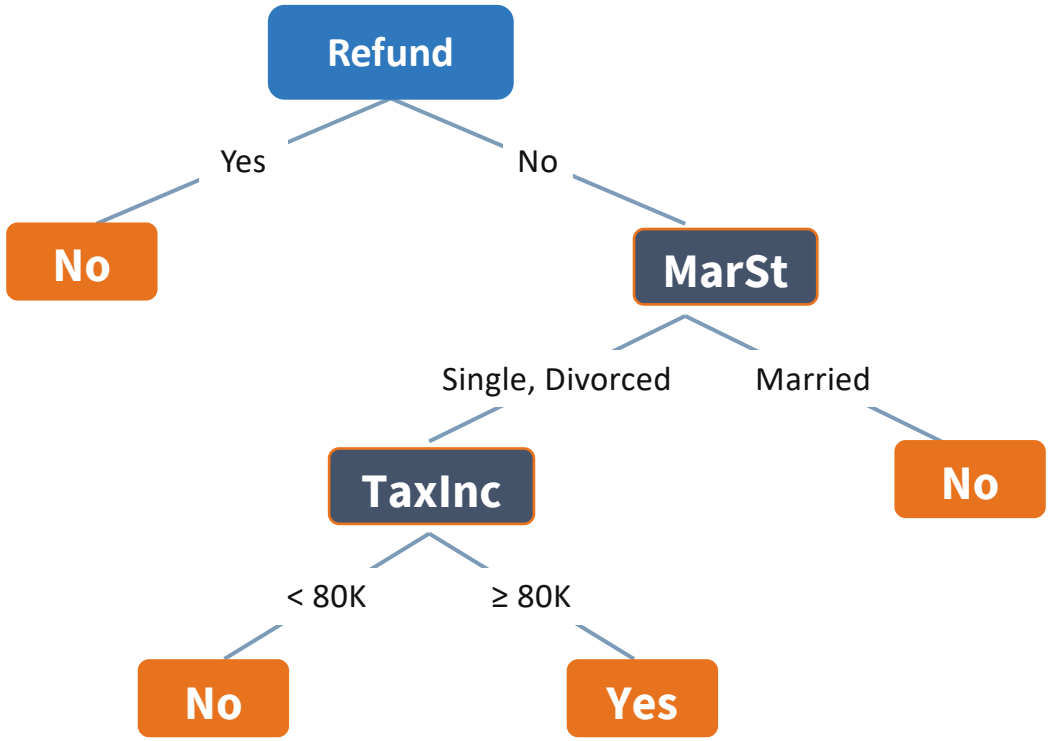
Did they evade tax, e.g. by hiding income?
Yes / No
The class to predict

Example of a Decision Tree

Training data

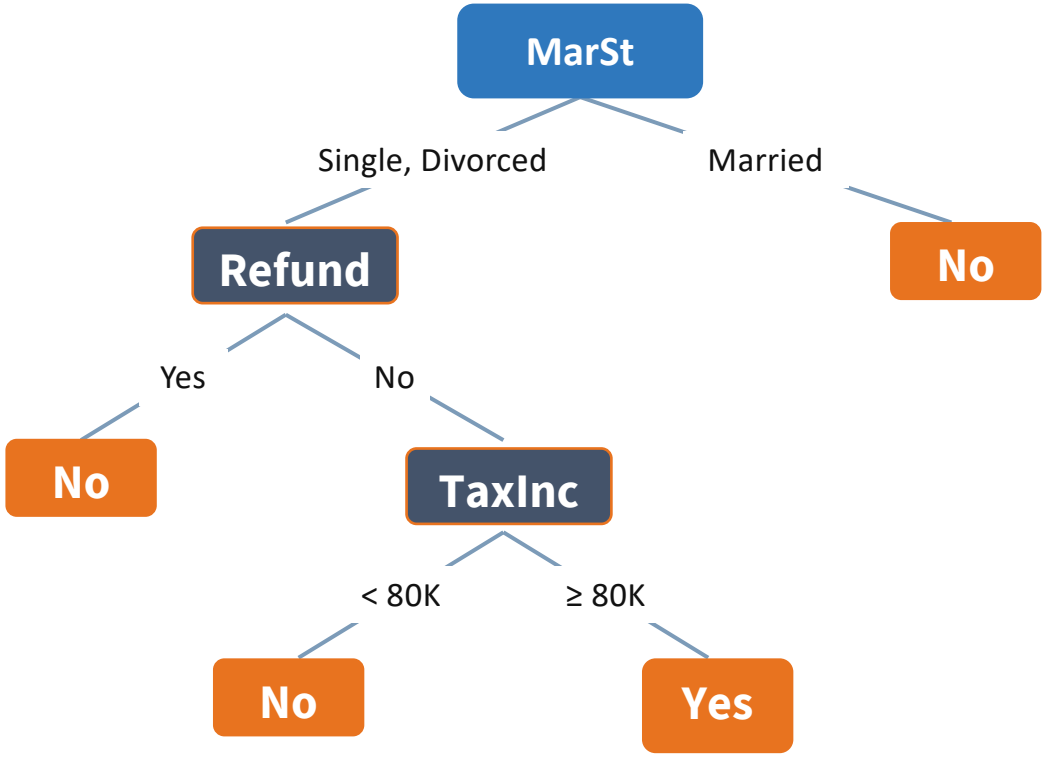
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Model: Decision Tree



Decision Tree Not Unique

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

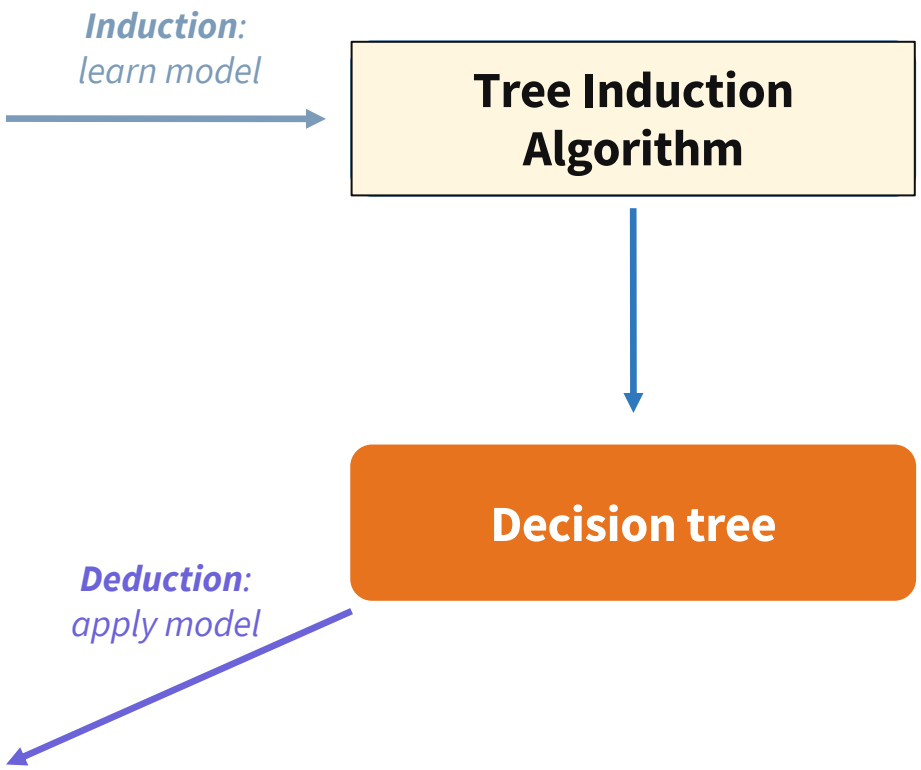
Decision Tree Classification Task

Training set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

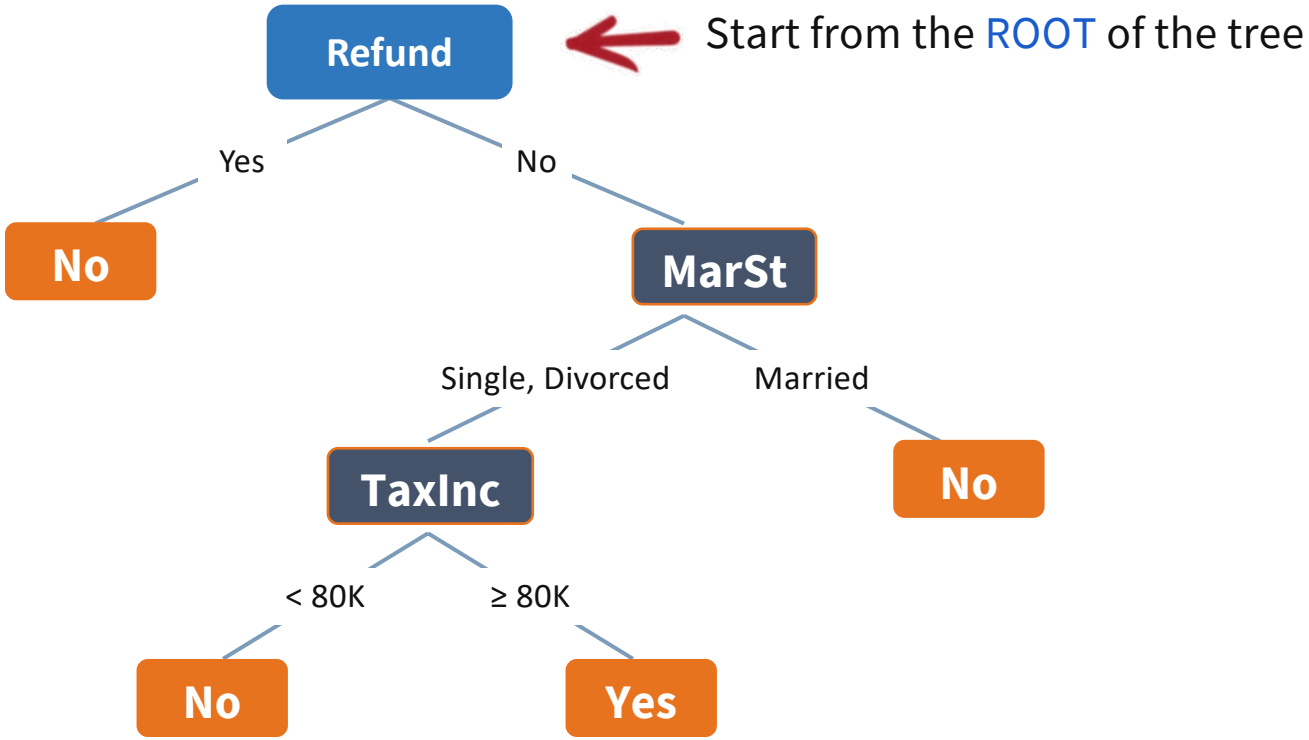


First: how is a learnt tree applied to a new record?

Apply Model to Test Data

Test record

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

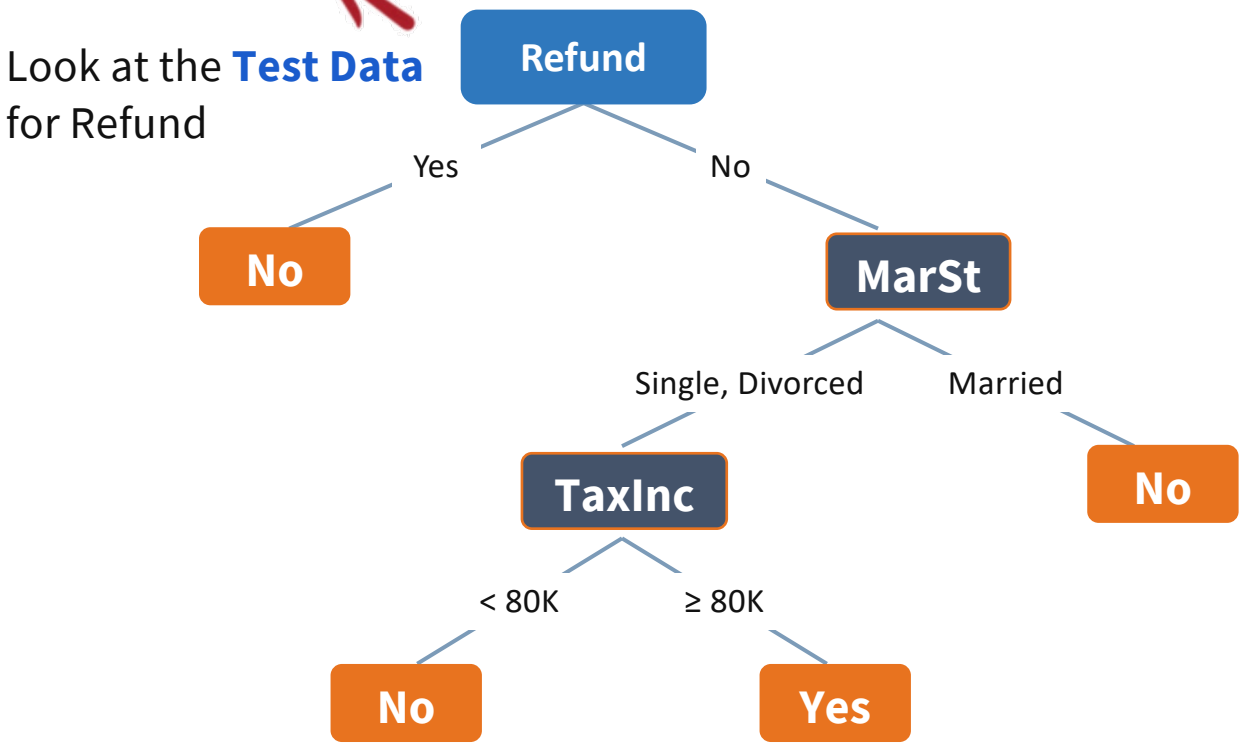


Look at Refund

Test record

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

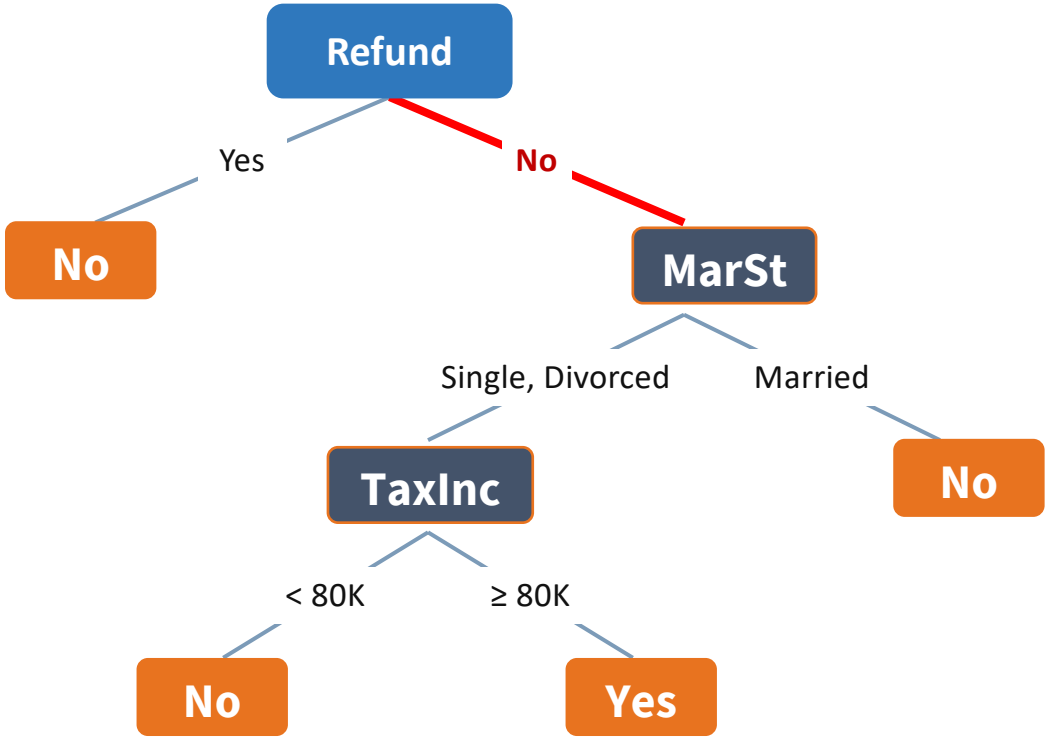
Look at the **Test Data** for Refund



Decide on Branch

Test record

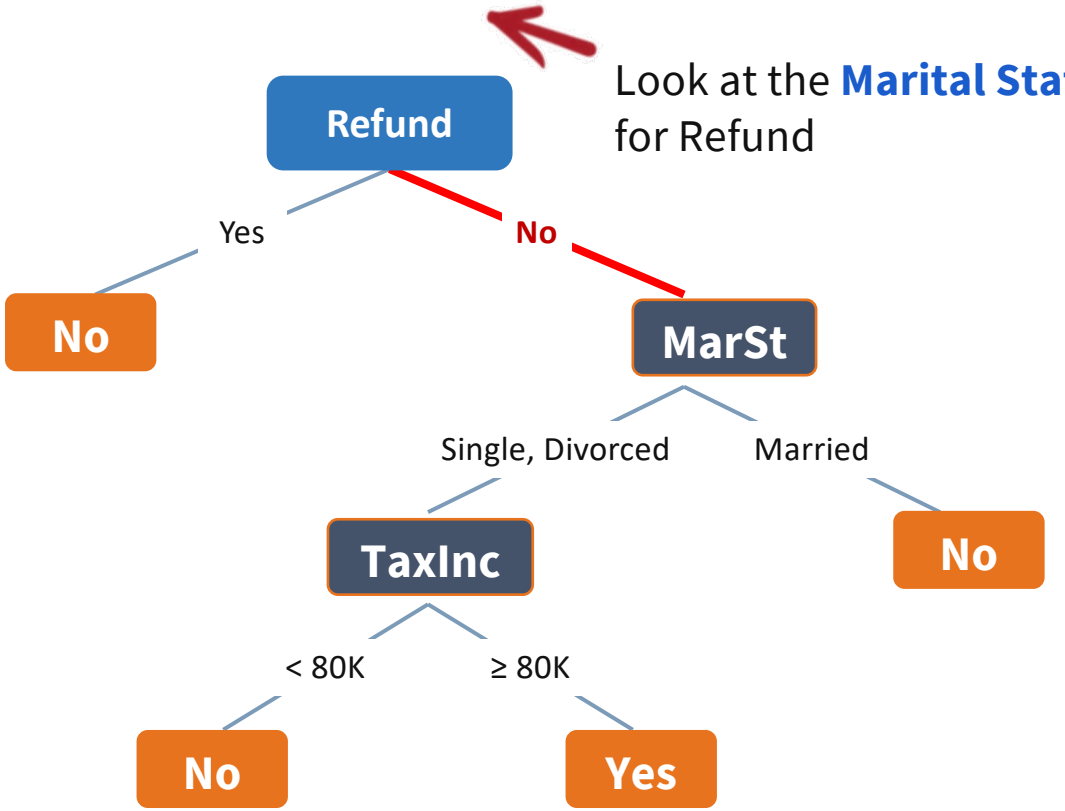
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Look at Marital Status

Test record

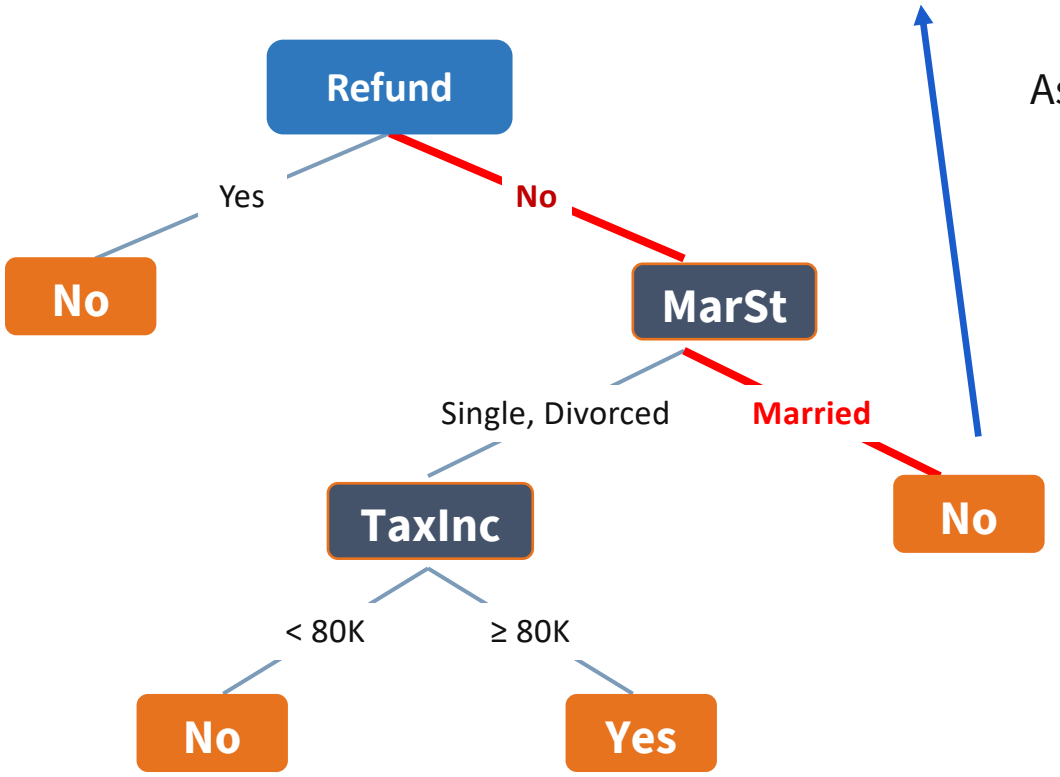
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Decide on Branch

Test record

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

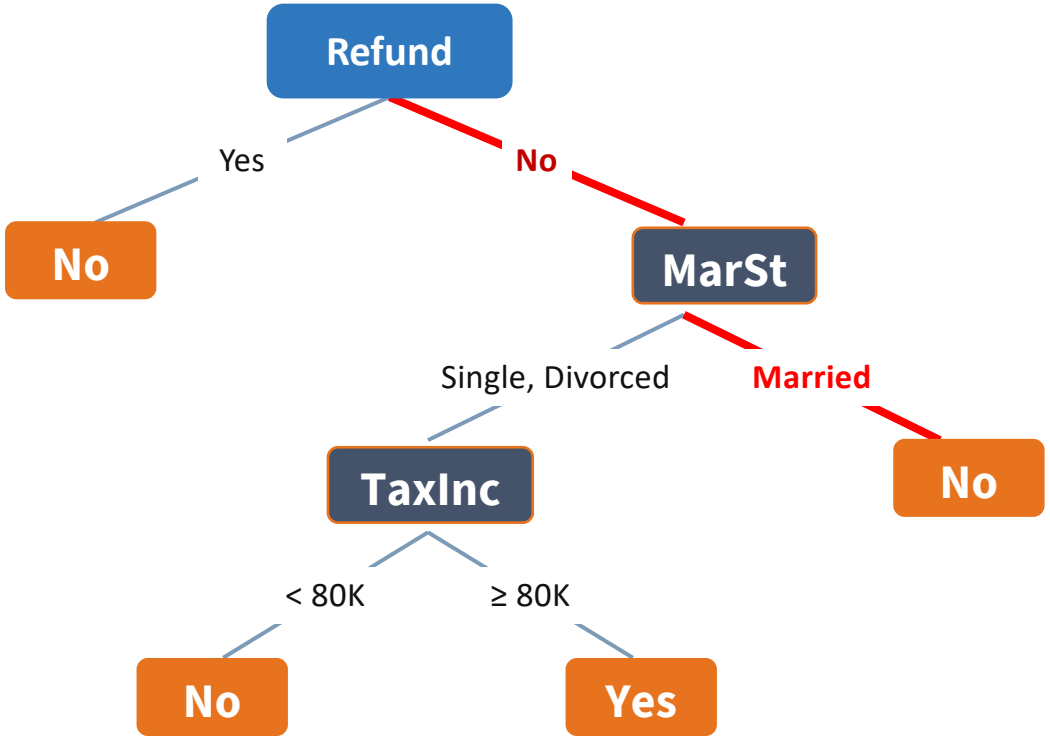


Assign **Cheat** to “No”

Reach Leaf Node: Final Decision

Test record

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	No



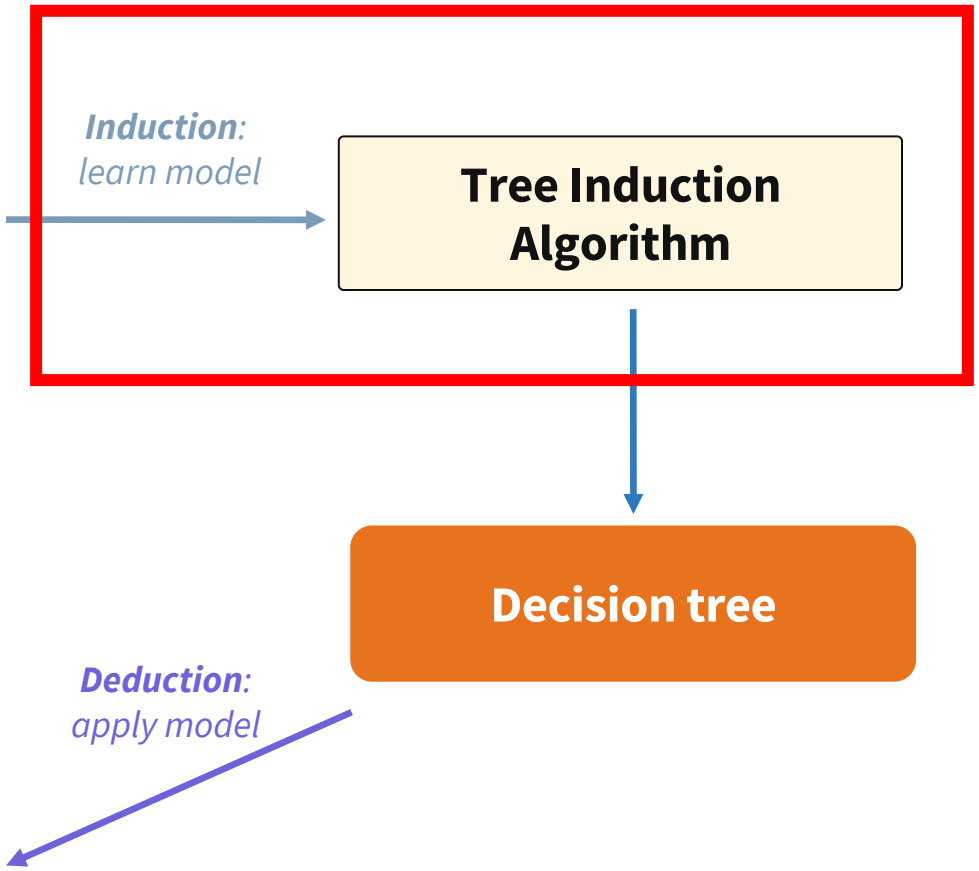
Decision Tree Classification Task

Training set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?



First: how is a learnt tree applied to a new record?

Decision Tree Induction

Splitting records, choosing the best split, and knowing when to stop

Decision Tree Induction

Many algorithms share one greedy, top-down recipe; they differ mainly in the impurity measure.

1966	1984	1986	1993	1996
Hunt's algorithm *Hunt, Marin and Stone* General recursive scheme	**CART** *Breiman et al.* Classification and Regression Trees; Gini	**ID3** *Quinlan* Iterative Dichotomiser 3. Entropy and information gain; categorical attributes	**C4.5** *Quinlan* Successor to ID3. Adds gain ratio, continuous attributes, pruning	**SLIQ, SPRINT** *Mehta et al.; Shafer et al. (IBM)* Scalable induction; Gini

Tree Induction

Greedy strategy

- Split the records based on an attribute test that optimises certain criterion, then repeat on each child.

Issues

- How to split the records
  - How to specify the attribute test condition?
  - How to determine the best split?
- When to stop splitting

Why greedy?

Finding the smallest tree that fits the data is **NP-hard**.

Choosing the locally best split at each node is fast and works well in practice, but an **early poor choice is never undone**.

Tree Induction

Greedy strategy

- Split the records based on an attribute test that optimises certain criterion, then repeat on each child.

Issues

- How to split the records
  - **How to specify the attribute test condition?**
  - How to determine the best split?
- When to stop splitting

How to Specify Test Condition?

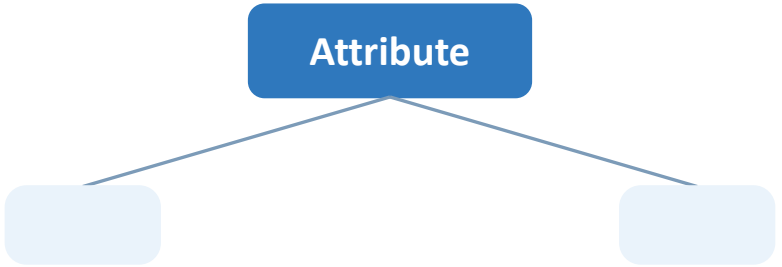
Depends on attribute type

Nominal
Unordered categories — *Car type: Family, Sports, Luxury*

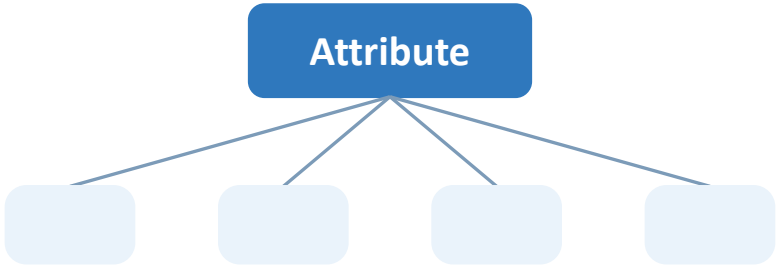
Ordinal
Ordered categories — *Size: Small < Medium < Large*

Continuous
Numeric values — *Taxable income: 60K, 95K, 220K*

Depends on number of ways to split



2-way (binary) split



Multi-way split

Tree Induction

Greedy strategy

- Split the records based on an attribute test that optimises certain criterion, then repeat on each child.

Issues

- How to split the records
  - **How to specify the attribute test condition?**
  - How to determine the best split?
- When to stop splitting

Tree Induction

Greedy strategy

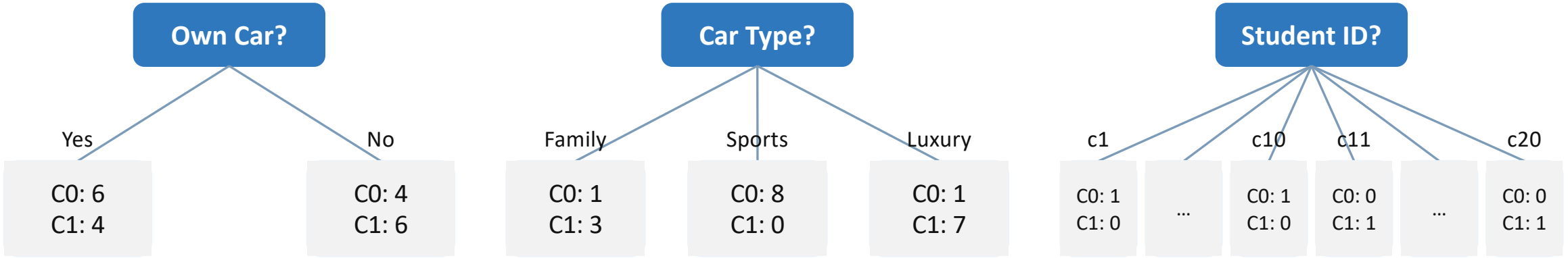
- Split the records based on an attribute test that optimises certain criterion, then repeat on each child.

Issues

- How to split the records
  - How to specify the attribute test condition?
  - **How to determine the best split?**
- When to stop splitting

How to Determine the Best Split

Before splitting: 10 records of class C0 and 10 of class C1. C0 and C1 are just names for two classes (say, Cheat = No and Yes); each box shows how many records of each class it holds. Which test is best?



Little information
Barely changes the class mix

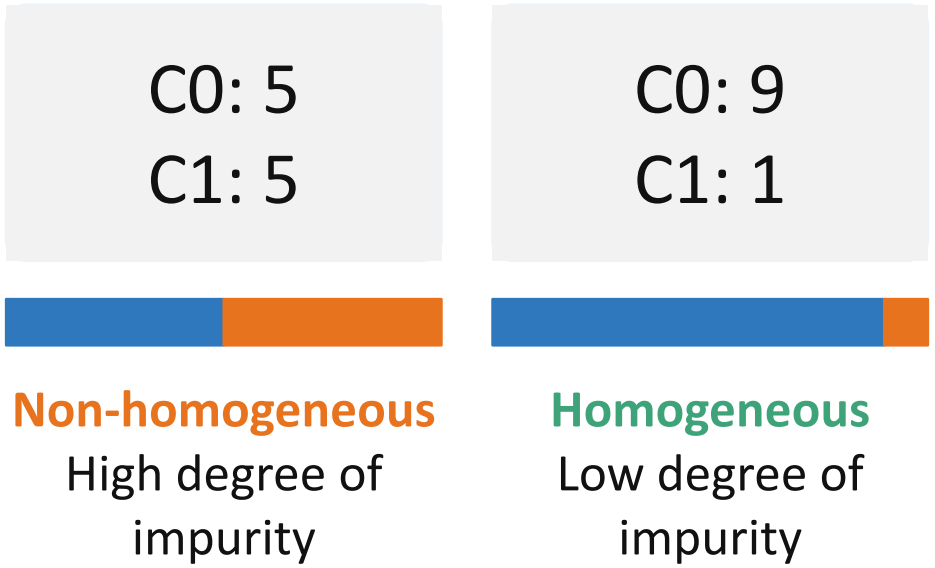
Promising
Children are close to pure

Pure, but cannot generalise
Twenty pure children of one record

We need a number that scores how "**clean**" each set of children is.

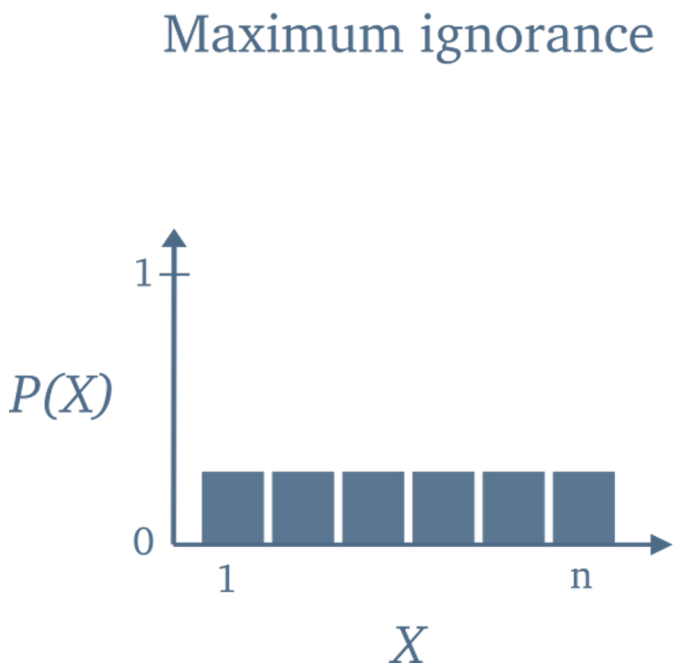
How to Determine the Best Split

- Greedy approach:
  - the best split at the root or a child node is the one that does the best job of separating the data into groups where a single class predominates
  - Nodes with homogeneous class distribution are preferred
- We therefore need a measure of **node impurity**



But first… Entropy!

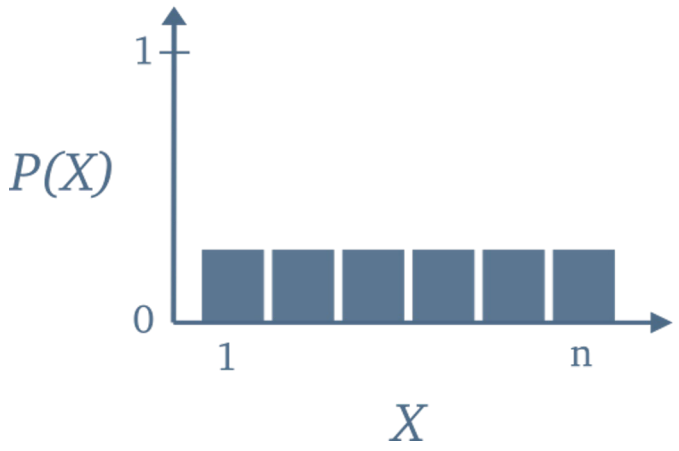
- The entropy of a probability distribution is a measure of **uncertainty**
  - It is maximized when the distribution is as **uniform** as possible



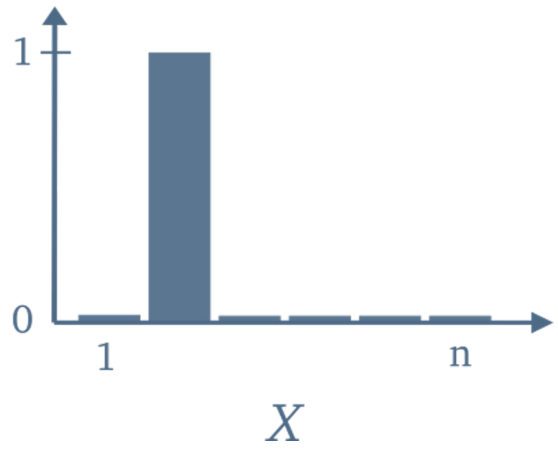
But first… Entropy!

- The entropy of a probability distribution is a measure of **uncertainty**
  - It is maximized when the distribution is as **uniform** as possible

Maximum ignorance



Absolute certainty



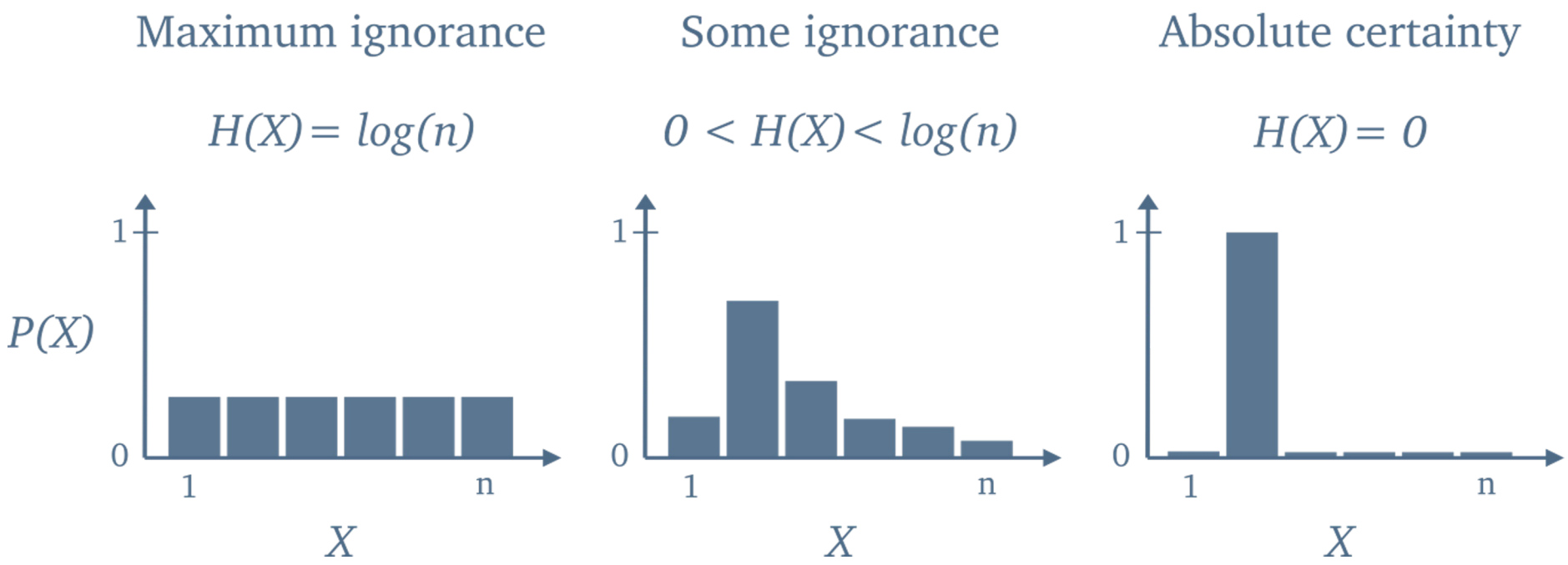
But first… Entropy!

- The entropy of a probability distribution is a measure of **uncertainty**
  - It is maximized when the distribution is as **uniform** as possible



But first… Entropy!

- The entropy of a probability distribution is a measure of **uncertainty**
  - It is maximized when the distribution is as **uniform** as possible

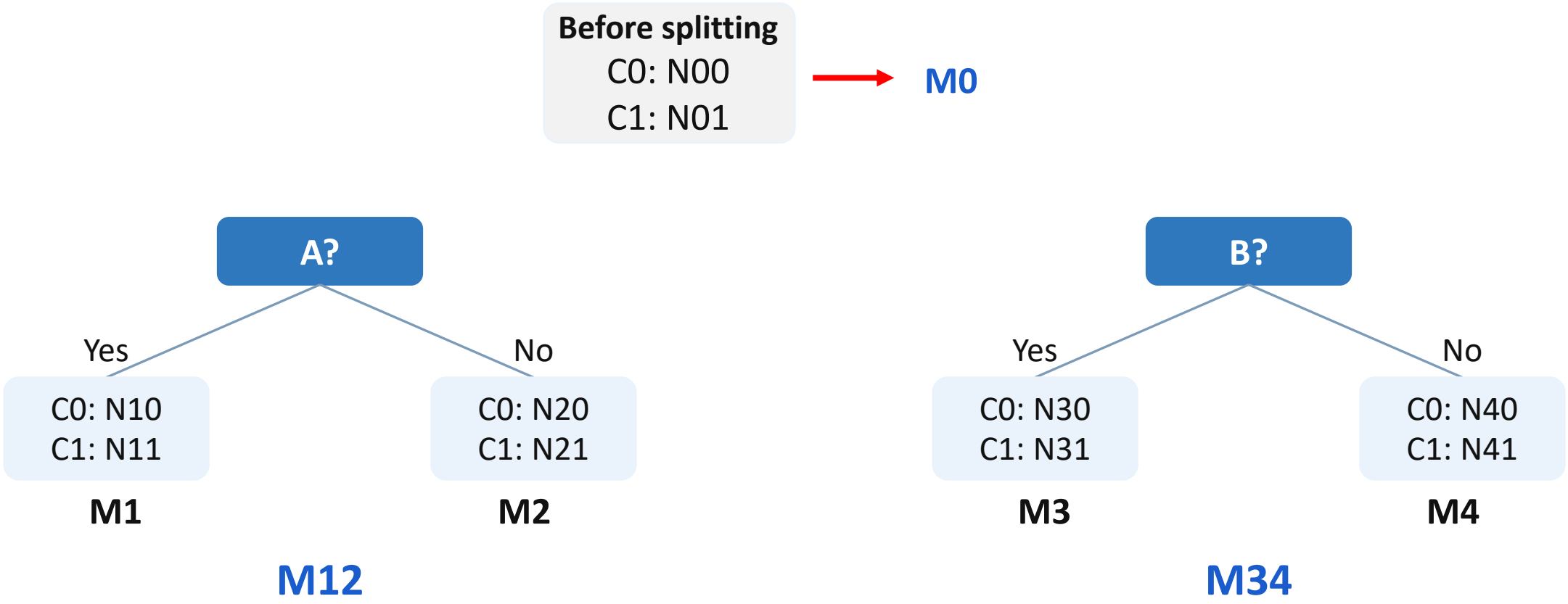


Measures of Node Impurity

- Entropy
- Gini Index

Both measures are 0 for a pure node and largest when all classes are equally frequent.

How to Find the Best Split

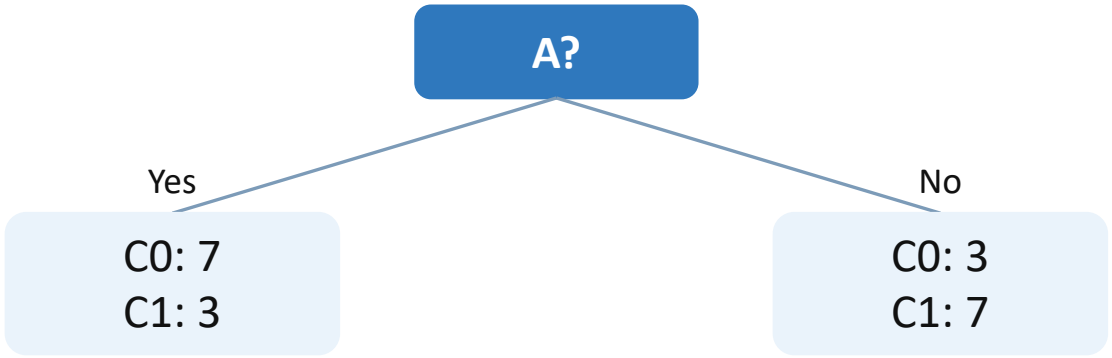


Gain = M0 – M12 **vs** M0 – M34
Choose the test with the larger gain

Best Split: A Worked Example

E.g., Entropy =
$-(1/10)\log_2(1/10) - (9/10)\log_2(9/10)$

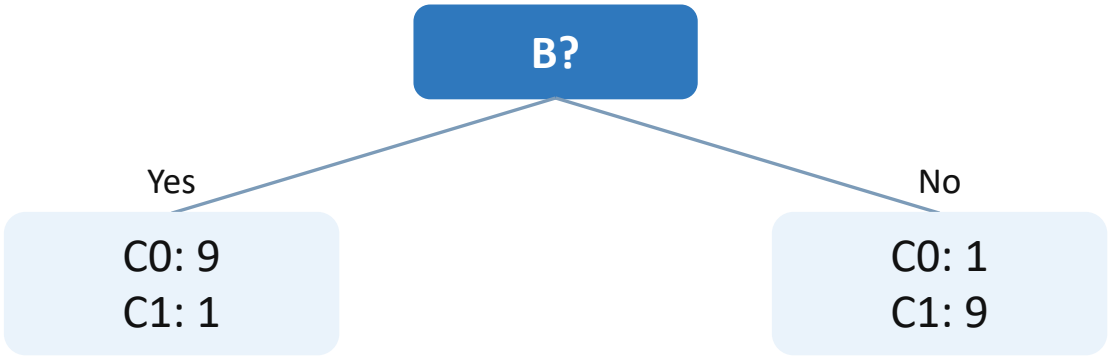
- **Parent node:** 20 records, **C0 = 10** and **C1 = 10**. **Entropy M0** = 1.0 (maximum impurity for two classes).



Entropy 0.881 **Entropy 0.881**

Weighted: (10/20) × 0.881 + (10/20) × 0.881 = 0.881

Gain = 1.000 − 0.881 = **0.119**



Entropy 0.469 **Entropy 0.469**

Weighted: (10/20) × 0.469 + (10/20) × 0.469 = 0.469

Gain = 1.000 − 0.469 = **0.531**

B has the larger gain, so we split on B.

Tree Induction

Greedy strategy

- Split the records based on an attribute test that optimises certain criterion, then repeat on each child.

Issues

- How to split the records
  - How to specify the attribute test condition?
  - **How to determine the best split?**
- When to stop splitting

Tree Induction

Greedy strategy

- Split the records based on an attribute test that optimises certain criterion, then repeat on each child.

Issues

- How to split the records
  - How to specify the attribute test condition?
  - How to determine the best split?
- **When to stop splitting (i.e., a node is a leaf)**

When to Stop Splitting?

Stop expanding a node when:

- **All records belong to the same class**
  The node is pure; nothing left to separate.

- **All records have “similar” attribute values**
  No test can separate them any further.

- **Some other criterion is met**
  For example a *maximum depth* or minimum node size.

Decision Tree Based Classification

Advantages

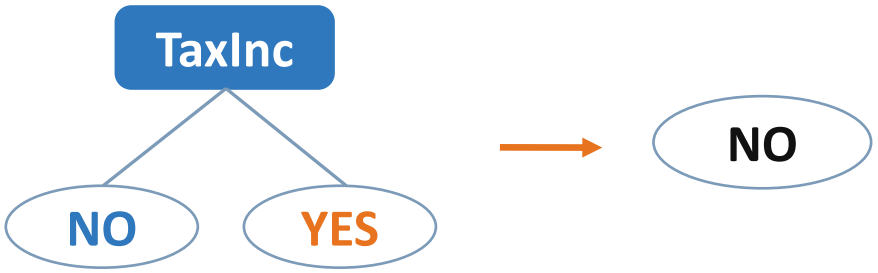
- **Inexpensive** to construct
- Extremely **fast** at classifying unknown records
- **Easy to interpret** for small-sized trees
- Accuracy comparable to other classification techniques for many simple data sets

Limitations

- **Greedy**: early choices are never revisited
- **Unstable**: small changes in the data can produce a very different tree
- Each test uses one attribute, so decision boundaries are axis-parallel
- Prone to **overfitting without pruning**

How to Address Overfitting with Post-Pruning

Example



Validation errors: 3 Validation errors: 2

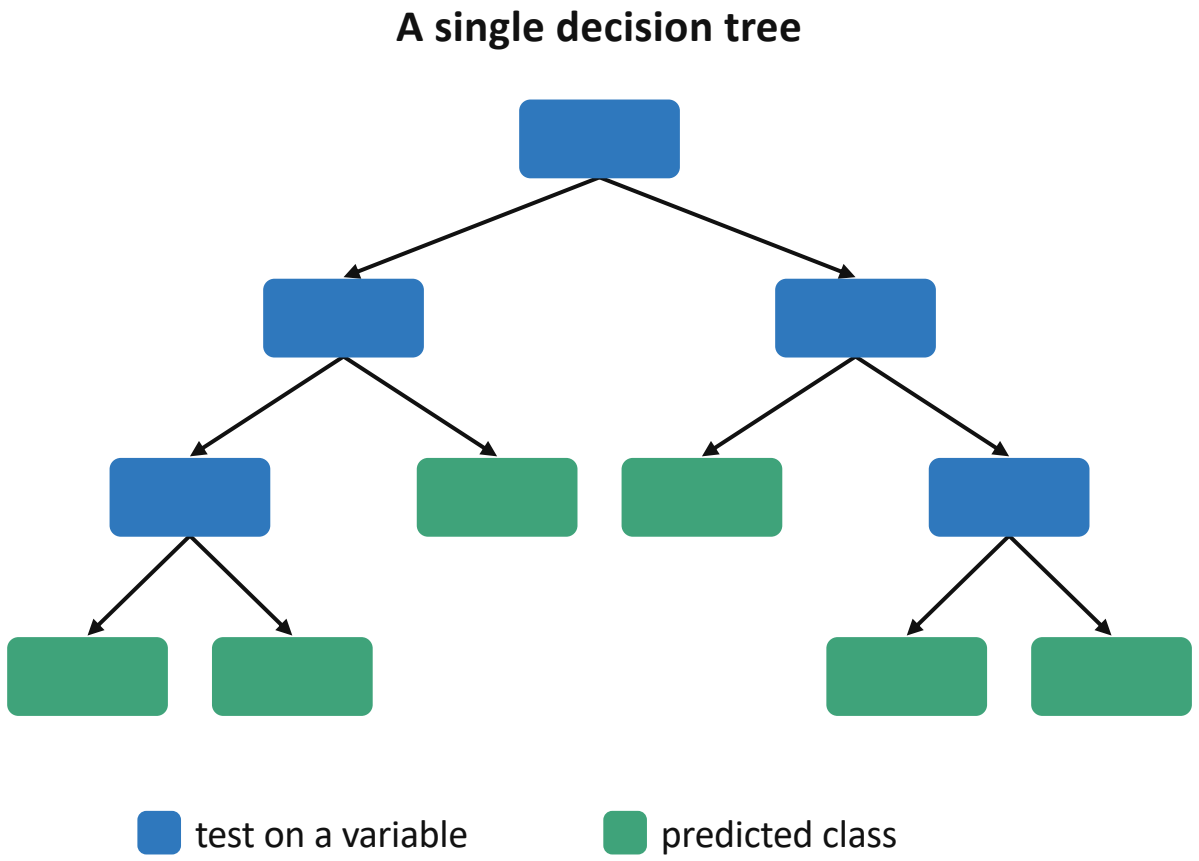
- **Replacing the sub-tree with a single leaf** (its majority class, NO) reduces validation error from 3 to 2, so the sub-tree is pruned.

  Repeat for the next node up.

Random Forests

Let’s build the forest!
Many randomised trees combined into one more accurate classifier

Why a Single Tree Is Not Enough



Strengths

- Simple to build and easy to read
- Fits the data used to build it very well

Weaknesses

- **Not flexible (overfitting)** when classifying new samples
- A small change in the data can give a very different tree
- Accuracy is often modest

Recall: Variance of the Mean

- The variance of the mean of n independent observations, each with variance σ^2, is σ^2/n.

In other words: averaging a set of observations reduces variance.

We could sample many training sets from the population, build a separate prediction model on each one, and average the predictions to obtain a model with lower variance.

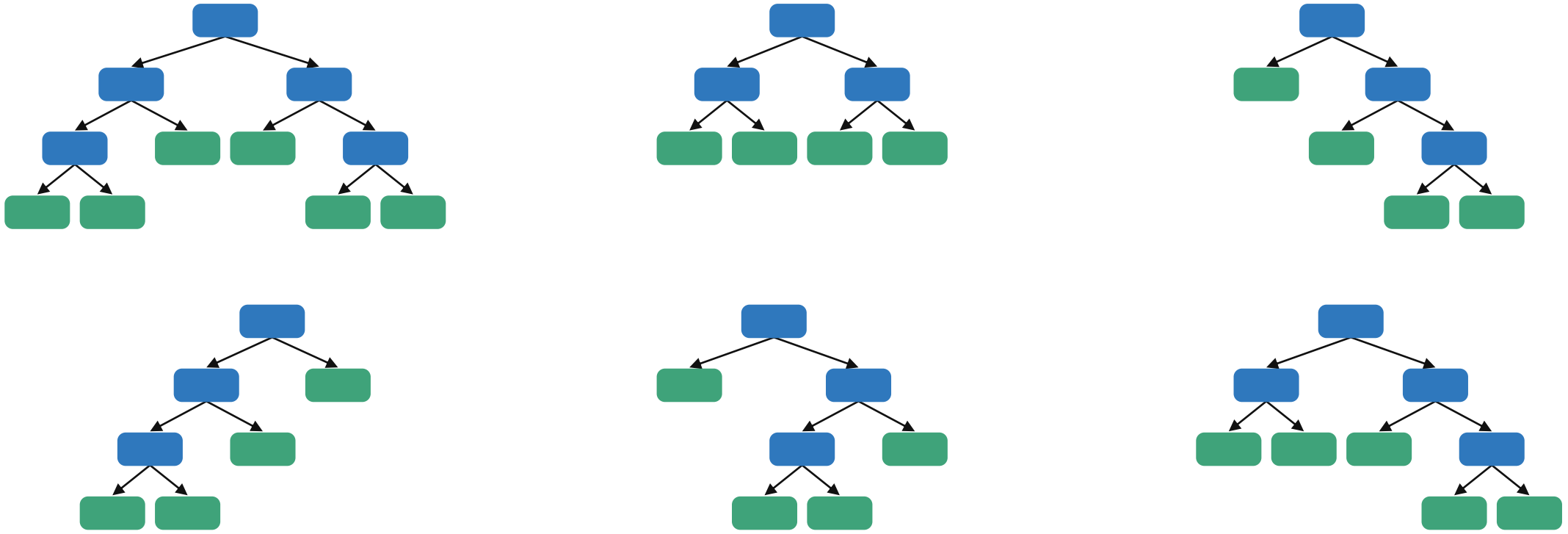
$$f^1(x),\ f^2(x),\ \ldots,\ f^B(x) \quad \rightarrow \quad f_{avg}(x) = (1 / B)\ \Sigma_{b=1..B} f^b(x)$$

Example
If $\sigma^2 = 16$ and we average n = 4 observations, the variance falls to 16 / 4 = 4, so the standard deviation halves from 4 to 2.

The catch, and the link to forests
We have only one training set. Bootstrapping imitates many, and random variable subsets keep the trees different, so averaging helps more.

The Idea: a Forest of Trees

Random forests combine the simplicity of decision trees with much more flexibility, giving a large improvement in accuracy. **The recipe: grow many different trees, then combine them**.



How are these trees built, and why are they all different?

Building a Random Forest

Step 1

Bootstrapped dataset

Resample the records at random, with replacement, to make a new dataset of the same size.



Step 2

Tree on random variables

Grow a decision tree from it, considering only a random subset of the variables at each step.



Step 3

Repeat

Go back to step 1 and build another tree, many times over.

Step 1: Create a Bootstrapped Dataset

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
?	*?*	*?*	*?*	*?*
?	*?*	*?*	*?*	*?*
?	*?*	*?*	*?*	*?*
?	*?*	*?*	*?*	*?*

- Imagine that these 4 records are the entire data set we build a tree from.
- **A bootstrapped dataset** has the same size as the original and is built by drawing records at random, with replacement.

Drawing Records at Random

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
?	?	?	?	?
?	?	?	?	?
?	?	?	?	?

Draw 1: record 2 is picked at random and copied into the new dataset.

The record also stays in the original dataset, so it can be picked again.

Drawing Records at Random

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
?	?	?	?	?
?	?	?	?	?

Draw 2: record 1 is picked and copied into the second row of the new dataset.

The Same Record Can Be Drawn Twice

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Draws 3 and 4: record 4 is picked both times.

Sampling is done with replacement, so the same record may appear more than once in the new dataset.

The Bootstrapped Dataset

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

- **We have created a bootstrapped dataset.**
- Some records appear several times; others, such as record 3 here, do not appear at all.
- *Left-out records are not wasted: they become useful later.*

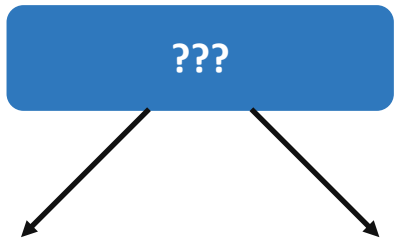
Step 2: Build a Tree on Random Variables

- Build a decision tree from the bootstrapped dataset
- At each step, consider only a random subset of the variables (columns) as candidate tests
- In this example: 2 variables at each step
  - How many to use is discussed later

Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Choosing the Root: Random Candidates



Instead of scoring all 4 variables for the root, we draw 2 at random: **Peak Hour** and **Late Inbound Aircraft**

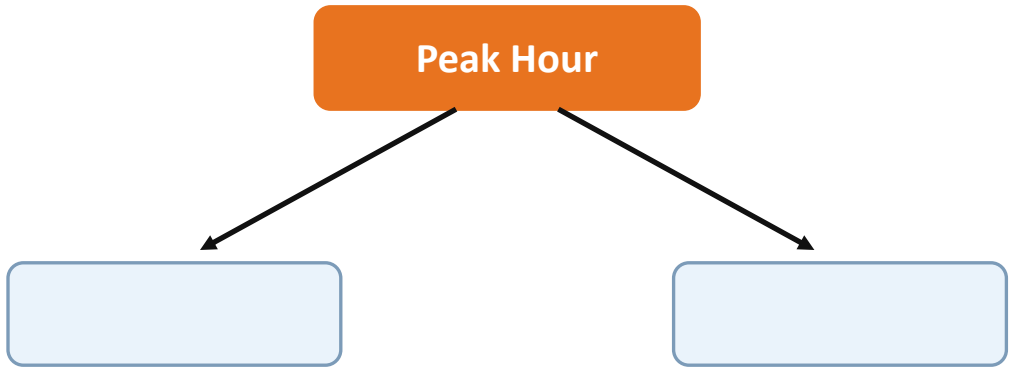
Only these two are scored, for example with information gain. The better one becomes the root.

Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

candidate not drawn

Root Node Selected



Just for the sake of the example, assume that **Peak Hour** did the best job of separating the records. It becomes the root node.

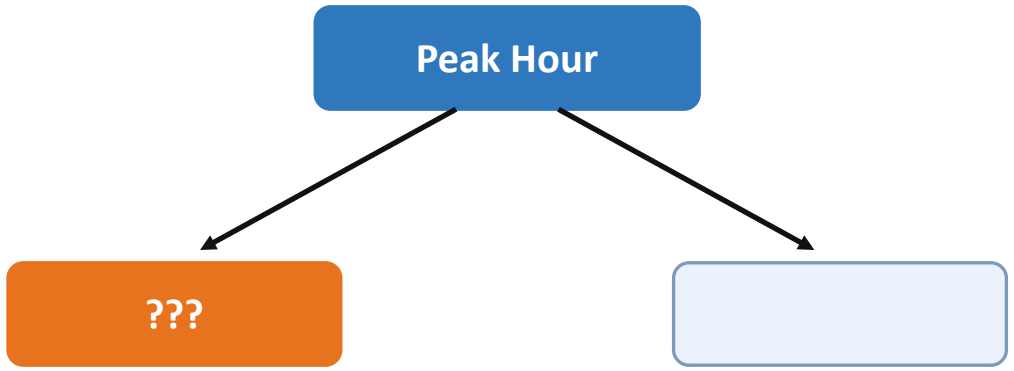
Note: Late Inbound Aircraft separates these four records perfectly. We keep Peak Hour so that the example has more steps to show.

Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

chosen for the root

Next Node: A New Random Subset



Peak Hour was used above, so we grey it out: a yes/no variable is constant below its own split.

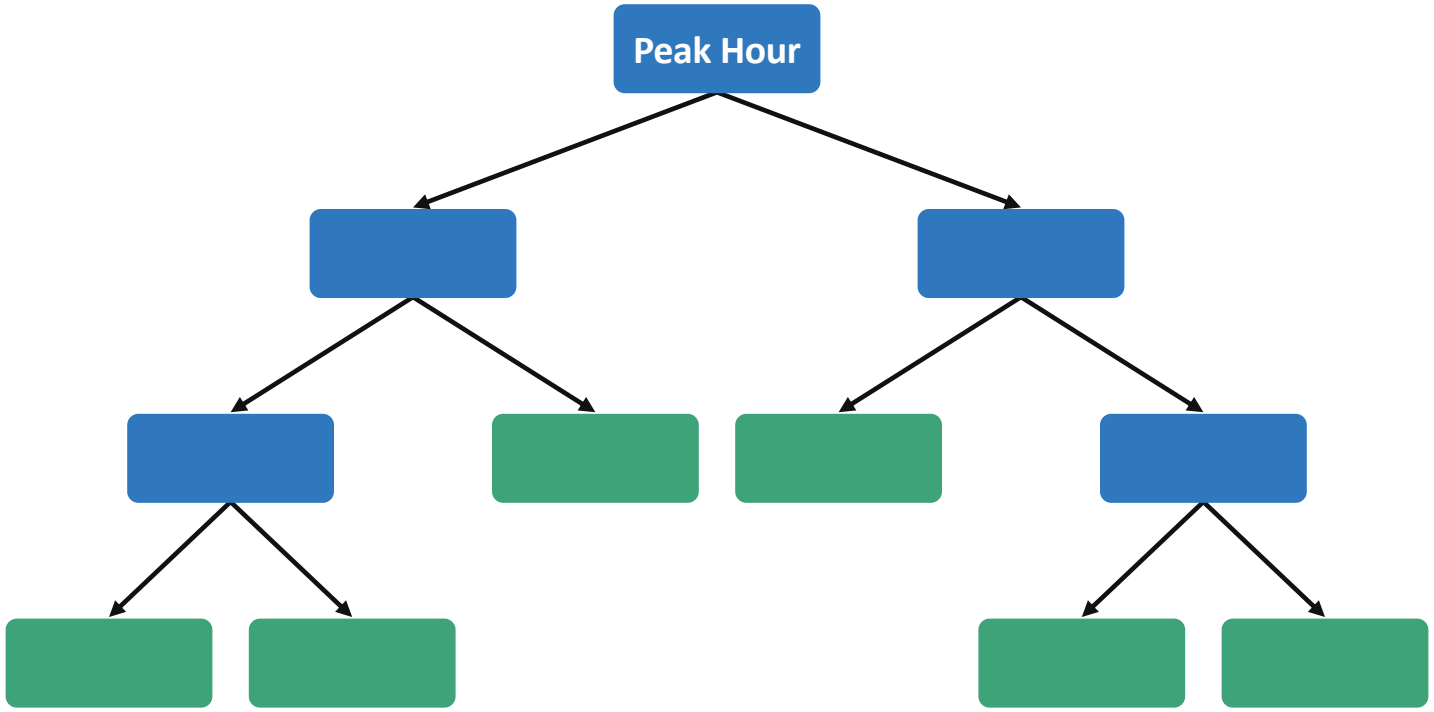
Just as for the root, we draw 2 of the 3 remaining variables at random: **Bad Weather** and **Passengers**.

Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

candidate used

A Tree Built from Randomness



Bootstrapped dataset

- The tree is grown from a random resample of the records.

Random variable subsets

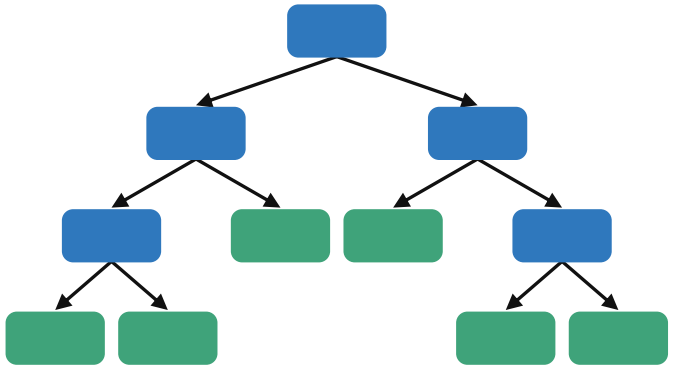
- Each node chooses its test from only a random handful of variables.

We built a tree using both sources (samples and variables) of randomness.

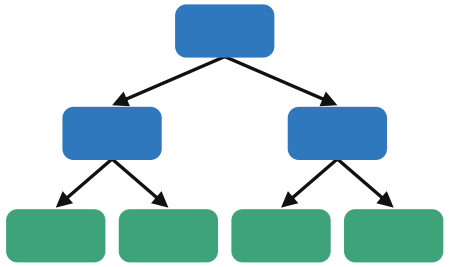
Now Repeat

Go back to Step 1 and repeat: make a new bootstrapped dataset and build another tree, again choosing from a random subset of variables at each step.

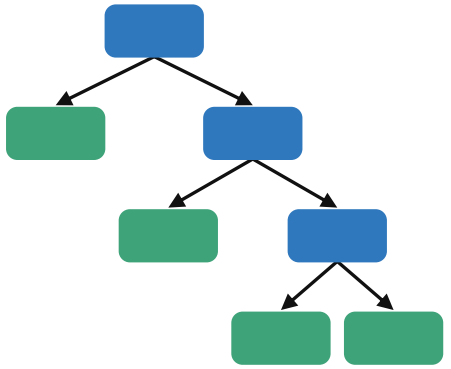
Tree 1



Tree 2



Tree 3

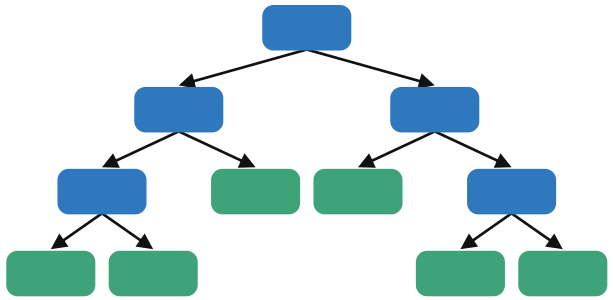


Every tree is different

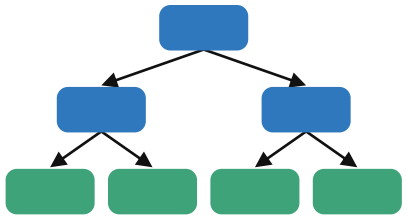
Each tree sees different records and different variables. Typically **hundreds of trees** are grown. Next: combining them.

A Wide Variety of Trees: A Random Forest

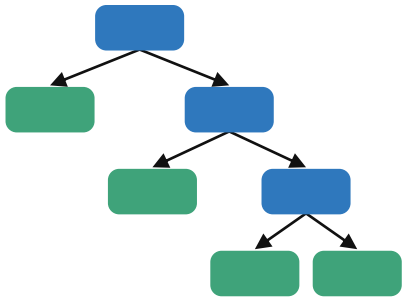
Tree 1



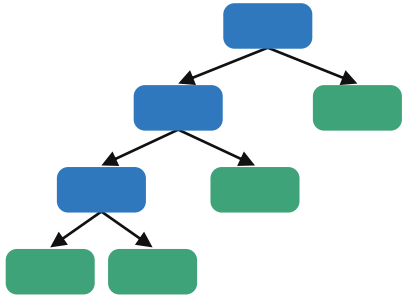
Tree 2



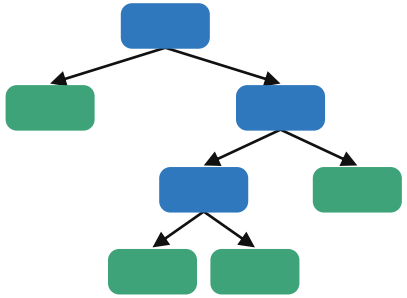
Tree 3



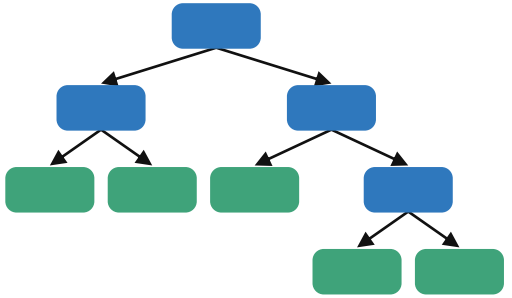
Tree 4



Tree 5



Tree 6



Using the Forest: Let the Trees Vote

A new flight: delay unknown

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	No	No	168	?

Run it down every tree

The new record follows its own path through each of the trees.

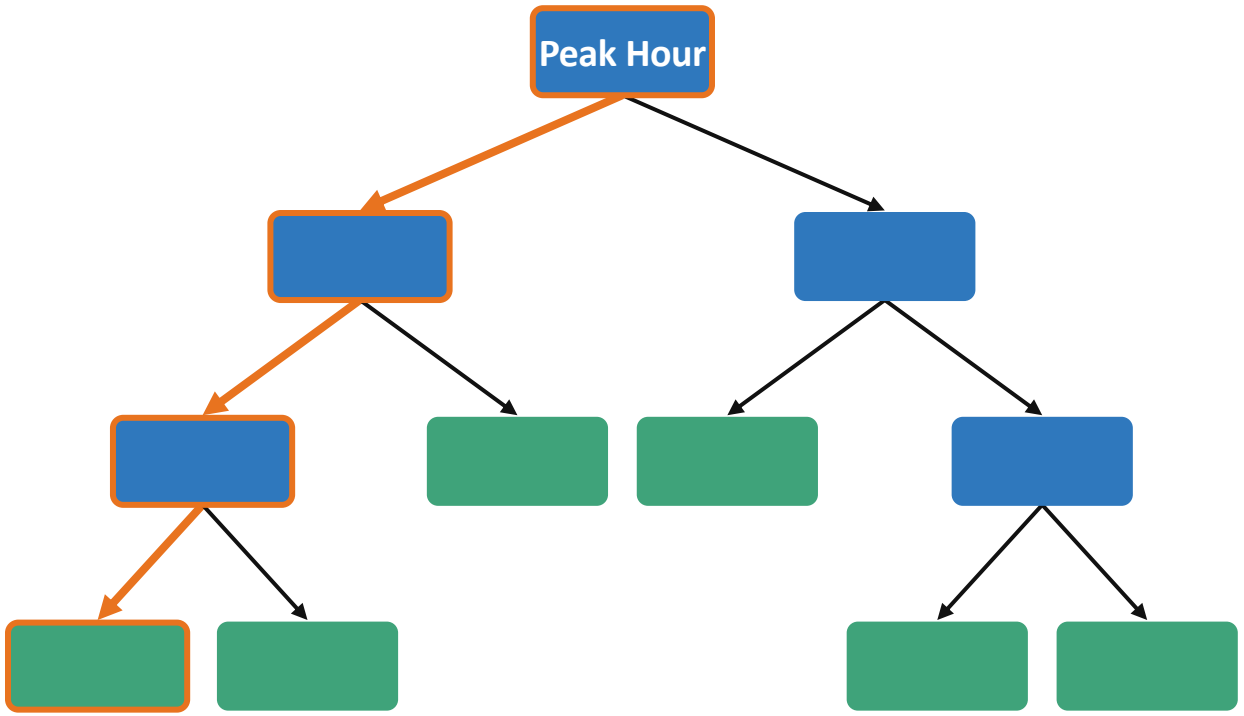
Each tree votes

The leaf it reaches gives that tree's predicted class: Yes or No.

Majority wins

The class with the most votes is the forest's prediction.

Run the Record Down the First Tree

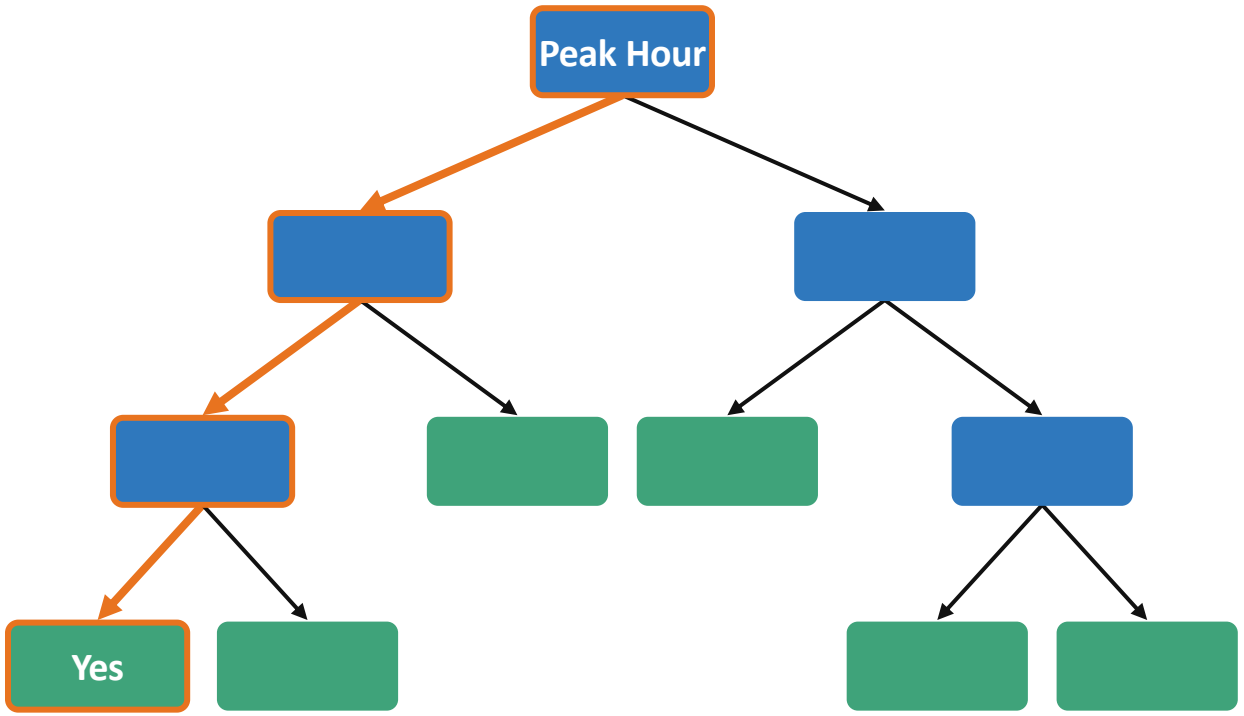


New record

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	No	No	168	?

Take the new record and run it down the first tree we built.

The First Tree Votes



New record

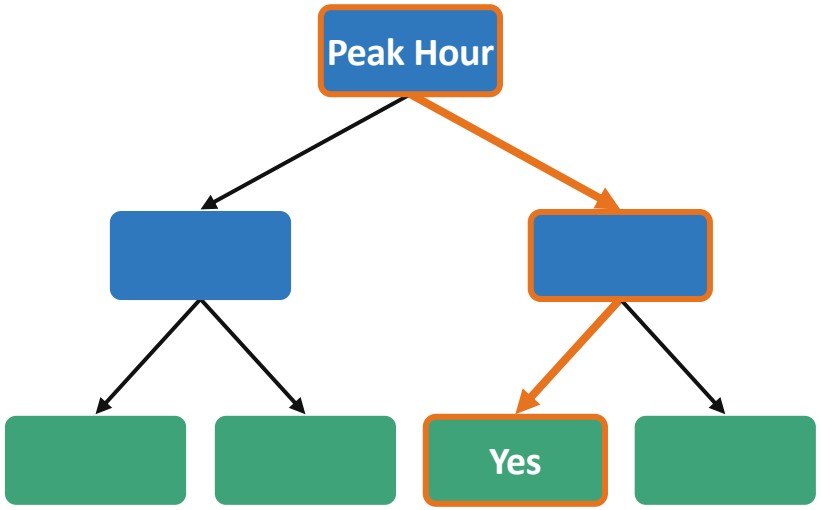
Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	No	No	168	?

Votes

Yes	No
1	0

The first tree says “Yes”, and we keep track of that vote in the tally.

The Second Tree Also Says Yes



New record

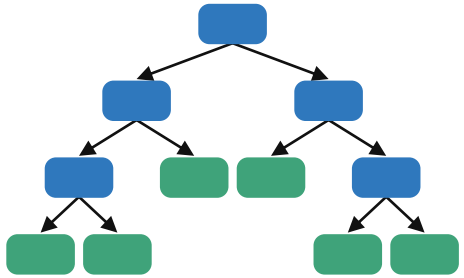
Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	No	No	168	?

Votes

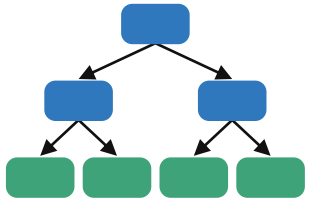
Yes	No
2	**0**

The second tree takes a different route, but it also says “Yes”.

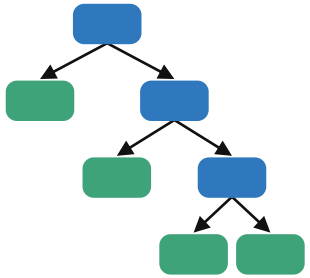
Repeat for All the Trees



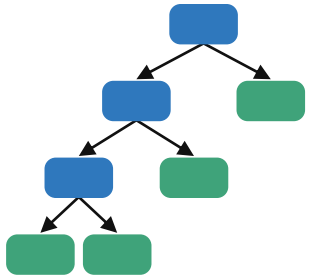
Yes



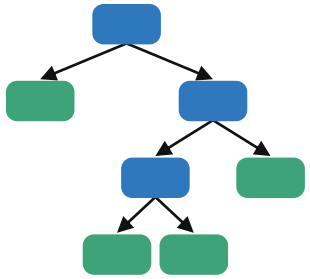
Yes



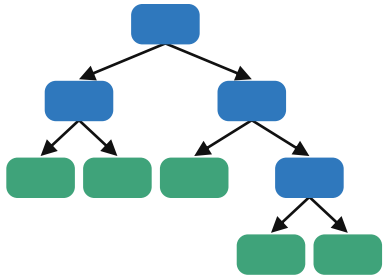
Yes



Yes



No



Yes

Votes from all trees

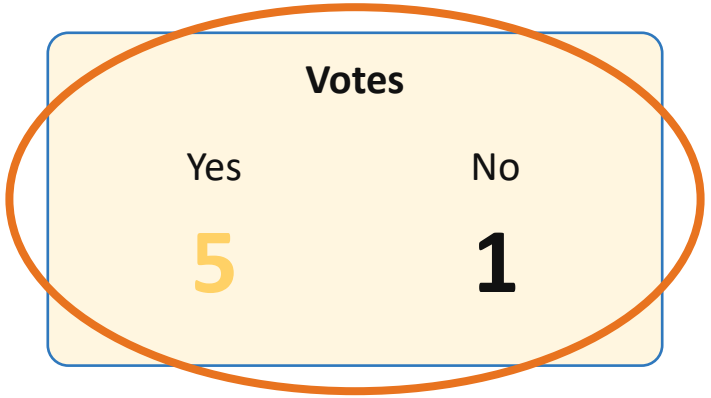
Yes	No
5	**1**

Each tree casts one vote for the new record.

The Majority Vote Decides

New record, now classified

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	No	No	168	**YES**



“Yes” received the most votes (5 against 1), so we predict that this flight will be delayed.

Terminology: Bagging



Bootstrap
the data

+

aggregate
the votes of the trees

=

Bagging
(bootstrap aggregating)

Where random forests fit
A random forest is bagging applied to decision trees, with one extra ingredient: each split may only use a random subset of the variables.

Records Left Out of a Bootstrap

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped dataset

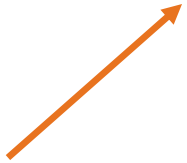
Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Remember the bootstrapped dataset? Typically, about one third of the original records do not end up in it.
Here, record 3 was never drawn, so it was not included.

The Out-of-Bag Dataset

Original dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

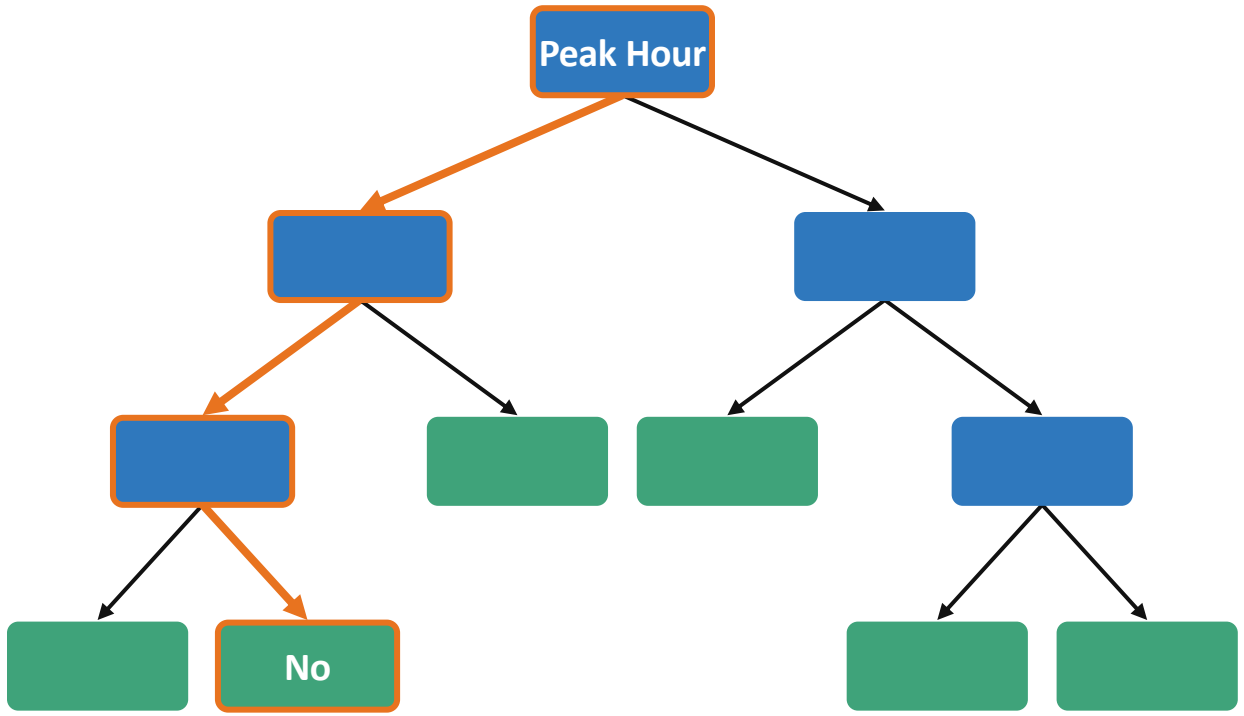


Out-of-bag dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	No	210	No

The records left out of a bootstrapped dataset form its **out-of-bag dataset**.
They were not used to build that tree, so they can be used to test it.

Testing a Tree on an Out-of-Bag Record



Out-of-bag record (true class: No)

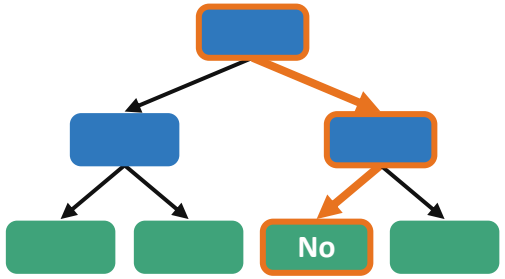
Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	No	210	No

Predicted: No

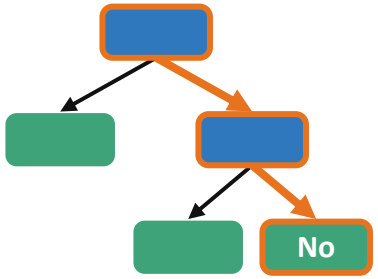
Matches the true class, so this tree classifies the out-of-bag record correctly.

Run the out-of-bag record through a tree that was built without it, and see whether it is classified as “Not delayed”.

Run It Through Every Tree Built Without It



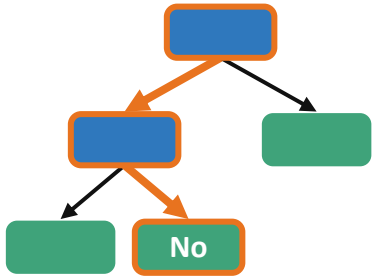
Votes: No



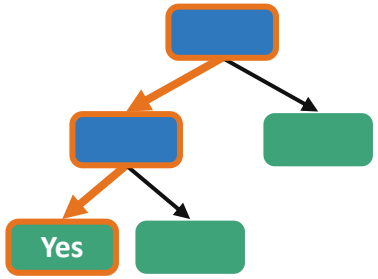
Votes: No

Out-of-bag record

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	No	210	No



Votes: No



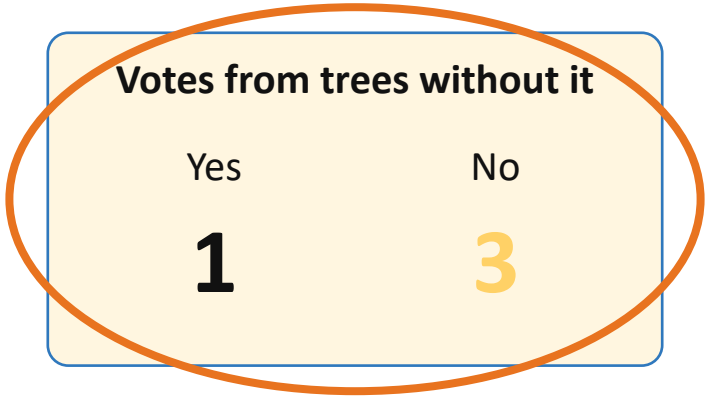
Votes: Yes

The same record goes down all the trees that were built without it. Most of them say “No”, but not all.

Majority Vote for the Out-of-Bag Record

Out-of-bag record (true class: No)

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	No	210	No



The label with the most votes wins, so this record is classified as **“No”**, which is its true class: a correct classification.

Repeat for Every Out-of-Bag Record

Out-of-bag record 1

Votes	
Yes	No
1	3

Classified as No

Out-of-bag record 2

Votes	
Yes	No
4	0

Classified as Yes

Out-of-bag record 3

Votes	
Yes	No
3	1

Classified as Yes

etc… etc… etc…

We then do the same thing for all the other out-of-bag records, for all of the trees.

Out-of-Bag Error

Accuracy = proportion of out-of-bag records correctly classified by the forest

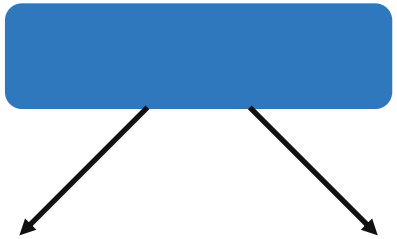
Out-of-bag error = proportion of out-of-bag records incorrectly classified = 1 – accuracy

Example
If 90 of 100 out-of-bag records are classified correctly, the accuracy is 0.90 and the out-of-bag error is 0.10.

Why it is useful
Each record is judged only by trees that never saw it, so no separate test set is needed. The error can also guide how many variables to consider at each step.

Remember: Only Two Variables per Step

Remember when we built our first tree and considered only 2 variables (columns of data) at each step?



Bootstrapped dataset

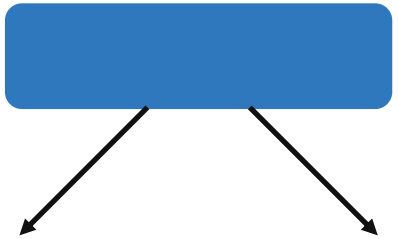
Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

A different random pair of columns is drawn at every step of every tree.

Comparing Out-of-Bag Error

Now we can compare the out-of-bag error of a forest built using only 2 variables per step…

…with the out-of-bag error of a forest built using 3 variables per step, and so on.

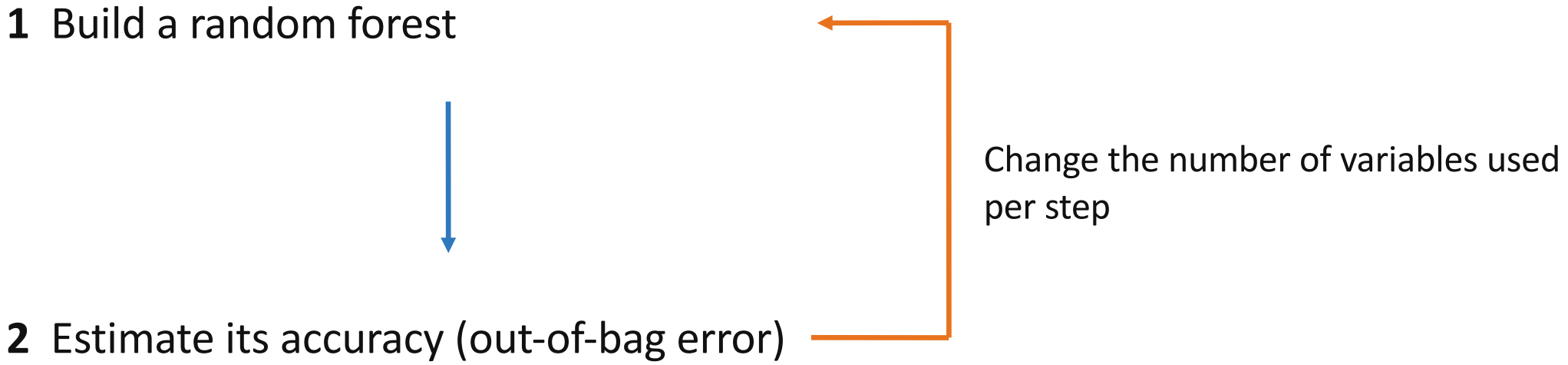


Bootstrapped dataset

Bad Weather	Peak Hour	Late Inbound Aircraft	Passengers	Delayed
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Each setting gives one out-of-bag error. The setting with the lowest error is the best choice.

Choosing the Number of Variables per Step



Rule of thumb
Start with the square root of the number of variables (e.g, for 16 variables, 4), then try a few settings above and below that value. Keep the setting with the lowest out-of-bag error.

References

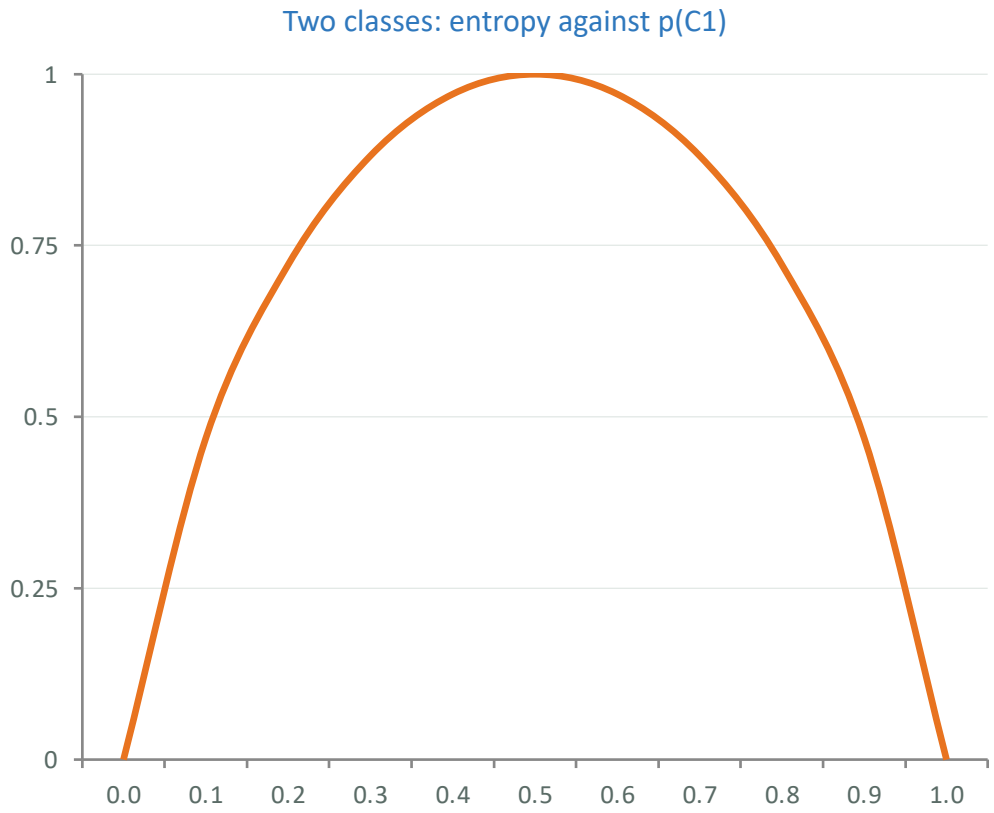
- **Hastie, Tibshirani and Friedman, *The Elements of Statistical Learning* (2nd ed., 2012)**
  - Section 9.2, Tree-Based Methods: regression and classification trees, impurity measures, pruning and the known weaknesses of trees.
  - Section 8.7, Bagging: bootstrap aggregation and why averaging reduces variance.
  - Chapter 15, Random Forests: the definition, out-of-bag samples and error, variable importance, and the effect of the number of variables tried at each split.
  - Sections 7.10 and 7.11 (optional): cross-validation and the bootstrap, as the background for out-of-bag estimation.

- **scikit-learn User Guide: Decision Trees**
  - scikit-learn.org/stable/modules/tree.html (hands-on practice, including minimal cost-complexity pruning)

Extra: Entropy (Lower is Better)

$$Entropy(t) = -\Sigma_{j=1..n}\ p_j \log_2 p_j$$

- p(j), the term in the sum, is the relative frequency of class j at node t; n is the number of classes
- Maximum = $\log_2$ n: records equally distributed among all classes (impure)
- Minimum = 0: all records belong to one class (pure)
- Interpreted as uncertainty, or the level of randomness, of the class at the node



Peak of 1 bit when p = 0.5; zero when the node is pure.

Extra: Computing Entropy: Three Nodes

C1: 0
C2: 6

P(C1) = 0/6 = 0, P(C2) = 6/6 = 1

Entropy =
$- 0 \log_2 0 - 1 \log_2 1$

Pure

C1: 1
C2: 5

P(C1) = 1/6, P(C2) = 5/6

Entropy =
$- (1/6) \log_2(1/6) - (5/6) \log_2(5/6)$

Mostly pure

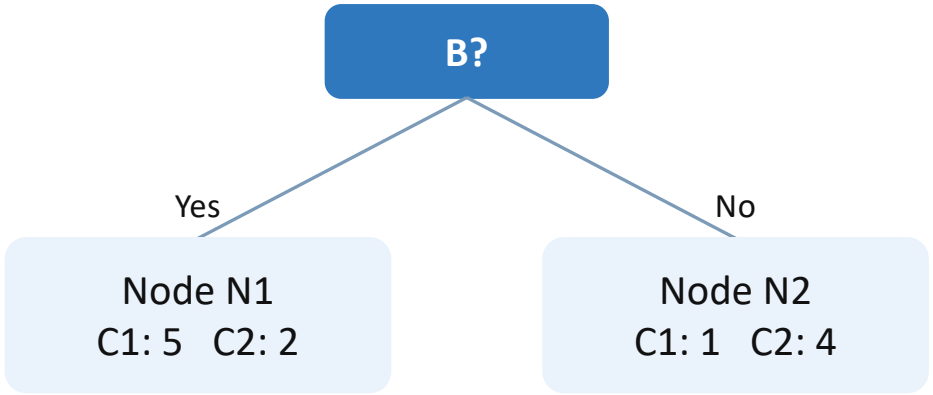
C1: 3
C2: 3

P(C1) = 3/6, P(C2) = 3/6

Entropy =
$- (1/2) \log_2(1/2) - (1/2) \log_2(1/2)$

Maximally impure

Extra: Entropy (Binary Attributes)



Parent: C1 = 6, C2 = 6

Entropy = $- 0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Entropy(N1) = $- (5/7) \log_2(5/7) - (2/7) \log_2(2/7) = 0.863$

Entropy(N2) = $- (1/5) \log_2(1/5) - (4/5) \log_2(4/5) = 0.722$

Entropy(children) = $(7/12) \times 0.863 + (5/12) \times 0.722 = 0.804$

Information Gain = 1 – 0.804 = 0.196

- Splits into two partitions
- Effect of weighting partitions: larger and purer partitions are sought