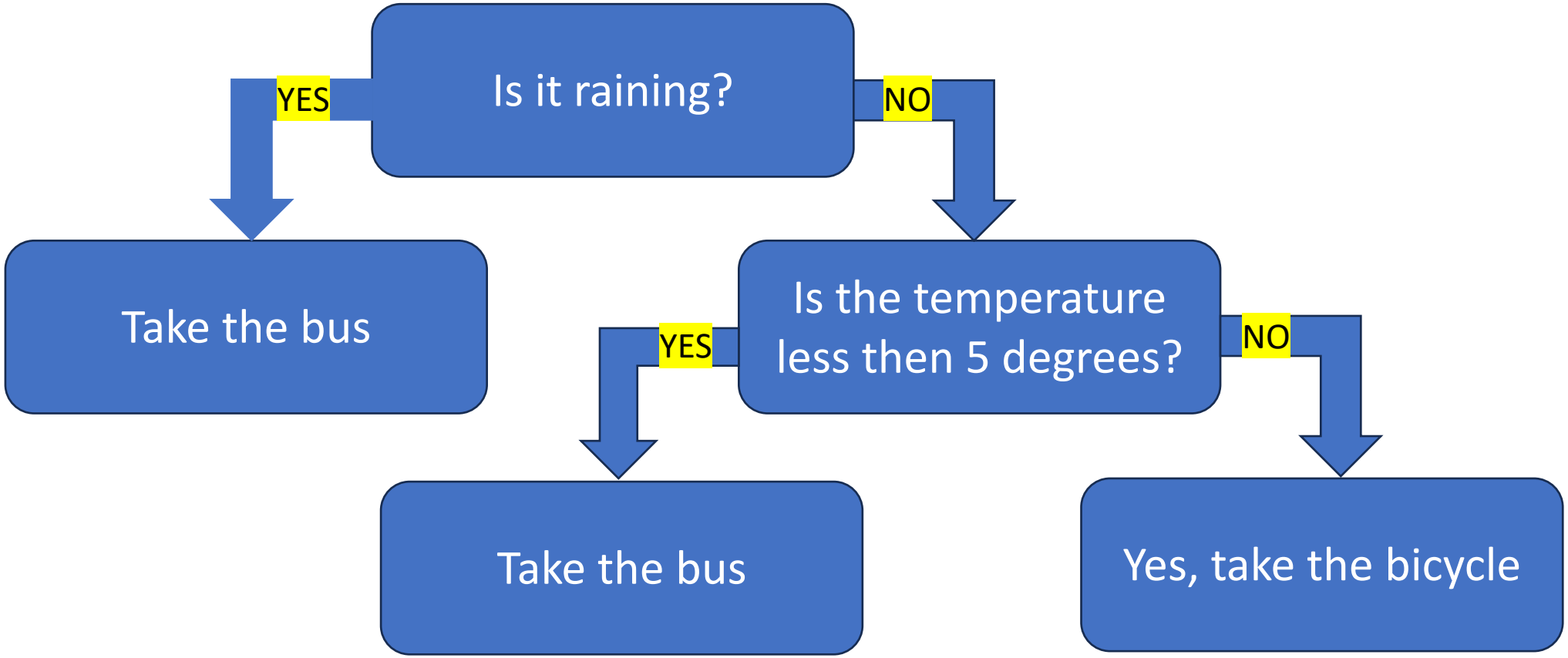
Utrecht University

Lecture 4: Tree Based Methods

Utrecht Summer school on
Machine Learning with Python

July 2024

Should I bike to the Univeristy?



So far

- Main concepts in machine learning
- Model fit
- Data splits and feature engineering
- Linear regression
- Classification models and classification evaluation

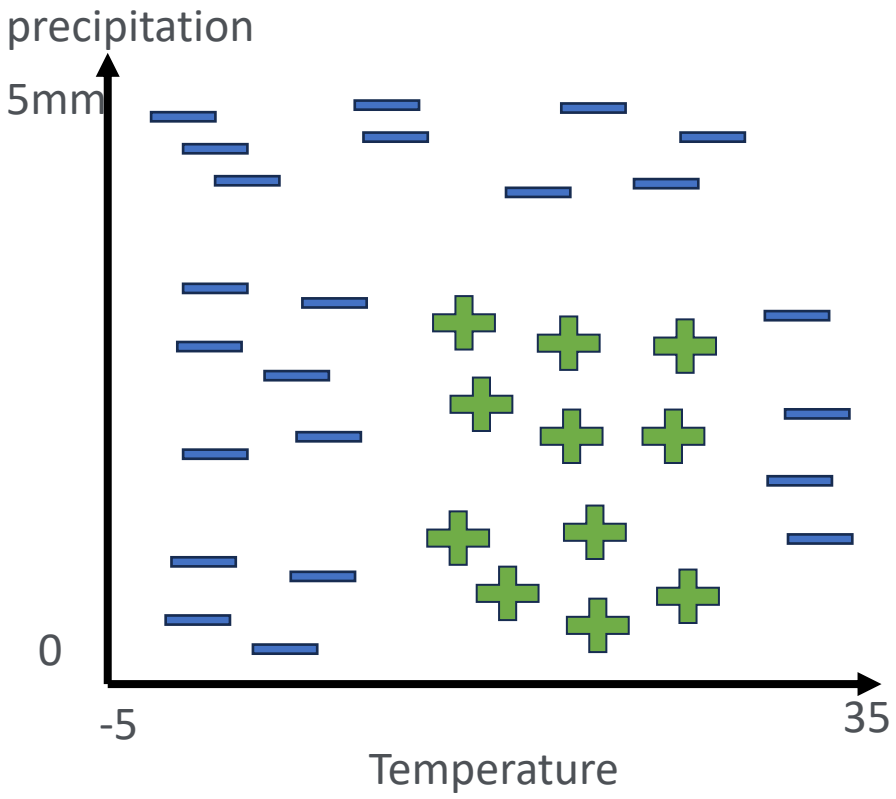
Main points of this lecture

- Tree-based methods: segment predictor space into a number of simple regions
- Using decision trees for prediction
- Improvement 1: Bagging
- Improvement 2: Random forests
- Approximate the test error using the Out of Bag (OOB) error
- Variable importance measures
- Conclusions

Decision Trees

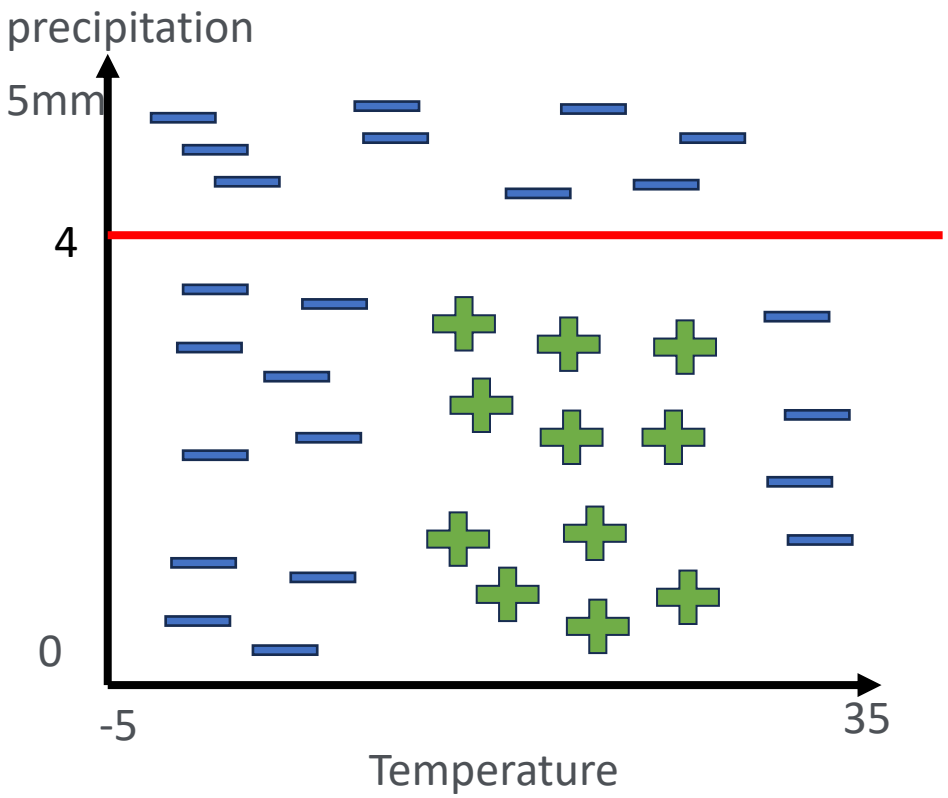
Basics of Decision Trees

- Shall we go to a picnic?



Basics of Decision Trees

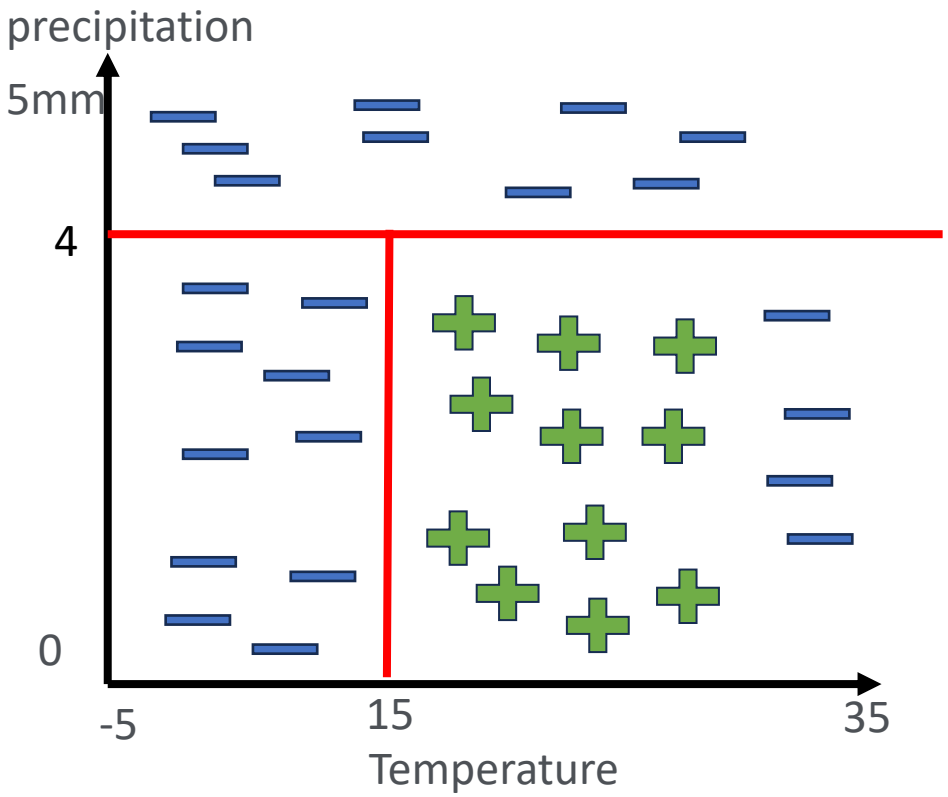
- Shall we go to a picnic?

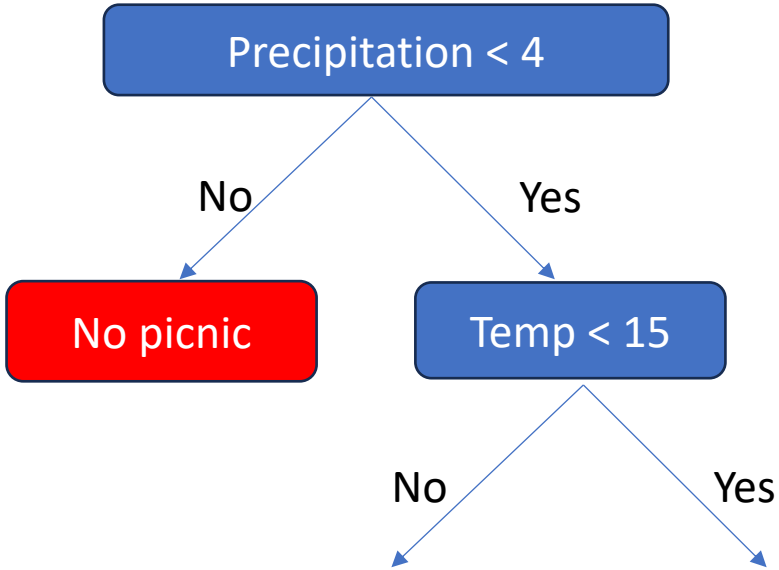


Precipitation < 4

Basics of Decision Trees

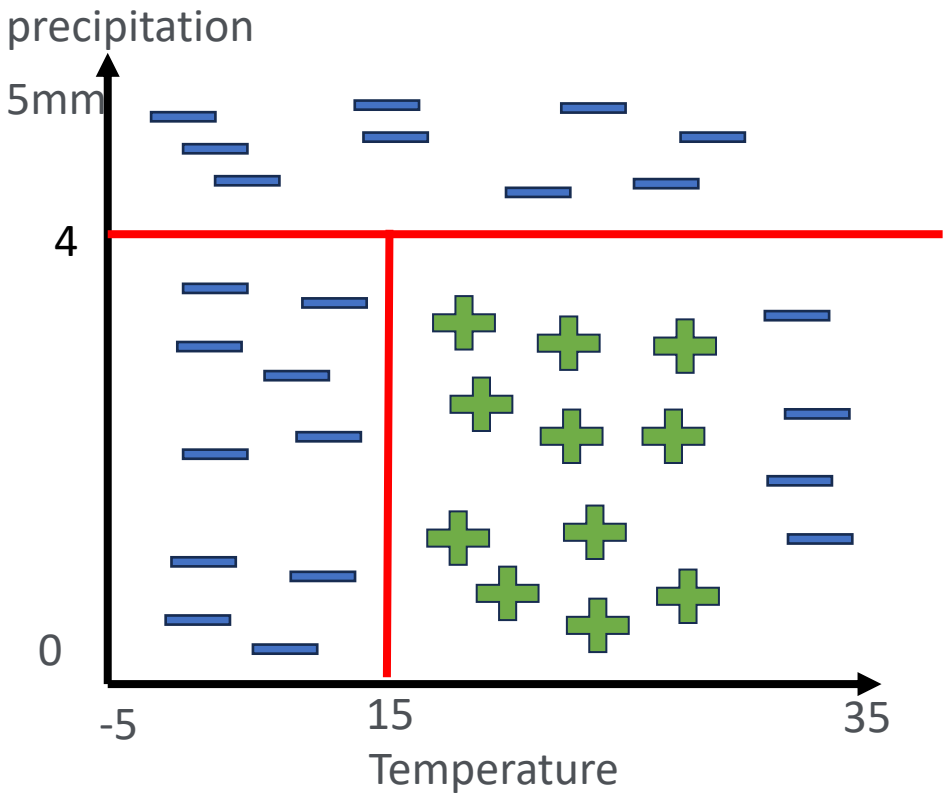
- Shall we go to a picnic?

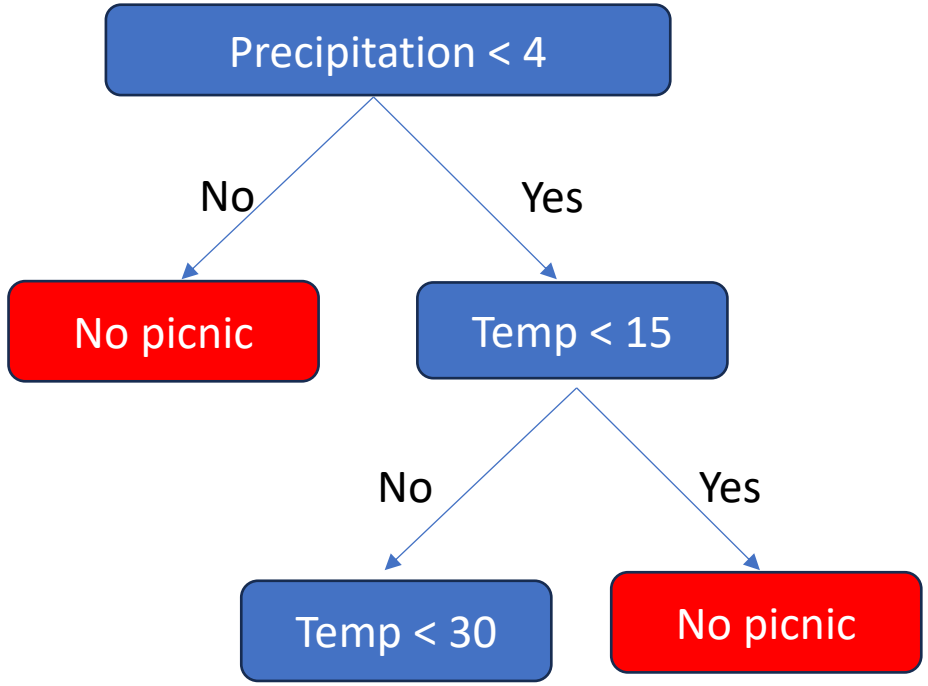




Basics of Decision Trees

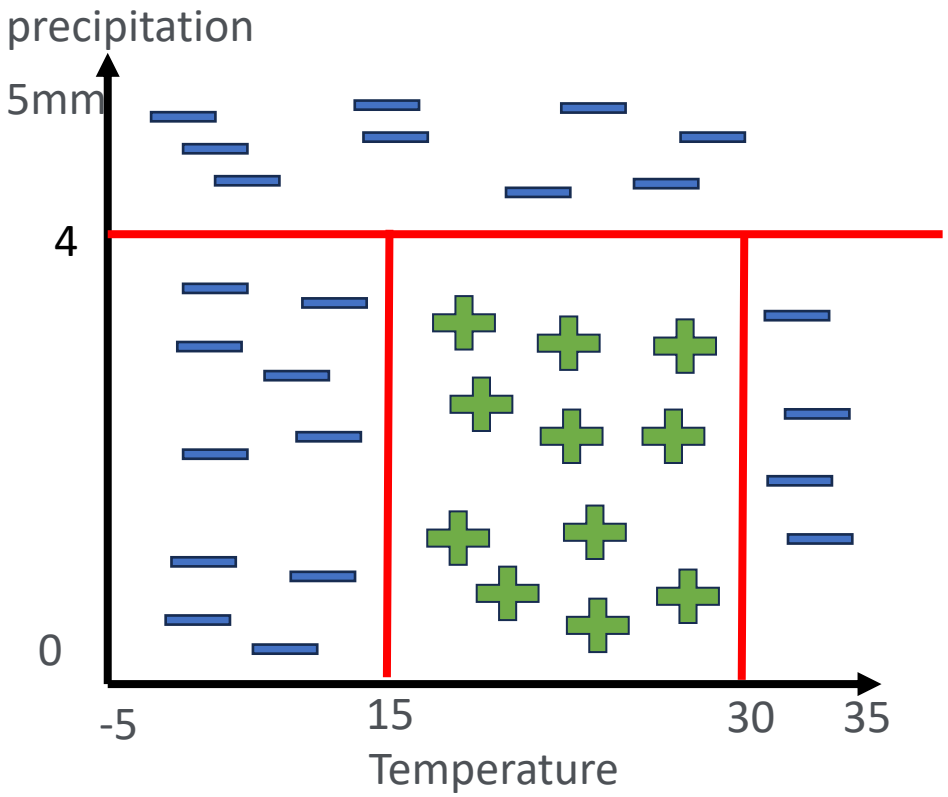
- Shall we go to a picnic?

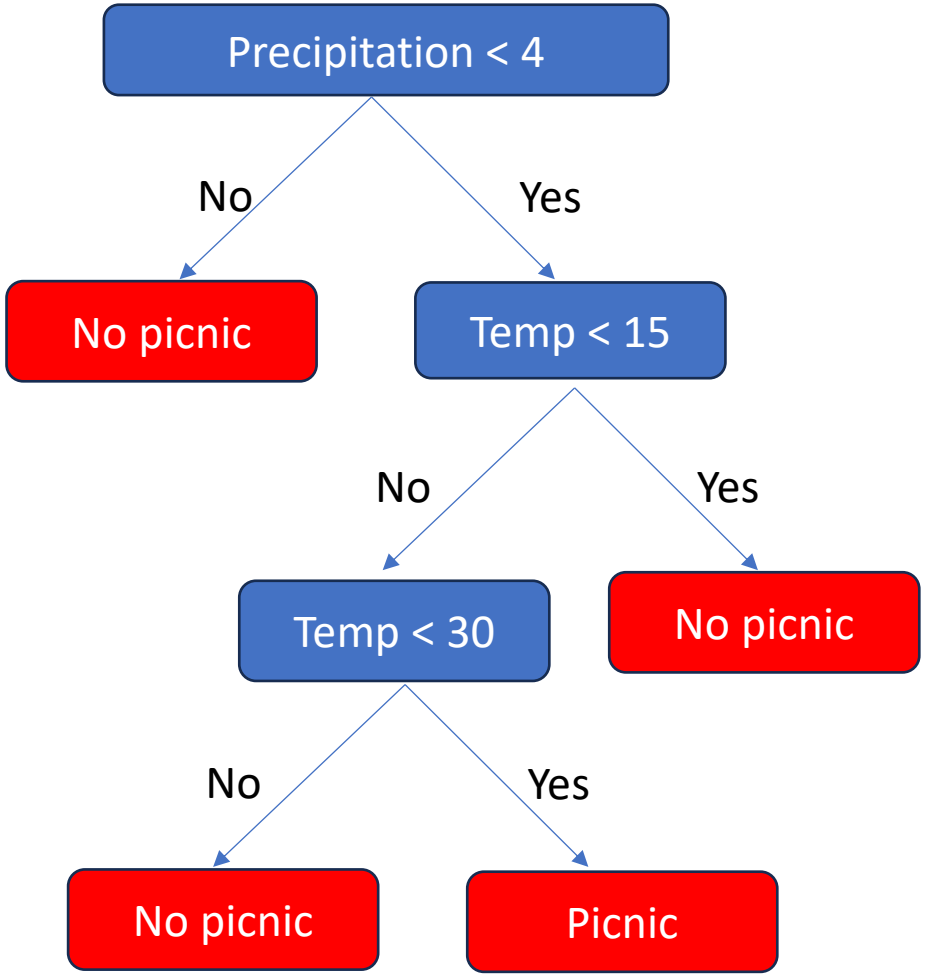




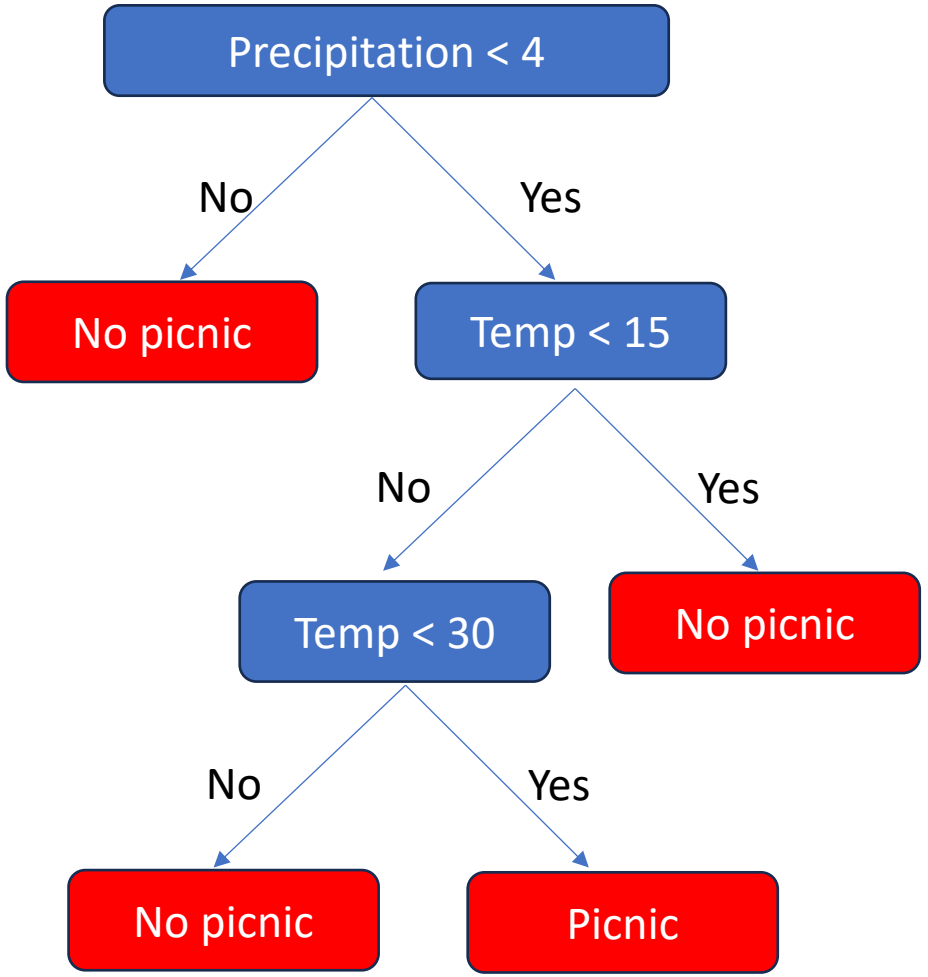
Basics of Decision Trees

- Shall we go to a picnic?





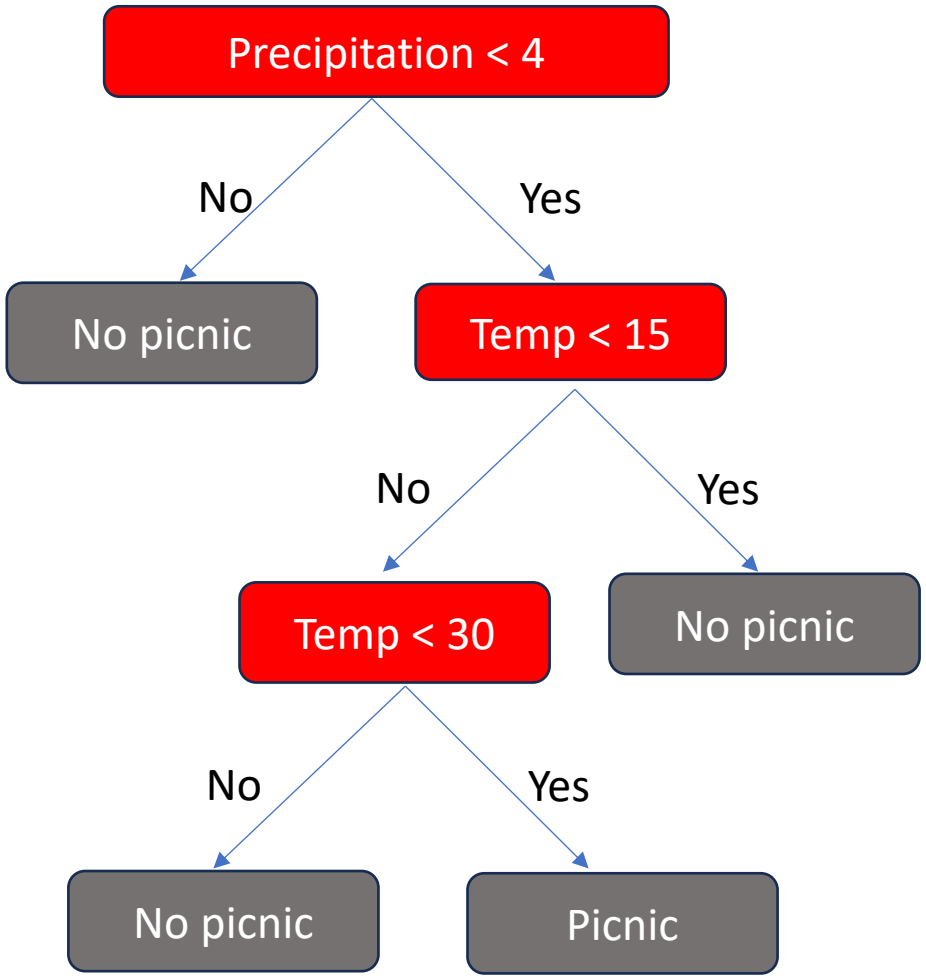
Basics of Decision Trees



Features:
x1: precipitation
x2: temparature

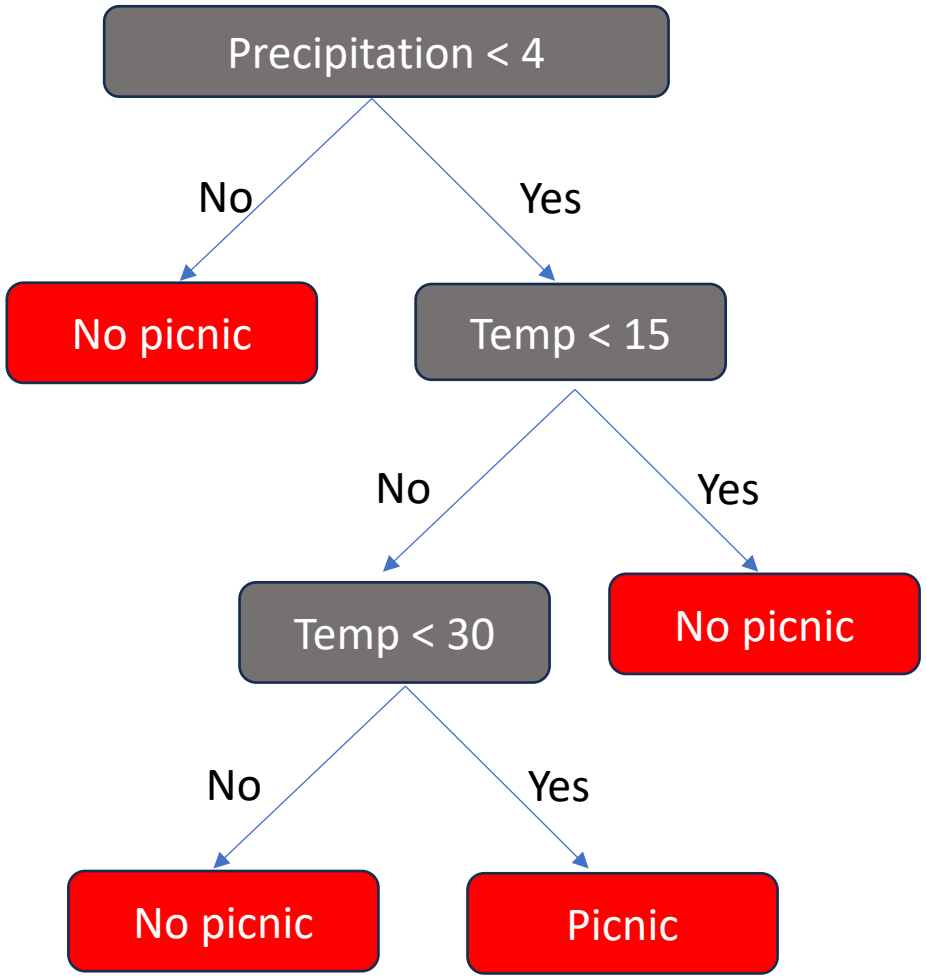
labels y:
1: picnic
0: no picnic

Basics of Decision Trees



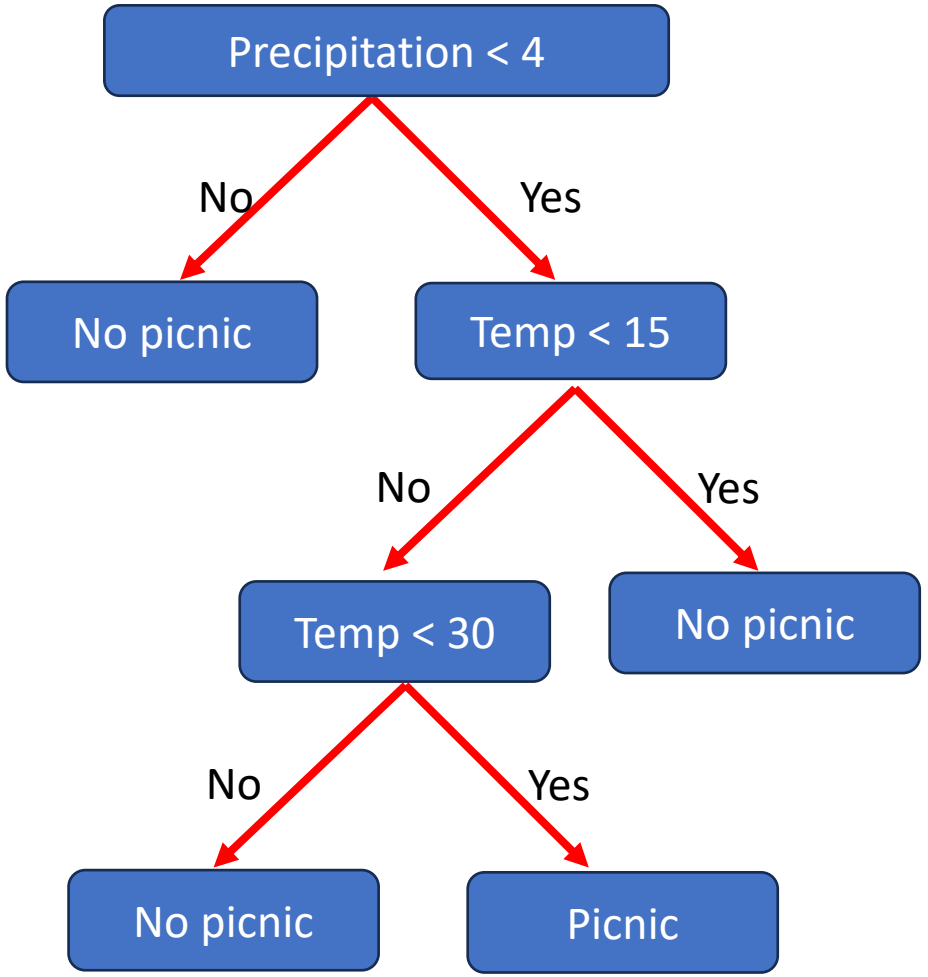
- Internal nodes:
- Two children that can be:
    - internal node

Basics of Decision Trees



- Internal nodes
- Two children that can be:
    - internal node
    - leaves or terminal node

Basics of Decision Trees



- Internal nodes
- Two children that can be:
    - internal node
    - leaves or terminal node
- Connections with branches

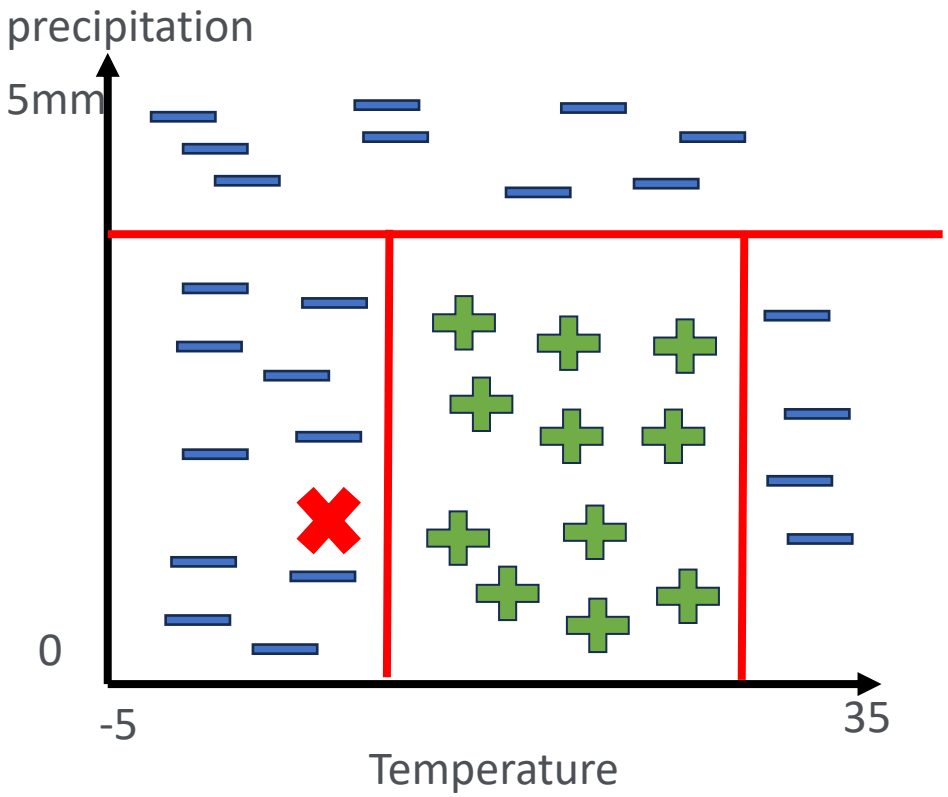
Build a decision tree

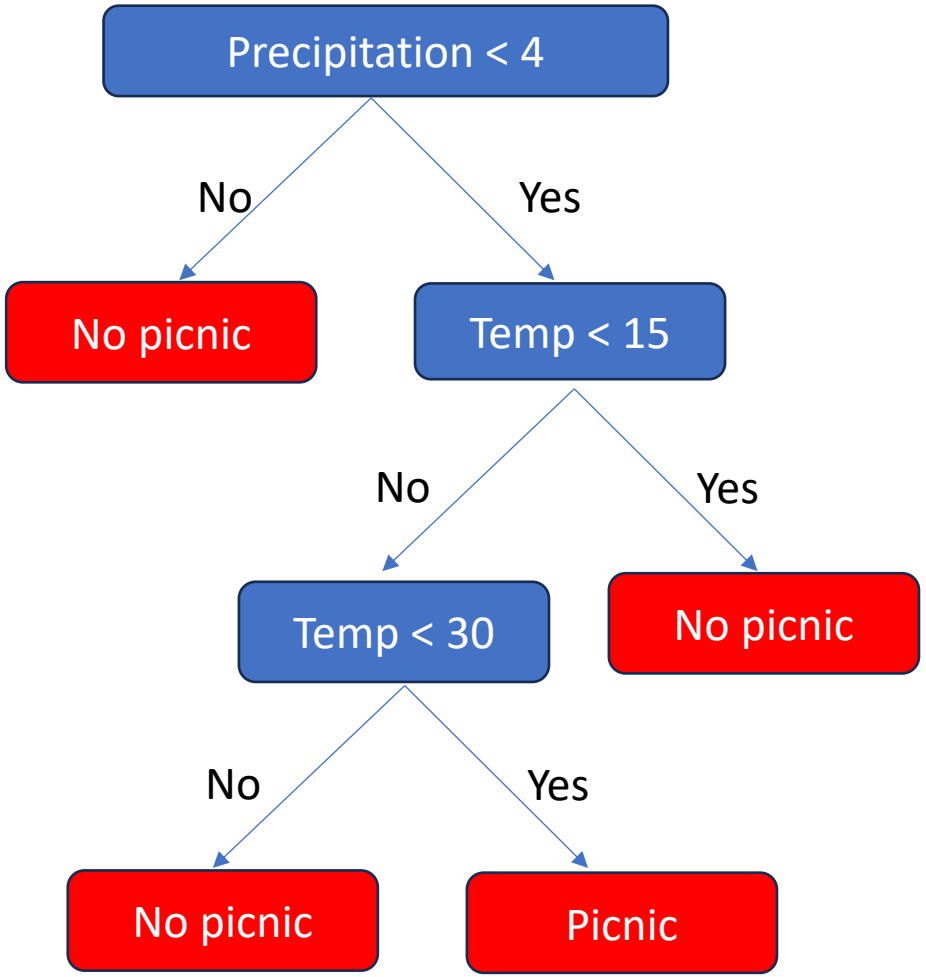
1.Divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ — into **J** distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.

2. For every observation that falls into the region R_j , we make the same prediction, which is simply the majority class of the outcome for the training observations in R_j .

Basics of Decision Trees

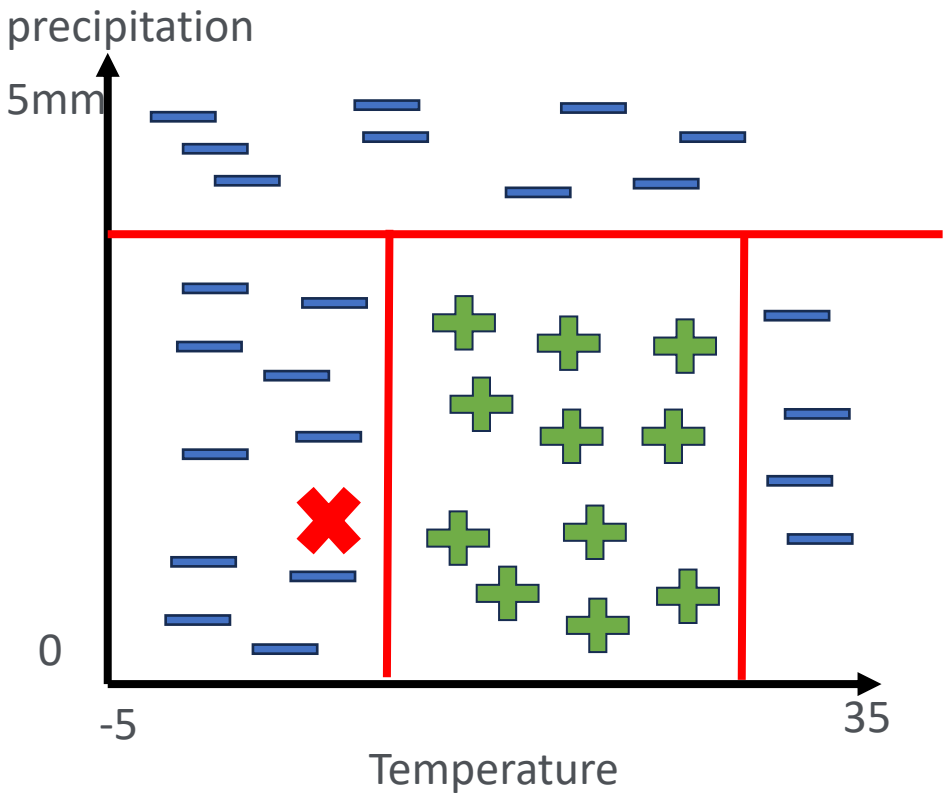
- Shall we go to a picnic?

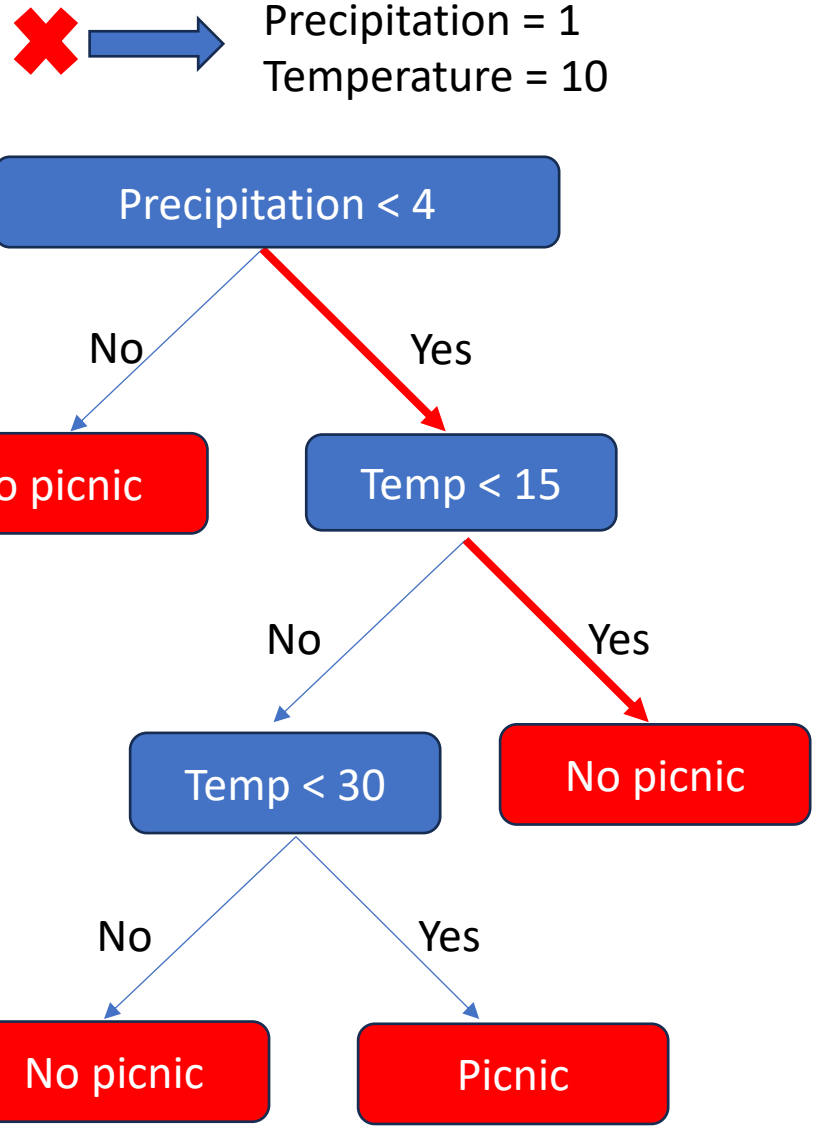




Basics of Decision Trees

- Shall we go to a picnic?





Growing decision trees from data

- Recursive (binary) partitioning:
    1. Find the split that makes observations as similar as possible on the outcome within that split;
    2. Within each resulting group, do (1).

- Early stopping: add after (2):
- ‘unless there are fewer than n_{min} observations in the group’ (typically 10)

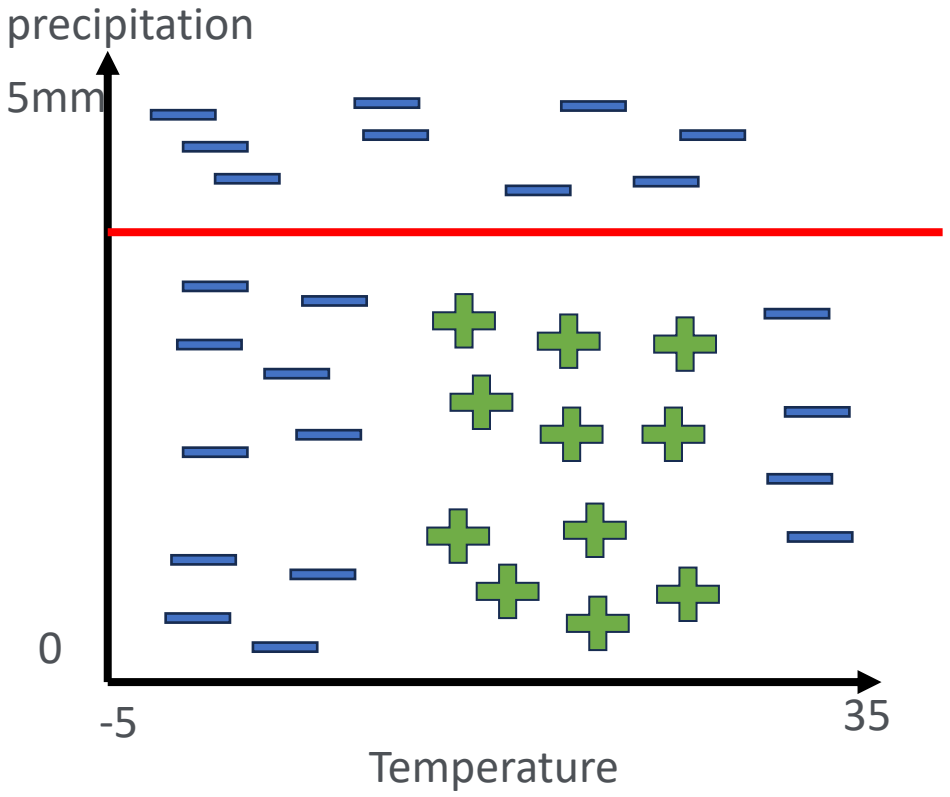
Growing decision trees from data

- Criteria for ‘as similar as possible’: reduction in classification error rate such as the **Gini impurity** or **entropy**

- Gini impurity: $1-\sum_{i=1}^{n}(P_i)^2$

- where p_i is the proportion of training observations in partition with category *i*
- Small value: almost all values in the partition belong to one class
- Hence, Gini index is a measure of node *impurity*: small value indicates that a partition contains predominantly observations from a single class

Gini impurity



$$1-\sum_{i=1}^{n}(P_i)^2$$

For the upper part:
$1 -(9/9)^2 – (0/9)^2 = 1 – 1 – 0 = 0$

For the lower part:
$1 – (14/25)^2 – (11/25)^2 = 1 – 0{,}3136 – 0{,}1936 = 0{,}4928$

Weighted average impurity:
$(9/34 * 0 + 25/34 * 0{,}4928)/2 = 0{,}18$

Choosing Features

- Key Idea: good features partition the data into subsets that are either “all positive” or “all negative” (ideally)

Tree building: top-down and greedy

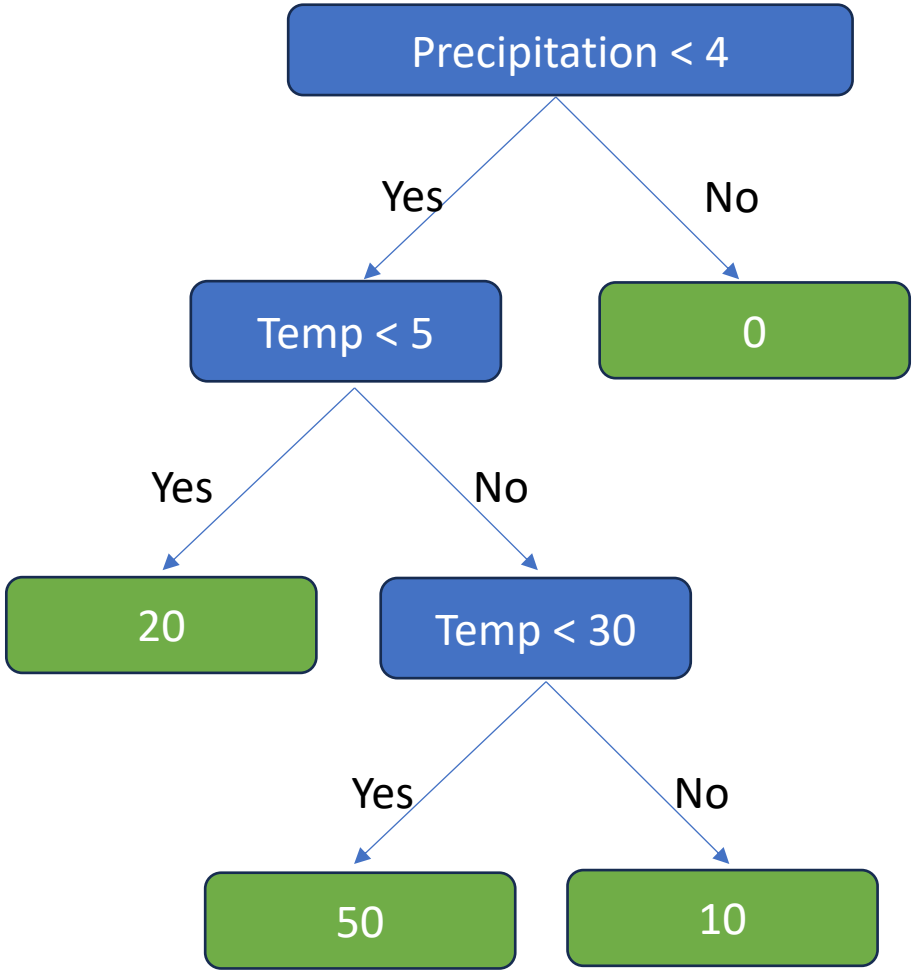
- Recursive (binary) partitioning is a top-down and greedy algorithm:
- **Top-down**: algorithm begins at the top of the tree and then successively splits the predictor space. Each split is indicated via two new branches further down on the tree.
- **Greedy**: at each step, the best split for that step is made, instead of looking ahead and picking a split that will result in the best tree in a future step.

Regression Trees

Regression trees

- Decision trees can be used for regression as well!
- Instead of predicting class label in each ‘box’ (partition), we predict the outcome in each partition:
  - mean of the training observations in the partition to which the test observation belongs
- Cutpoints are selected such that splitting the predictor space into the regions leads to the greatest possible reduction in residual sums of squares (RSS).

Regression trees

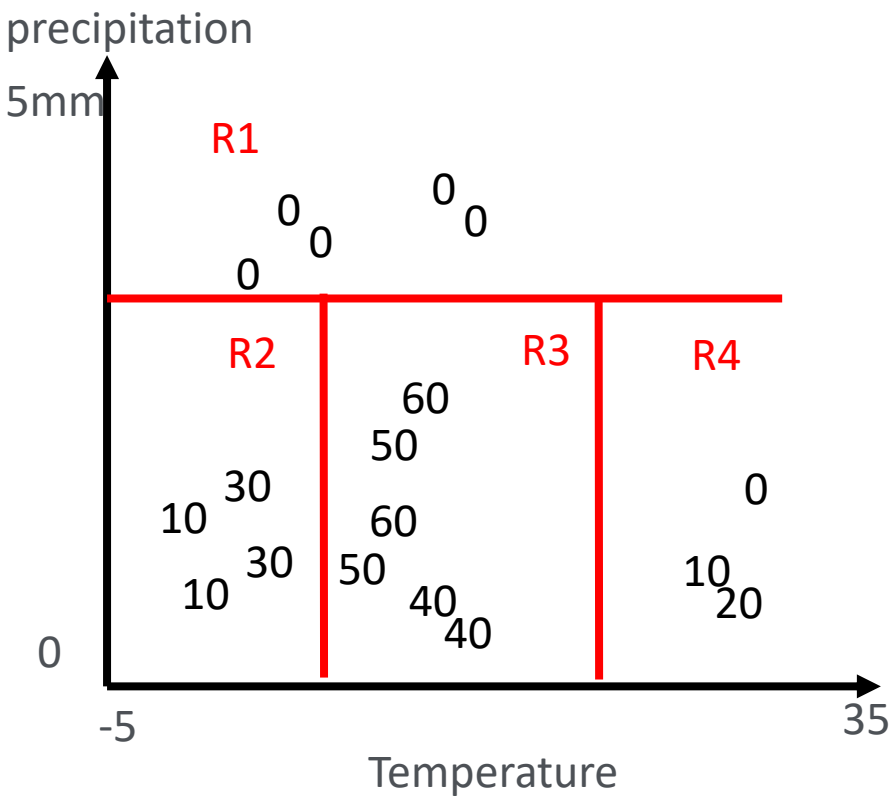


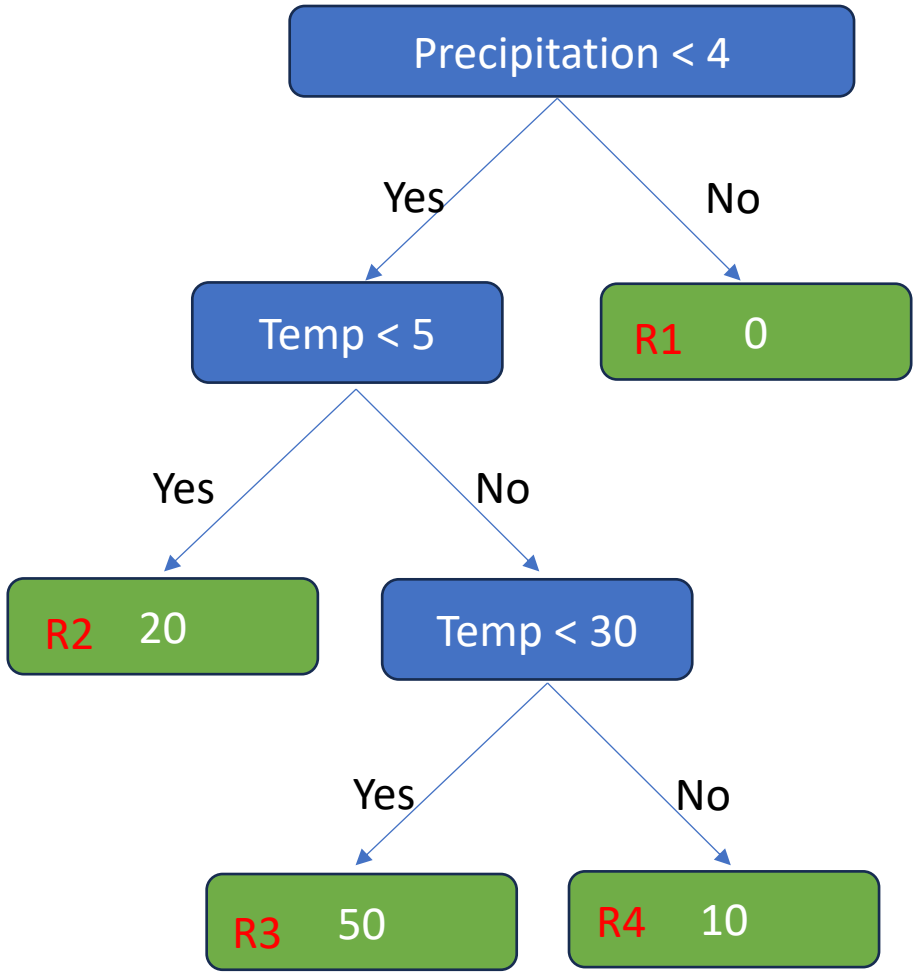
Features:
x1: precipitation
x2: temparature

y:

- minutes of outdoor training

Regression trees





How to construct regions?

- The goal is to find boxes R1, . . . ,RJ that minimize the RSS

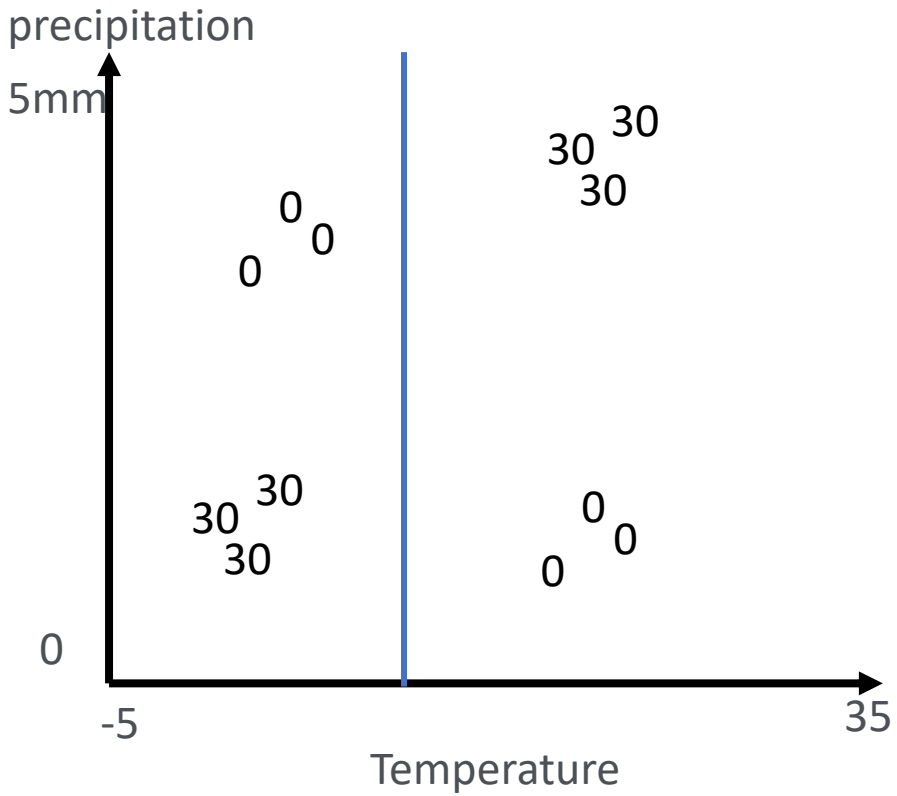
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

- computationally infeasible to consider every possible partition of the feature space into J boxes

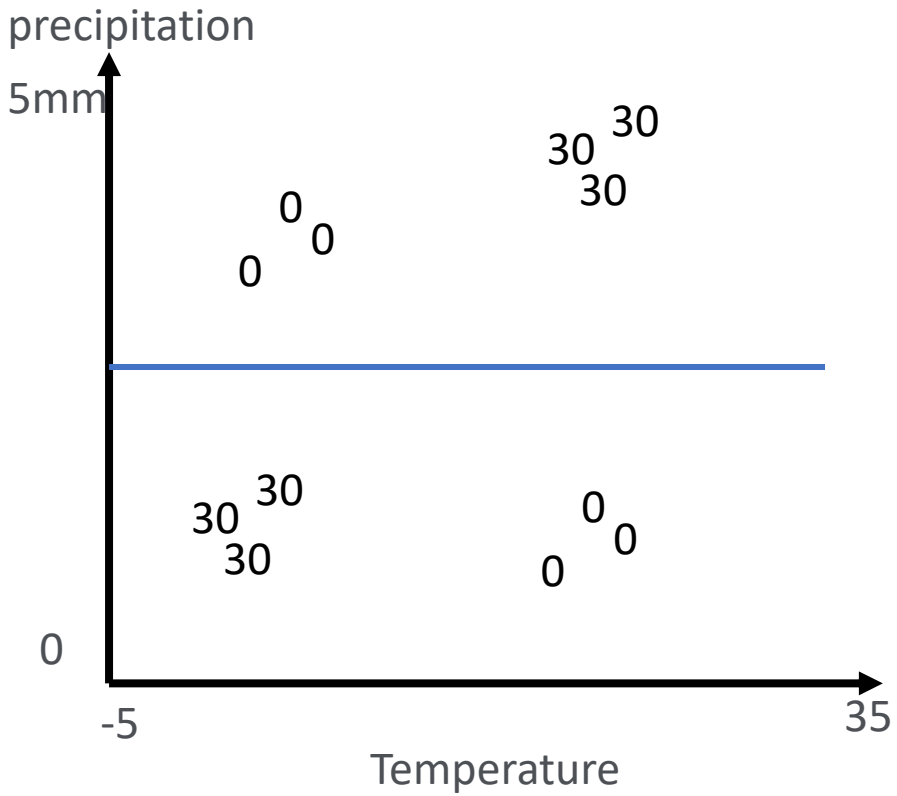
Pruning

- Process described to grow trees most likely overfits the data → poor test performance
- Solution 1: build the tree until the decrease in classification error / RSS exceeds some threshold
- This strategy will result in smaller trees
- However, a seemingly worthless split early on in the tree might be followed by a very good split

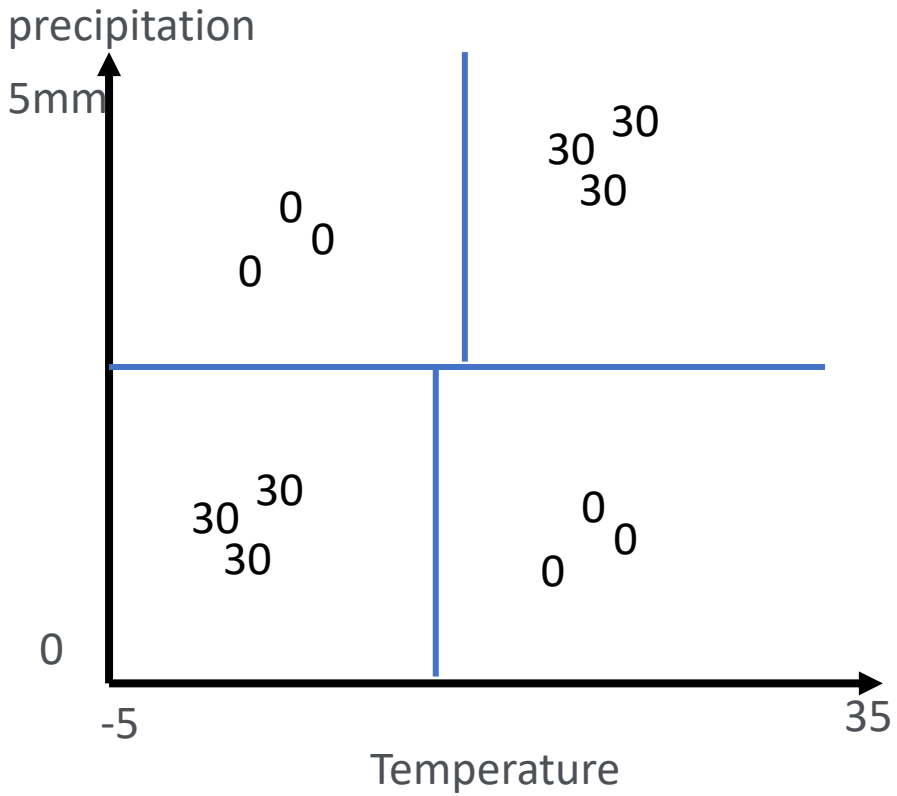
Pruning



Pruning



Pruning



Cost complexity pruning

- Grow a large tree and then prune it back

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- |T| is the number of terminal nodes
- R_m is the rectanclge that corresponds to the m^{th} terminal node
- The tuning parameter *α* controls a trade-off between the subtree's complexity and its fit to the training data

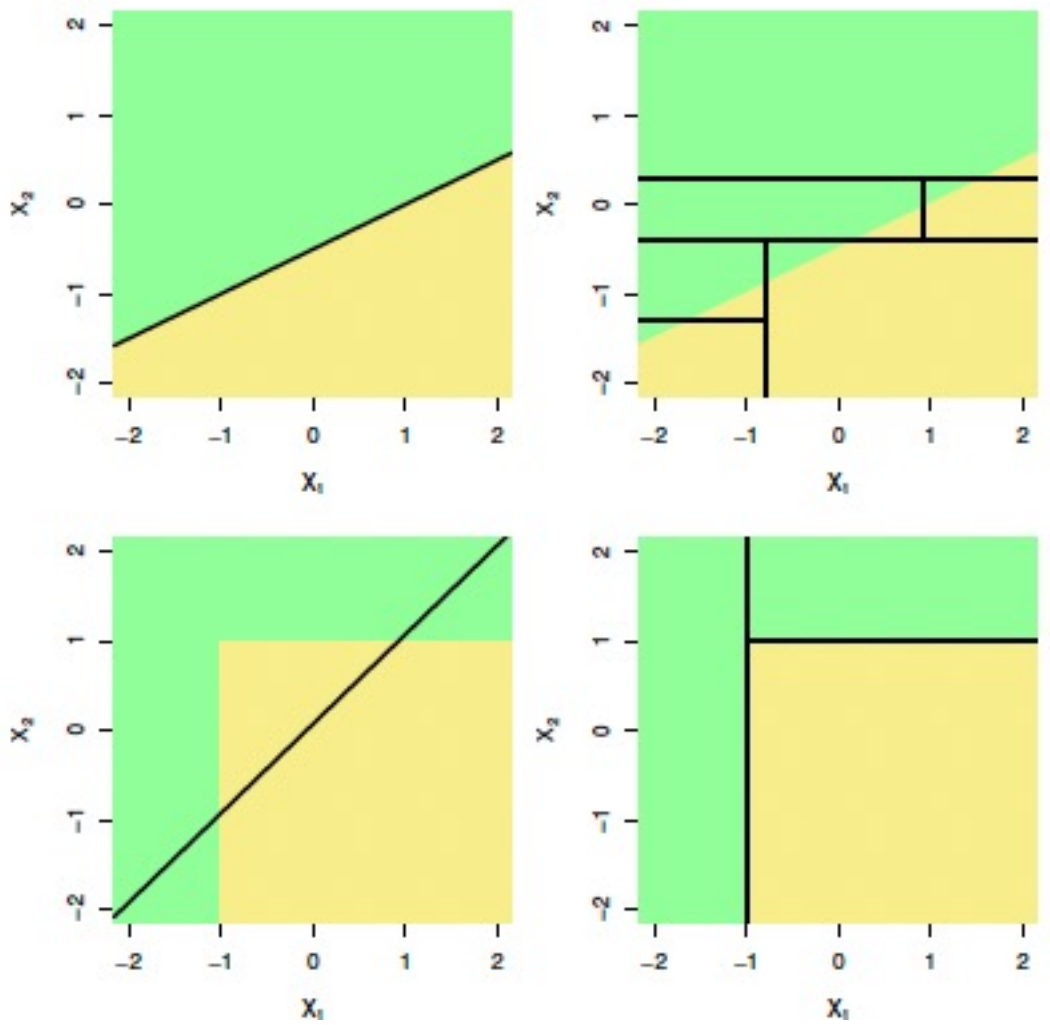
Cost complexity pruning

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- α=0 → subtree equals the original (long) tree
- As α increases, becomes more expensive to have a tree with many terminal nodes, resulting in a smaller subtree. Very similar to Lasso!
- Use K-fold cross-validation to choose α.

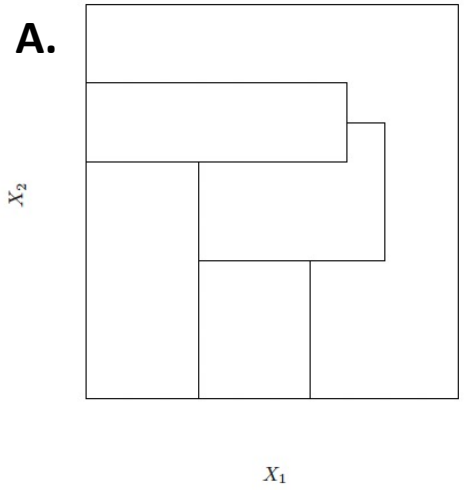
Linear Models VS Trees

- Which model is better?
- It depends on the problem

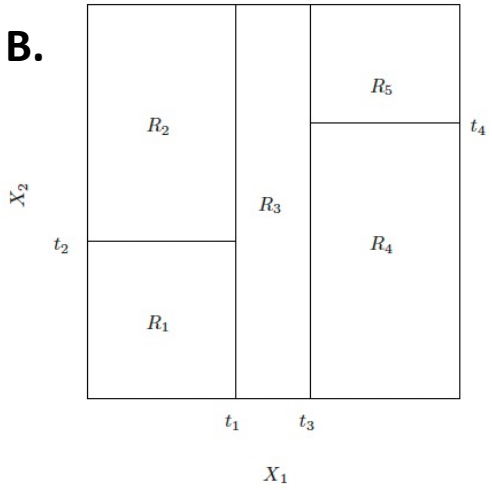


Which of the panel(s) in the graph below cannot be representative of a decision tree model using recursive binary splitting?

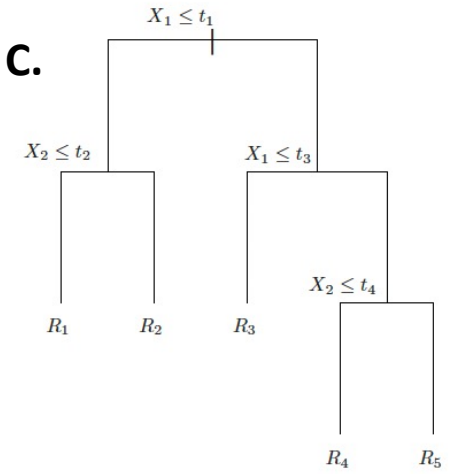
A.



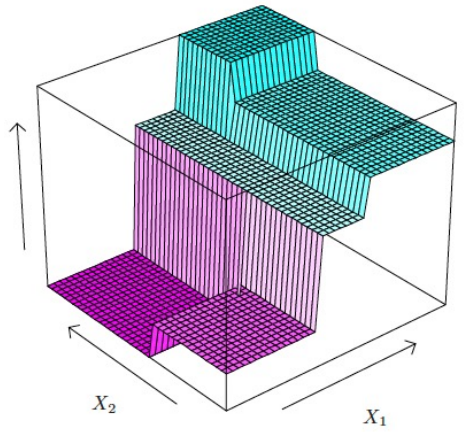
B.



C.



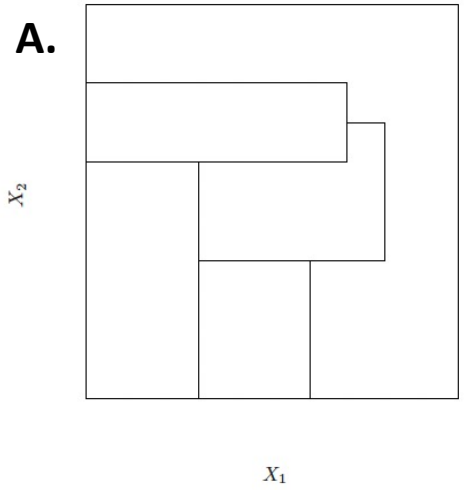
D.



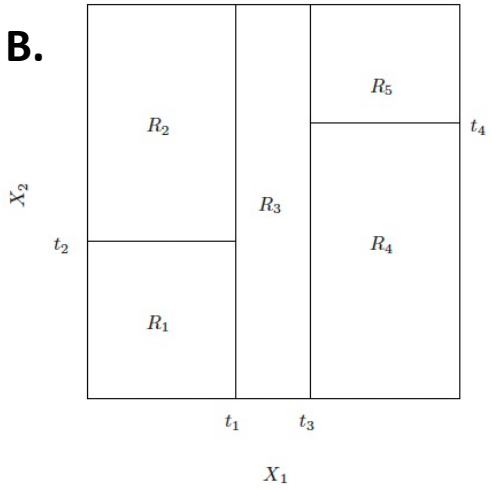
Which of the panel(s) in the graph below cannot be representative of a decision tree model using recursive binary splitting?

A can’t be representative of a decision tree

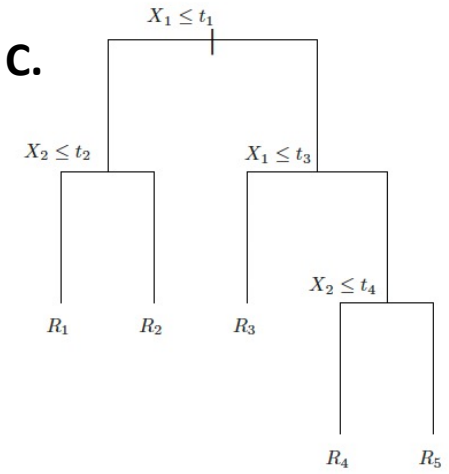
A.



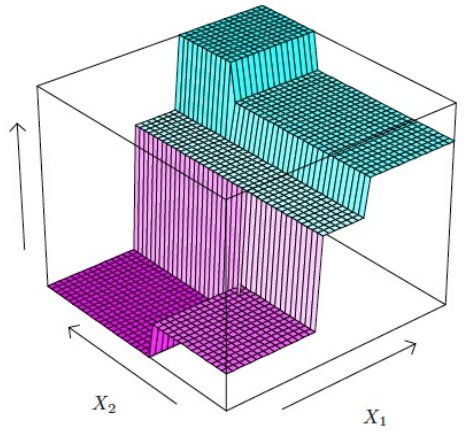
B.



C.



D.



Summary

- ✓Trees are very easy to explain and interpret
- ✓Mirror human decision-making better than other the regression and classification approaches
- ✓Trees can be displayed graphically (sometimes)
- ✓Detects non-linear relationships

Do not have the same level of predictive accuracy than other approaches

They are **prone to overfitting**: a small change in data can cause a large change in the final tree

Bagging and Random Forests

10 minutes break

Bagging

Intuition behind bagging

When you fiddle with the observations just a little:

1. Some things vary a lot;
2. Some things stay pretty much the same.

Intuition:

- Overfitting is caused by (1);
- but (1) happens randomly, causing predictions to go up or down haphazardly;
- Therefore, (1) should be cancelled out by fiddling with the observations a little and averaging
- “Wisdom of the crowds”

Bagging

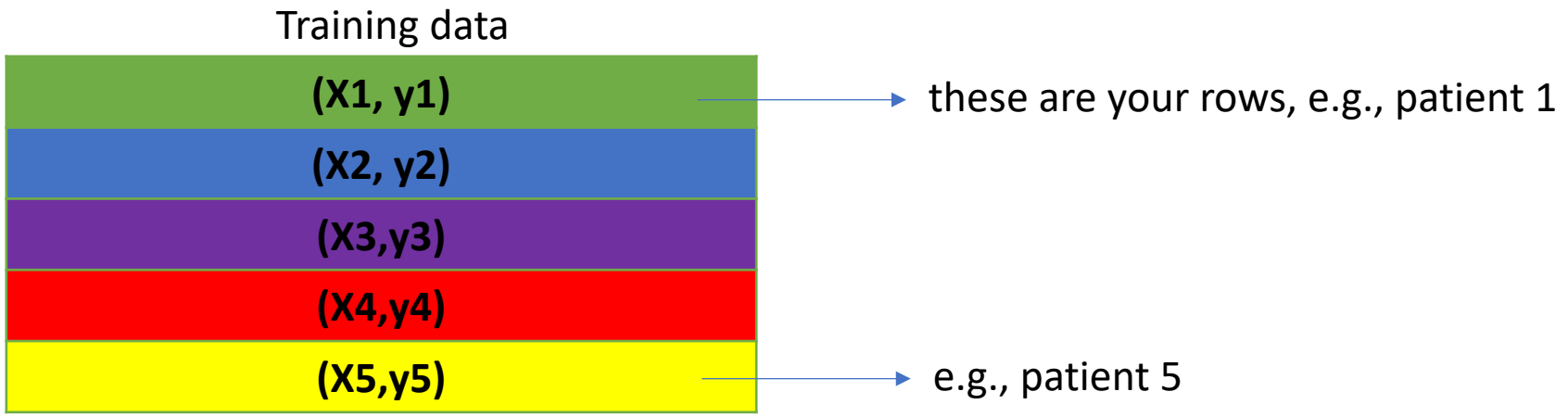
- *Bootstrap aggregation*, or bagging (Breiman 1994):
- A general-purpose procedure for reducing the variance of statistical learning methods. It is very useful and often applied in the context of decision trees
- Averaging a set of independent observations reduces variance. But what if we only have one training set?

Bagging

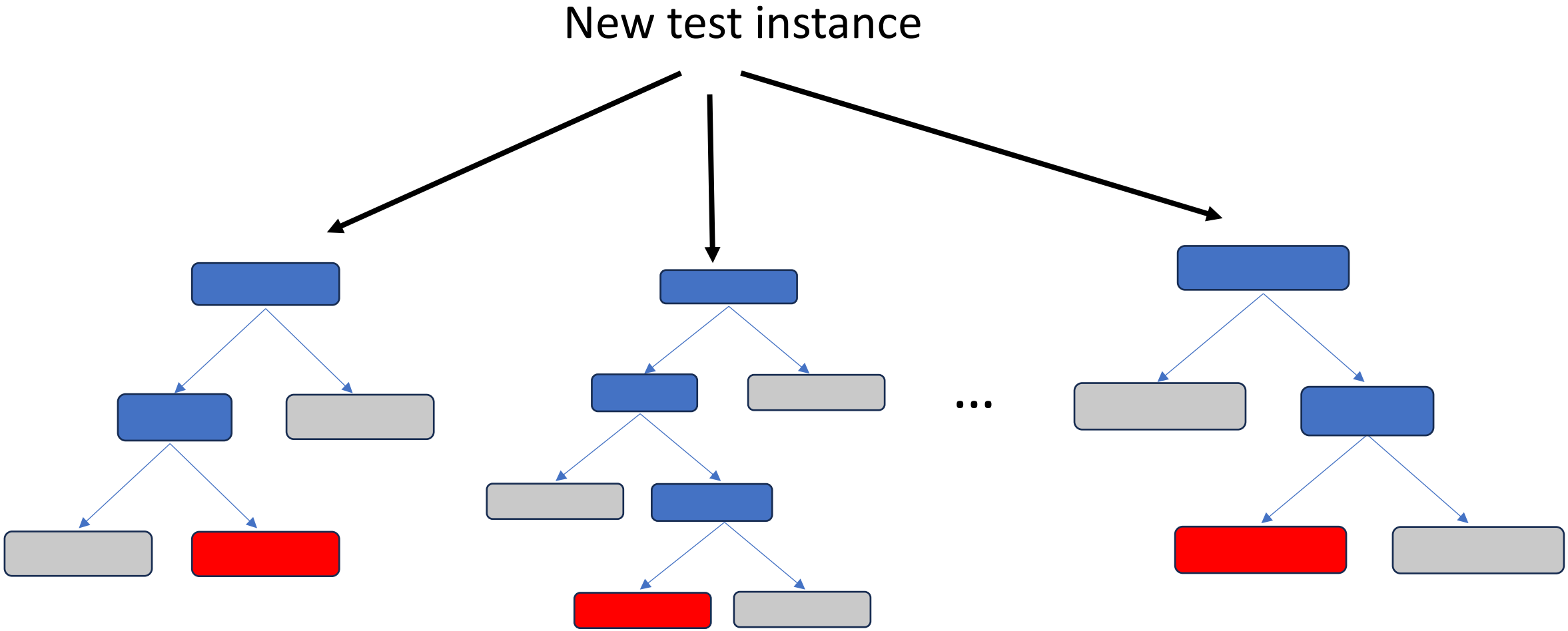
- We can mimic having multiple training sets by bootstrapping:
- Generate *B* different bootstrapped training data sets.
- Train a decision tree on each of the b^{th} bootstrapped training set to get a prediction for each observation *x:* $\hat{f}^{*b}(x)$

- For regression trees, we average all predictions to obtain: $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$

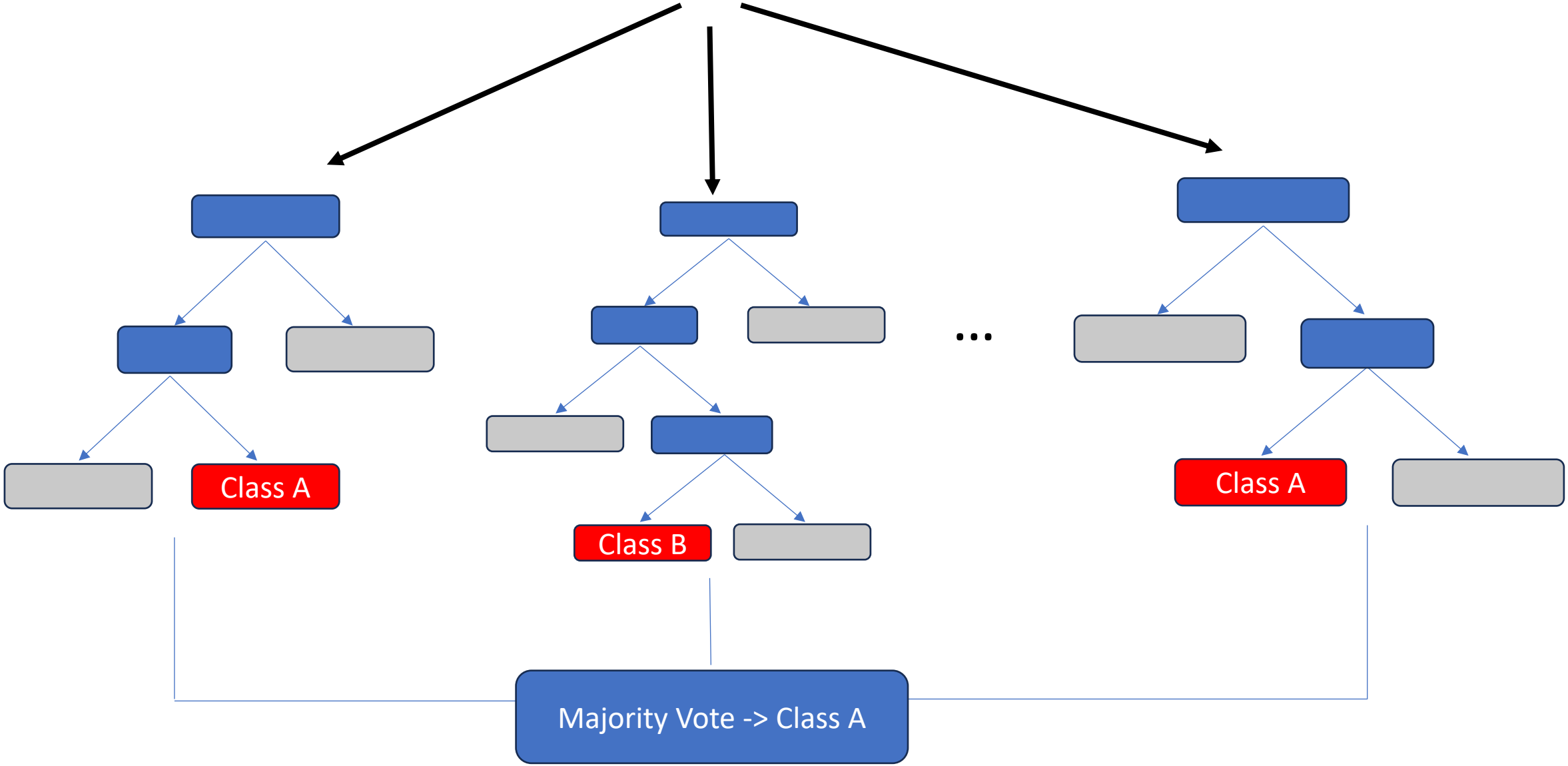
- For classification trees, we take the majority vote


Training data
(X1, y1)
(X2, y2)
(X3,y3)
(X4,y4)
(X5,y5)
these are your rows, e.g., patient 1
e.g., patient 5

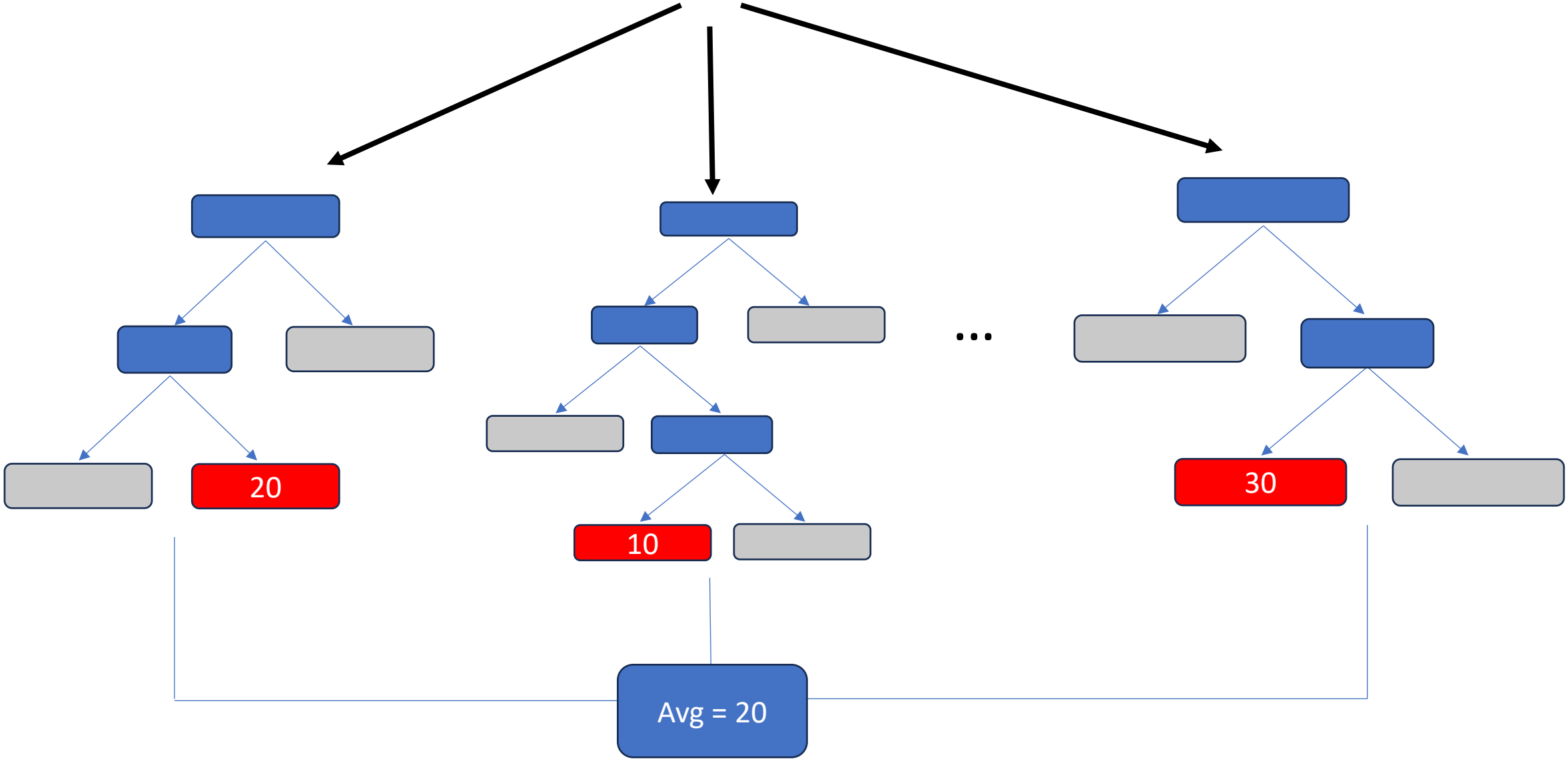

Training data
(X1, y1)
(X2, y2)
(X3,y3)
(X4,y4)
(X5,y5)
there are your rows, e.g., patient 1
e.g., patient 5
Bootstrap sample 1
(X1, y1)
(X2, y2)
(X5,y5)
(X4,y4)
(X5,y5)
Bootstrap sample 2
(X2, y2)
(X1, y1)
(X3,y3)
(X1, y1)
(X2, y2)
Bootstrap sample 3
(X4,y4)
(X2, y2)
(X3,y3)
(X4,y4)
(X1, y1)


New test instance
...

Classification: New test instance



Regression: New test instance



Random Forests

Random forests

- In bootstrapping, the samples taken are independent.
- But the predictions from trees grown on the bootstrapped samples are not independent!
- They share the same features and can therefore create similarly overfitting decision rules
- “Wisdom of crowds who peek at each other’s answers”
- This phenomenon is called “tree correlation” (Breiman 2001)

Random forests

- Random forests try to remove the tree correlation by feature sampling: randomly sampling both rows (bootstrapping) and columns

Random forest: feature sampling

- Decorrelation obtained by:
  - When building a tree, instead of using all variables when making a split, take a random selection of m predictors as candidates to base the split on
  - At each split, take a fresh selection of m predictors
- *m* is typically set to $\sqrt{p}$
- Similar to bagging, the collection of trees (= forest) is built on bootstrapped training samples
- Hence, bagging is a special case of a random forest with $m = p$.

Bootstrap sample 1

(X1, y1)
(X2, y2)
(X5,y5)
(X4,y4)
(X5,y5)

p1	p2	p3	p4	y

Bootstrap sample 1

(X1, y1)
(X2, y2)
(X5,y5)
(X4,y4)
(X5,y5)

p1	p2	p3	p4	y

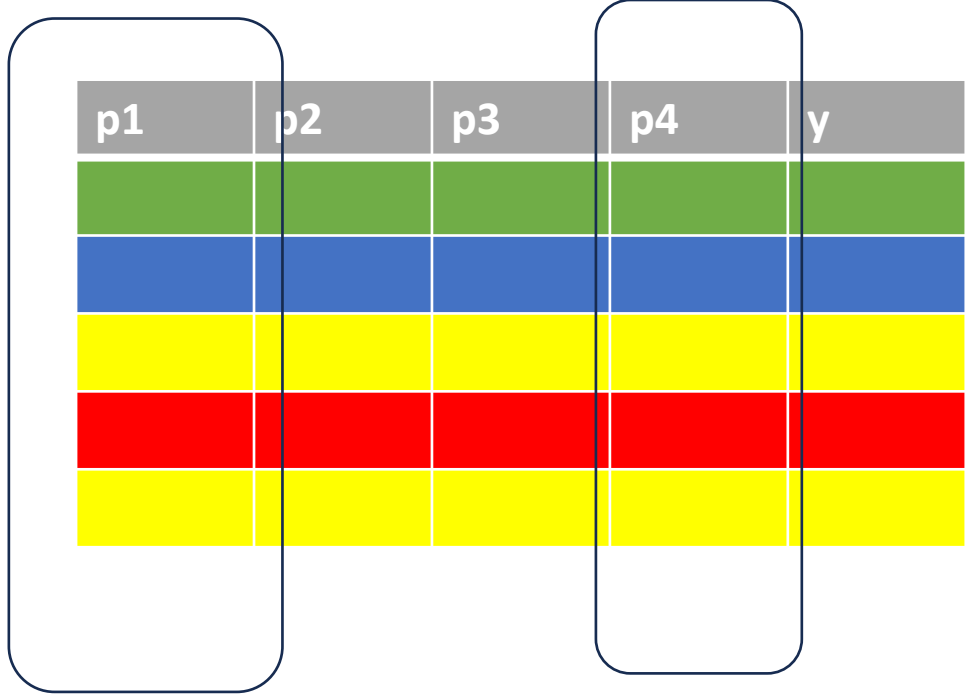
1. Take one bootstrapped sample
2. Build a decision tree:
- To find the best predictor, consider randomly 2 predictors to choose from

p3

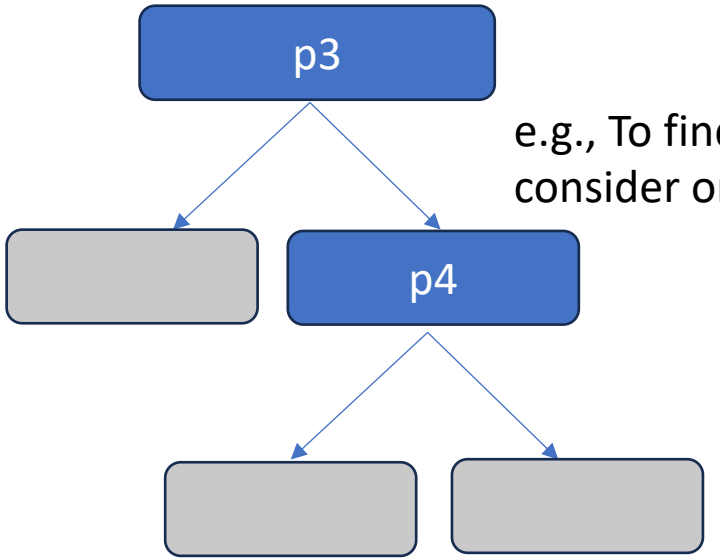
e.g., To find the root of the tree consider only p2, p3

Bootstrap sample 1

(X1, y1)
(X2, y2)
(X5,y5)
(X4,y4)
(X5,y5)



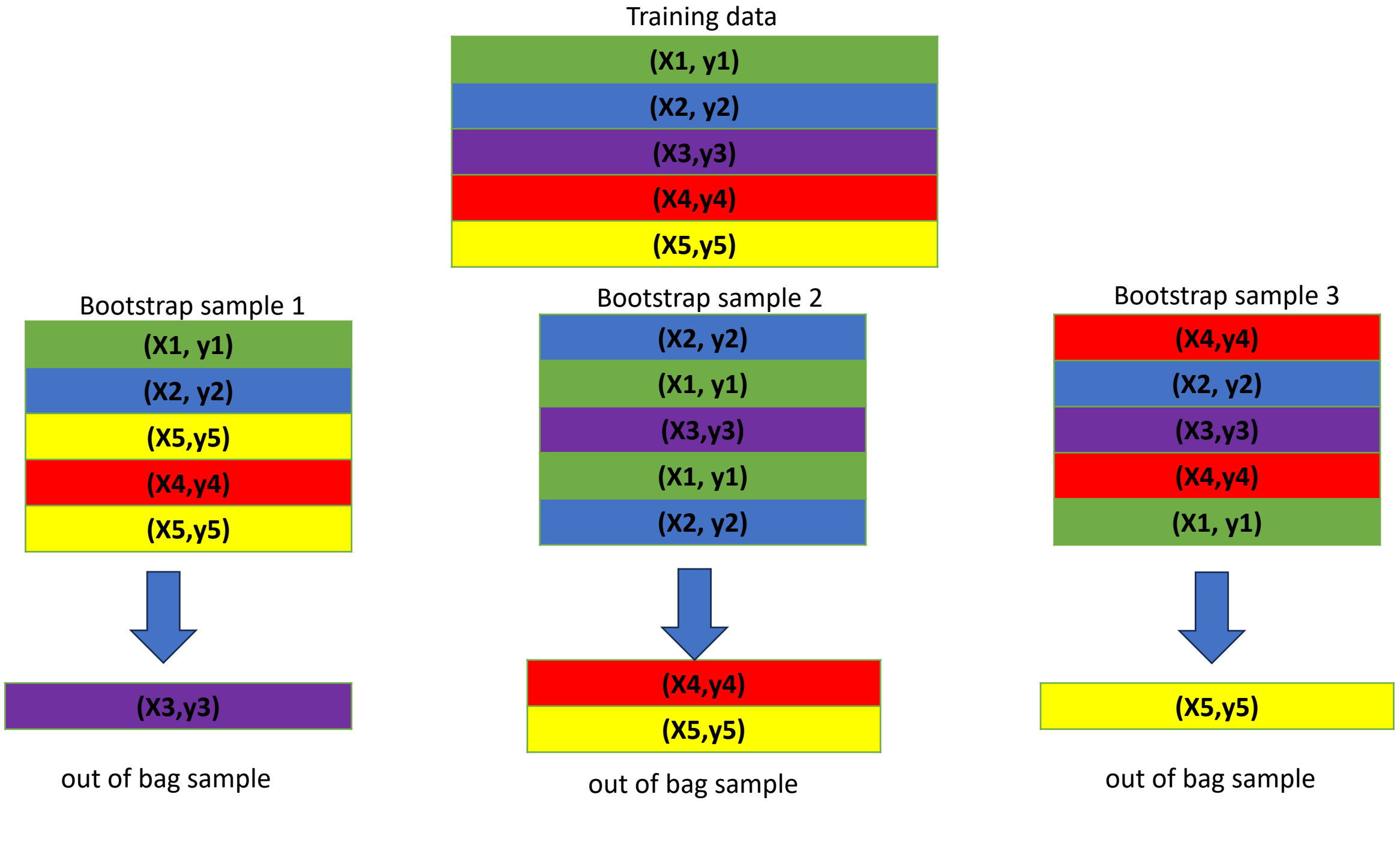
1. Take one bootstrapped sample
2. Build a decision tree:
- To find the best predictor, consider randomly 2 predictors to choose from

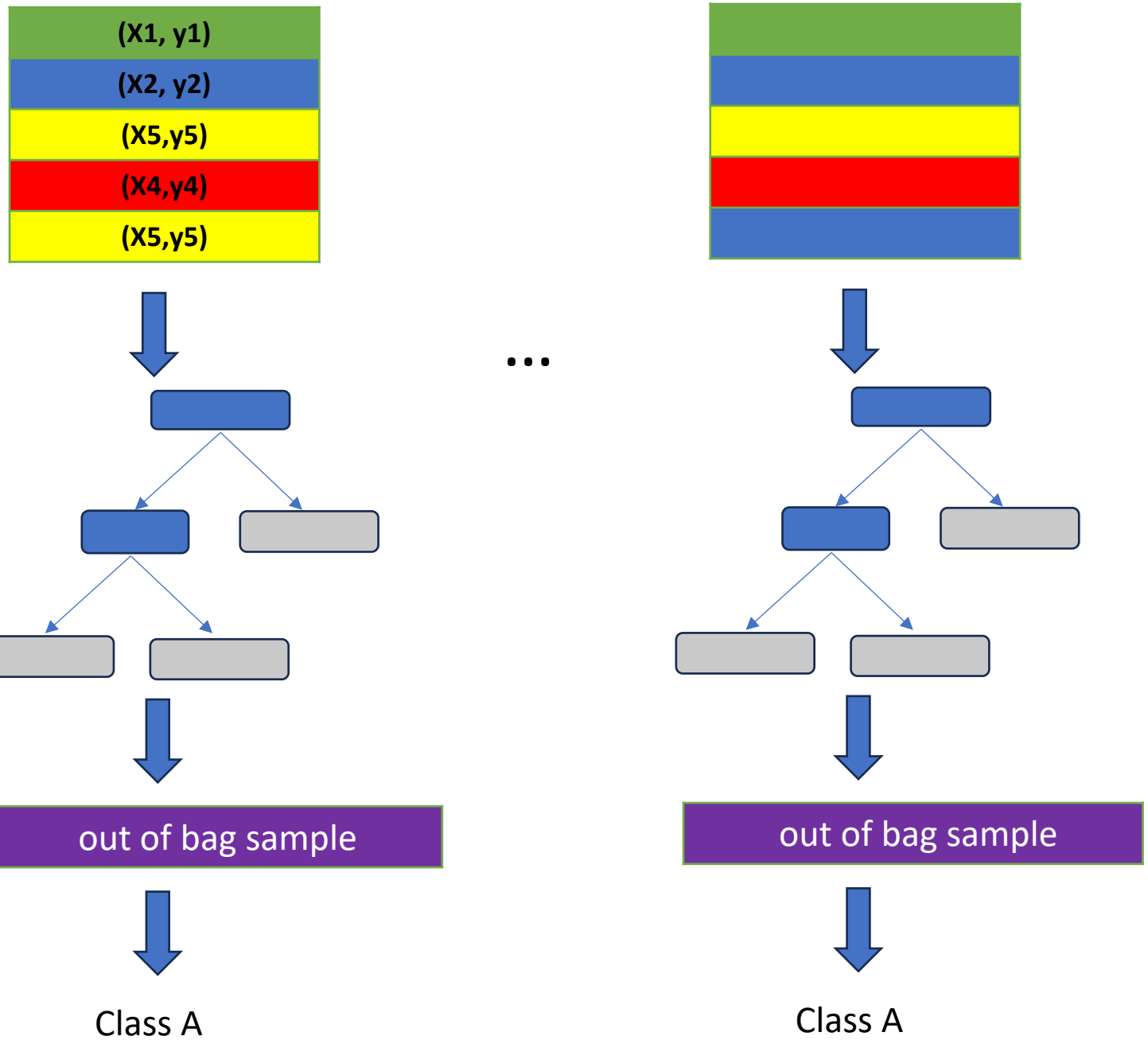


e.g., To find the predictor for the next split, consider only p1, p4

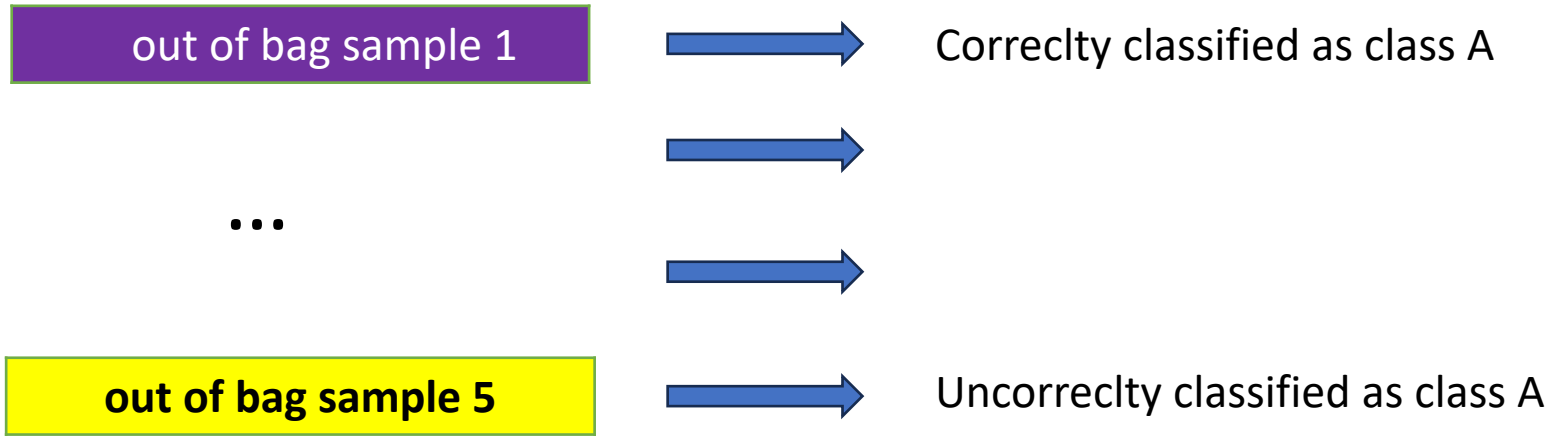
“Out-of-bag” error estimation

- When we do bagging and random forest, there is a very simple way to estimate the test error:
  - In both methods, we take multiple bootstrapped samples of the training data. On average, each tree uses about 2/3 of the observations
  - The remaining 1/3 of observations left out are referred to as the out-of-bag(OOB) observations
- If we want to calculate the error for a particular observation, we can predict the response using each of the trees in which it was OOB. This will give *B/3* predictions for this observation, which we average. When we do this for all observations, we get the OOB error.


Training data
(X1, y1)
(X2, y2)
(X3,y3)
(X4,y4)
(X5,y5)
Bootstrap sample 1
(X1, y1)
(X2, y2)
(X5,y5)
(X4,y4)
(X5,y5)
(X3,y3)
out of bag sample
Bootstrap sample 2
(X2, y2)
(X1, y1)
(X3,y3)
(X1, y1)
(X2, y2)
(X4,y4)
(X5,y5)
out of bag sample
Bootstrap sample 3
(X4,y4)
(X2, y2)
(X3,y3)
(X4,y4)
(X1, y1)
(X5,y5)
out of bag sample


(X1, y1)
(X2, y2)
(X5,y5)
(X4,y4)
(X5,y5)
...
out of bag sample
Class A
out of bag sample
Class A
out of bag sample
Correclty classified as class A

“Out-of-bag” error



- We can measure how accurate our random forest is by the proportion of out-of-the-bag samples that were correclty classified
- The proportion that was incorreclty classified is the Out-Of-Bag error

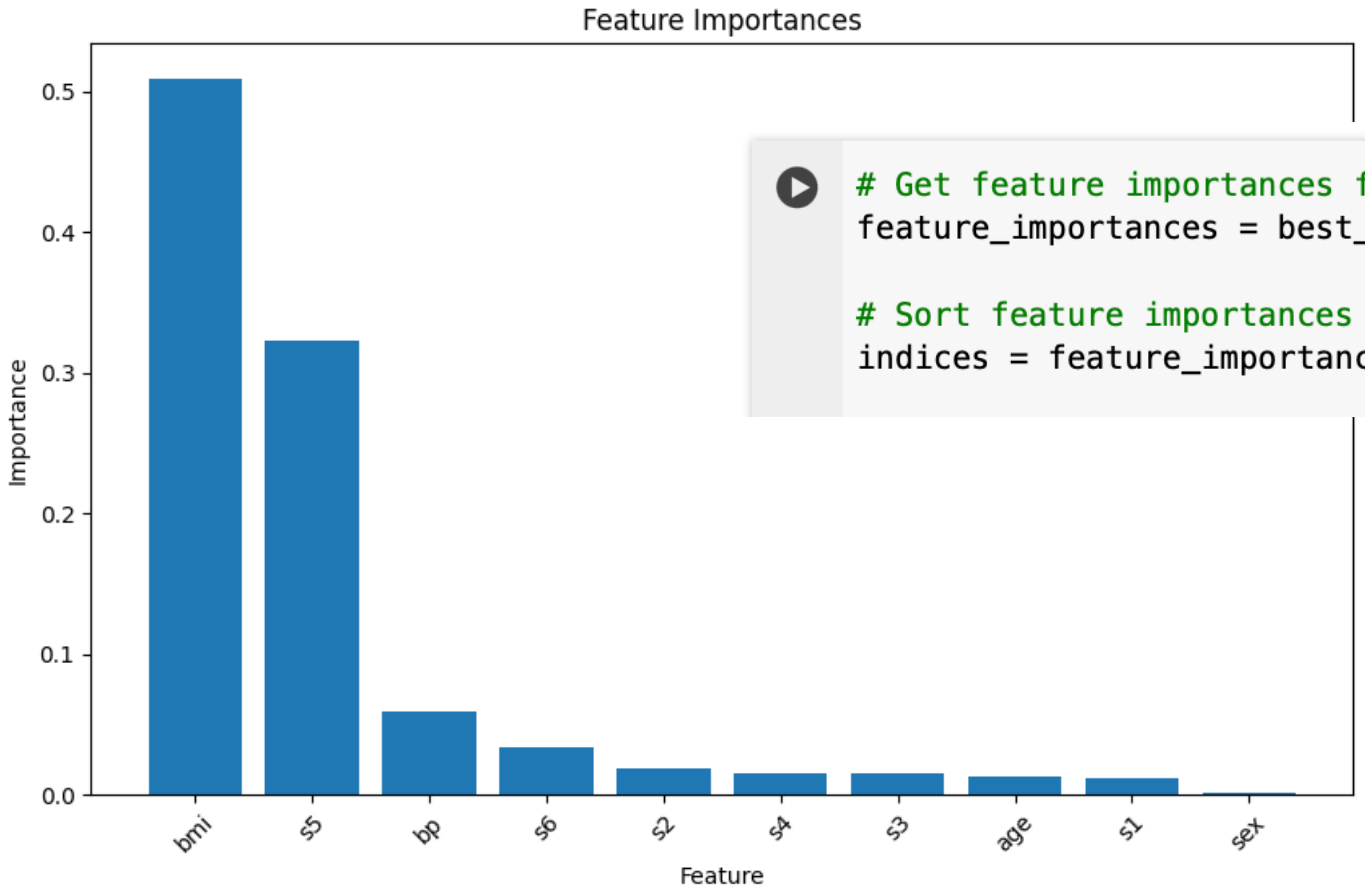
Variable importance measure

- In both bagging and random forest, it can be difficult to interpret the resulting model:
  - When we build a large number of trees, it is no longer possible to (graphically) represent the resulting statistical learning procedure using a single tree
  - How to find out which predictor(s) are most important in predicting a correct outcome?
- Variable importance measures
- Larger value indicates a more important predictor

Impurity-based feature importance (MDI)

- How “important” is variable x_j to the prediction?
- Recall that trees (classification or regression) are grown by minimizing “impurity” (e.g. Gini)
- IDEA: Record the total amount that impurity is decreased due to splits over x_j, averaged over all B trees
- Advantage: Obtained for free with estimation
- Disadvantages: computed on training set statistics and therefore do not reflect the ability of feature to be useful to make predictions that generalize to the test set

Impurity-based feature importance (MDI)




```
# Get feature importances from the best model
feature_importances = best_random_forest.feature_importances_

# Sort feature importances in descending order
indices = feature_importances.argsort()[::-1]
```

Permutation-based feature importance

- IDEA: randomly shuffle one column, and observe how much the performance drops
- Advantage: Doesn’t have the problems of MDI
- Disadvantages:
  - Can take a while, results vary
  - Ignores correlations among predictors (e.g. perfectly correlated features are all “unimportant”)

Summary

Summary

- Decision trees are simple and useful for interpretation
- However, prone to overfitting. Solutions: pruning, bagging and random forests
- Bagging: fit multiple trees to bootstrapped samples of the data, combine to yield a single consensus prediction
- Random Forest: fit trees to bootstrapped samples from the data AND sample predictors. Combine all trees to yield a consensus prediction

Summary

- When using bagging and random forests we can approximate the test error using the Out of Bag (OOB) estimate of the error
- Which predictor is most influential on the outcome can be inferred from variable importance measures
- Random forests often show top-tier performance out of the box, but the resulting model is difficult to interpret

Practical 4