Arboles de Decisión

ML…trade off between errors

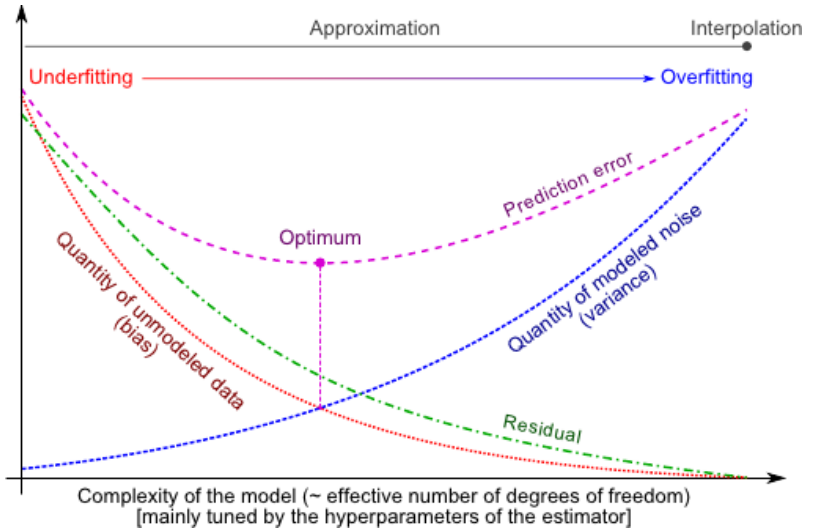
Bias-Variance trade off

A Model's generalization error can be expressed as the sum of three errors:

Bias wrong assumptions, such as assuming that the data is linear when it is actually quadratic. (underfitting)

Variance Sensitivity to small variations in the training data. Models with no o flexible assumptions (like a high-degree polynomial model) are likely to overfit the data.

Irreducible error This part is due to the noisiness of the data itself.



Increasing a model's complexity will typically increase its variance and reduce its bias.

Conversely, reducing a model's complexity increases its bias and reduces its variance

ML…trade off between errors

Bias-Variance trade off

Models can be:

Low Variance – High Bias:

We have a hypothesis and apply some optimization techniques to find the best **parameters**

$$y = \mathbf{m}x + \mathbf{b}$$

High Variance – Low Bias

Can we find a way to **learn** from the data **without making initial assumptions ?**

Decision trees

- **Decision trees**
- **Construction decision trees**
  - **Information measures**
- **The application of decision trees to classification or Regression**
- **Reducing variance: Ensemble methods (Random Forests)**

Decision trees: simple yet powerful

- **Decision tree**

Is a function that takes a vector of attribute values as its input, and returns a "decision" as its output.

Both input and output values can be measured on a nominal, ordinal, interval, and ratio scales, can be discrete or continuous.

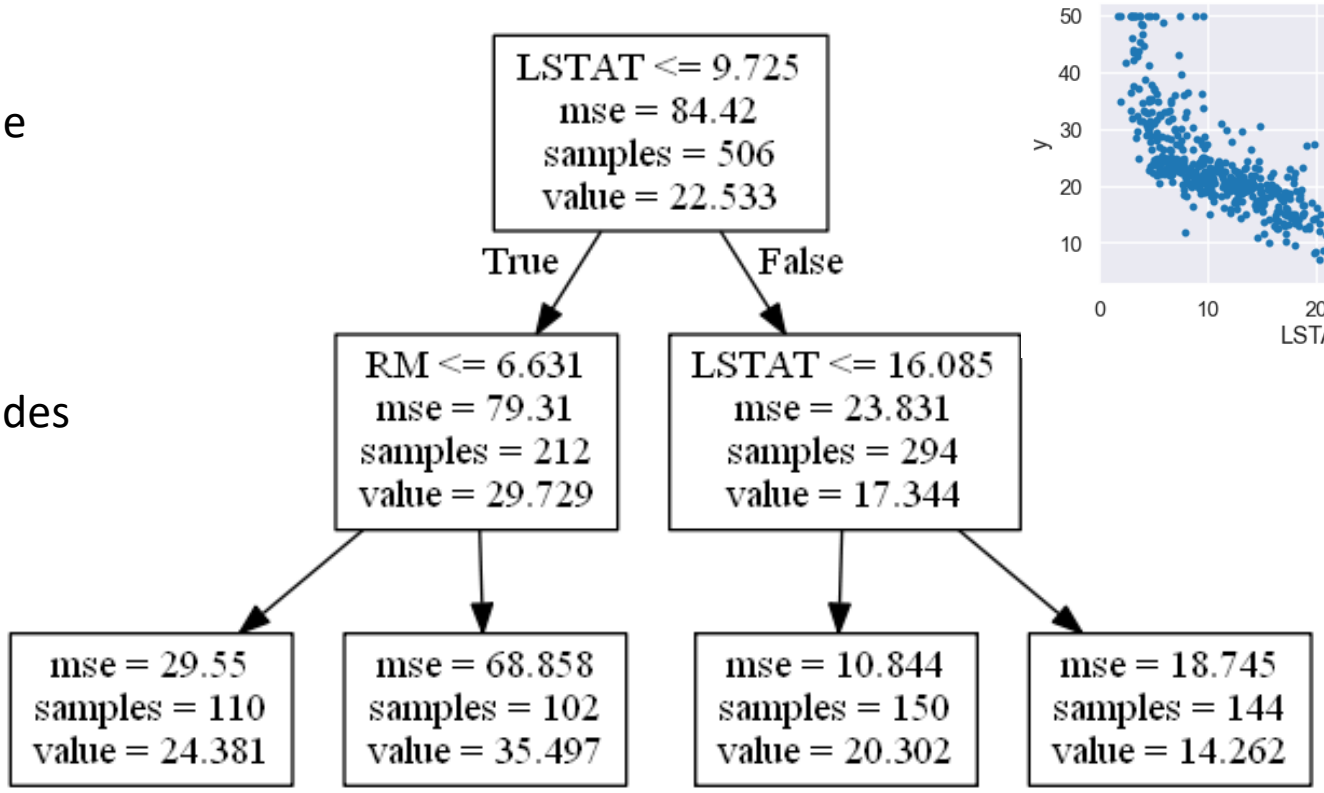
It consists in a **learned set of decisions that iteratively stratify the predictor space** into smaller regions that can be aggregated (mean or mode) in order to give a final answer.

Decision tree:
example & structure

Root node

Internal nodes

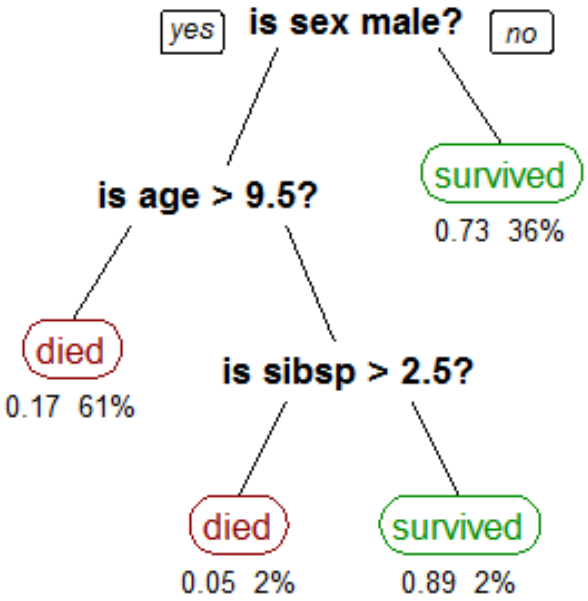
Leaf Node

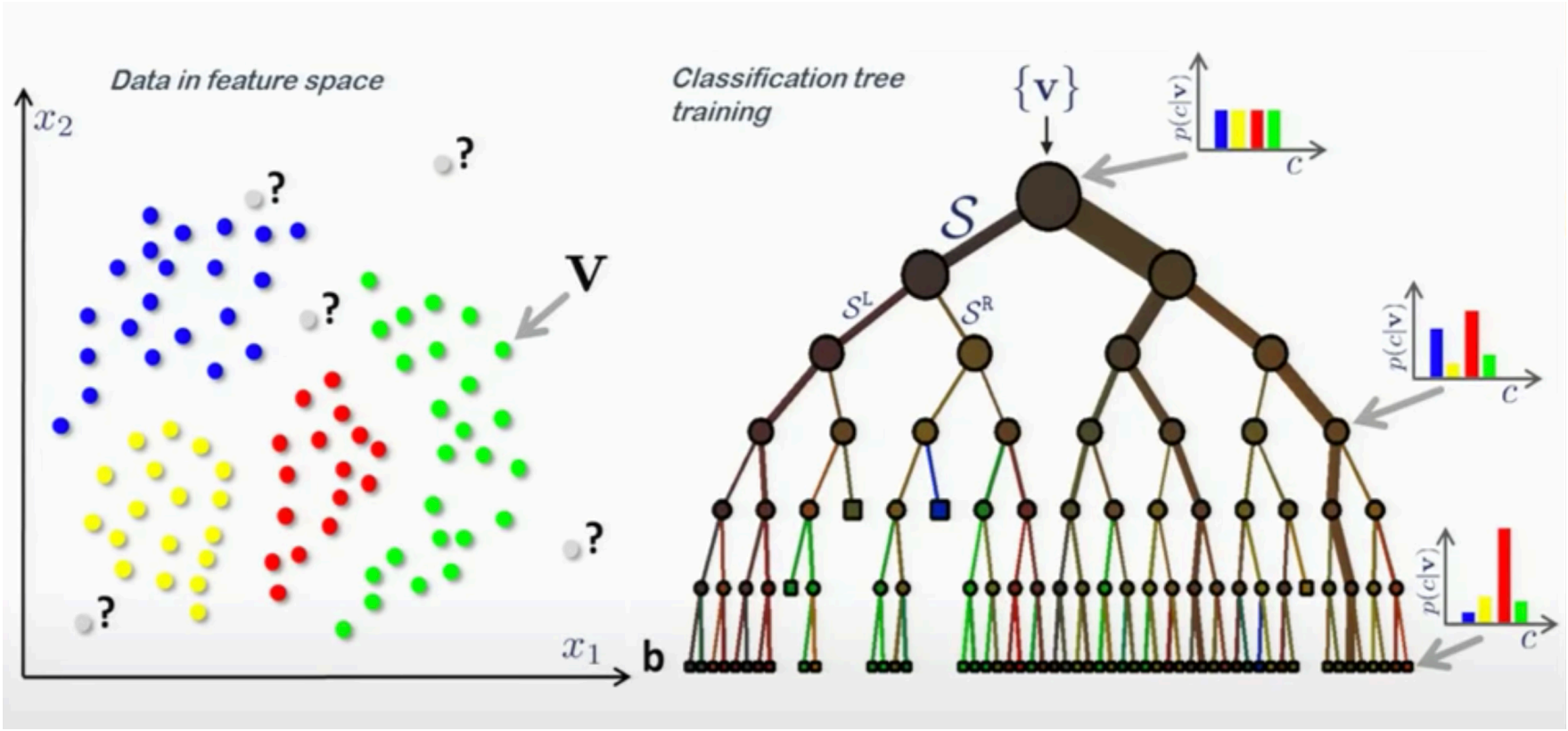


Classification Trees

For a classification tree, **we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs**.

Therefore, in interpreting the results, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region.





$$\mathcal{S}_j = \mathcal{S}_j^{\mathrm{L}} \cup \mathcal{S}_j^{\mathrm{R}}$$

child subsets are mutually exclusive

Classification Trees

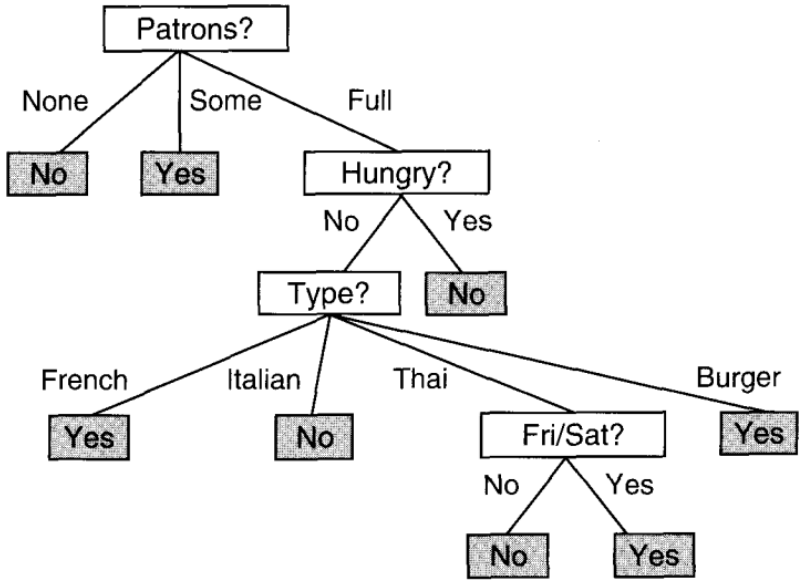
1. **Alternate**: whether there is a suitable alternative restaurant nearby.
2. **Bar**. whether the restaurant has a comfortable bar area to wait in.
3. **Fri/Sat:** true on Fridays and Saturdays.
4. **Hungry**: whether we are hungry.
5. **Patrons**: how many people are in the restaurant (values are None, Some, and Full).
6. **Price**: the restaurant's price range ($, $$, $$$).
7. **Raining**: whether it is raining outside.
8. **Reservation**: whether we made a reservation.
9. **Type**: the kind of restaurant (French, Italian, Thai, or Burger).
10. **WaitEstimate**: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

Will-Wait Dataset

ID	Has alternative?	Bar?	Fri/Sat?	Hungry?	Patrons?	Price	Raining?	Reservation?	Type	Wait-estimate	Will wait?
1	Y	N	N	Y	some	$$$	N	Y	French	0-10	Y
2	Y	N	N	Y	full	$	N	N	Thai	30-60	N
3	N	Y	N	N	some	$	N	N	Burger	0-10	Y
4	Y	N	Y	Y	full	$	Y	N	Thai	10-30	Y
5	Y	N	Y	N	full	$$$	N	Y	French	>60	N
6	N	Y	N	Y	some	$$	Y	Y	Italian	0-10	Y
7	N	Y	N	N	none	$	Y	N	Burger	0-10	N
8	N	N	N	Y	some	$$	Y	Y	Thai	0-10	Y
9	N	Y	Y	N	full	$	Y	N	Burger	>60	N
10	Y	Y	Y	Y	full	$$$	N	Y	Italian	10-30	N
11	N	N	N	N	none	$	N	N	Thai	0-10	N
12	Y	Y	Y	Y	full	$	N	N	Burger	30-60	Y

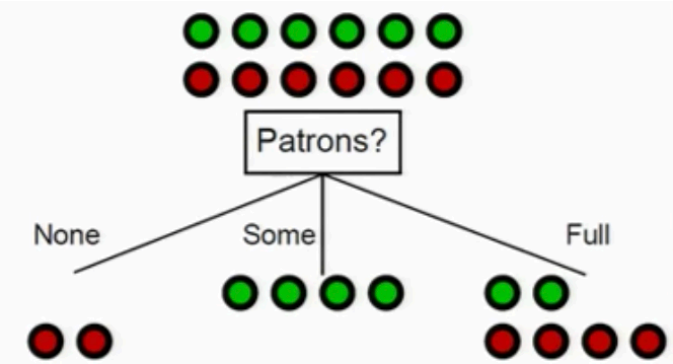
Classification Trees

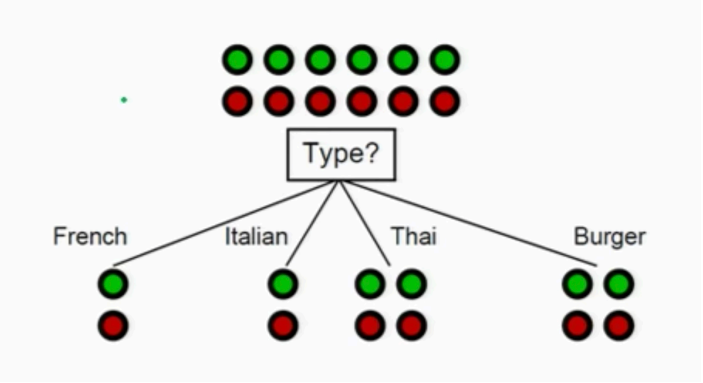
Will-Wait Dataset											
ID	Has alternative?	Bar?	Fri/Sat?	Hungry?	Patrons?	Price	Raining?	Reservation?	Type	Wait-estimate	Will wait?
1	Y	N	N	Y	some	$$$	N	Y	French	0-10	Y
2	Y	N	N	Y	full	$	N	N	Thai	30-60	N
3	N	Y	N	N	some	$	N	N	Burger	0-10	Y
4	Y	N	Y	Y	full	$	Y	N	Thai	10-30	Y
5	Y	N	Y	N	full	$$$	N	Y	French	>60	N
6	N	Y	N	Y	some	$$	Y	Y	Italian	0-10	Y
7	N	Y	N	N	none	$	Y	N	Burger	0-10	N
8	N	N	N	Y	some	$$					
9	N	Y	Y	N	full	$					
10	Y	Y	Y	Y	full	$$$					
11	N	N	N	N	none	$					
12	Y	Y	Y	Y	full	$					



Classification Trees: Which feature ?

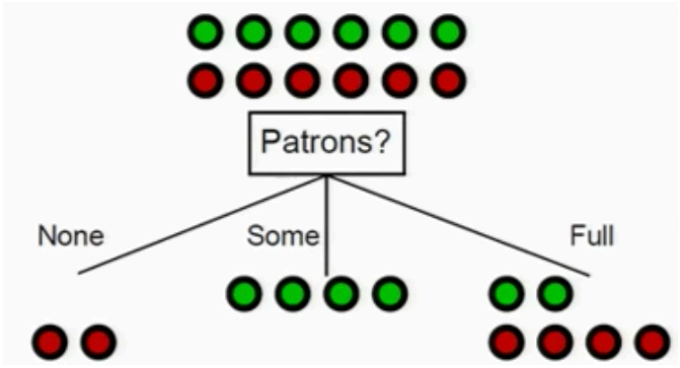
Which feature would you peak ?

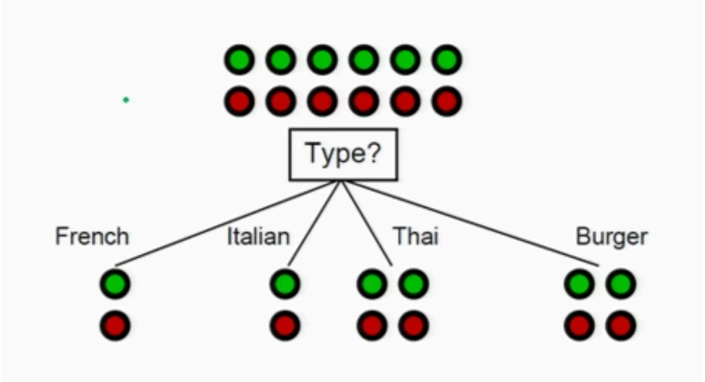




Classification Trees: Which feature ?

Which feature would you peak ?





By choosing **Patrons is more informative**

How do we measure information ?

Classification Trees: information measures

In practice, two other measures are preferable:

- Gini
- Entropy:
  - In Machine Learning, entropy is frequently used as an impurity measure: a set's entropy is zero when it contains instances of only one class.

Here p_{mc} represents the proportion of training observations in the region R_m that are from the class c.

$$G = \sum_{c=1}^{C} p_{mc}(1 - p_{mc})$$

$$E = -\sum_{c=1}^{C} p_{mc} \log p_{mc}$$

A mathematical theory of communications – C. Shannon (1948)

Classification Trees: Which feature ?

Entropy

$$H(s) = -\sum p_c log_2 p_c$$

Entropy is 0 if all samples at a node belong to the same class

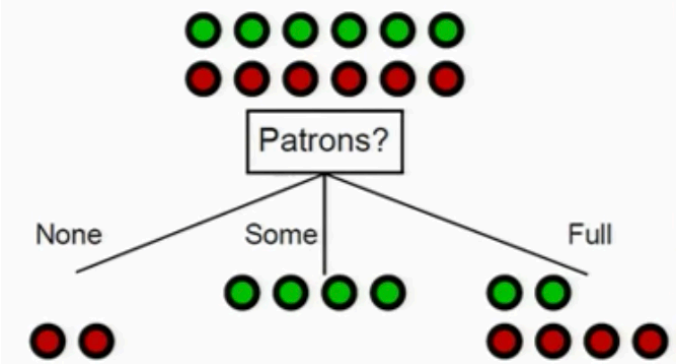
Expected Entropy: after choosing an attribute **A** with values **K** and generate **E1…Ek splits**

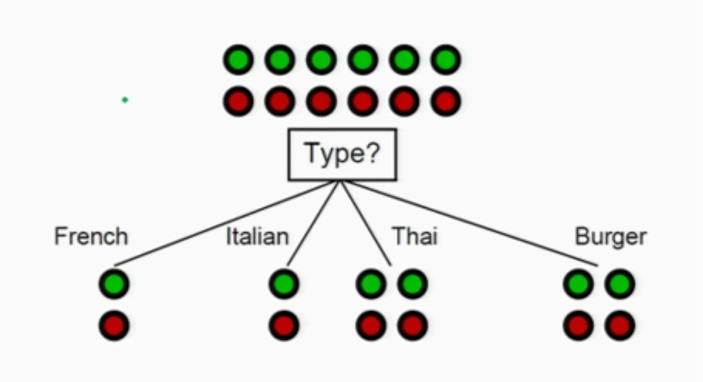
$$EH(s) = \sum_i^k \frac{p_i + n_i}{p + n} H(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i})$$

Information gain: the amount of information that we gain (or how much entropy we reduce)

$$IG = H(s) - EH(s)$$

Classification Trees: Which feature ?





Patrons

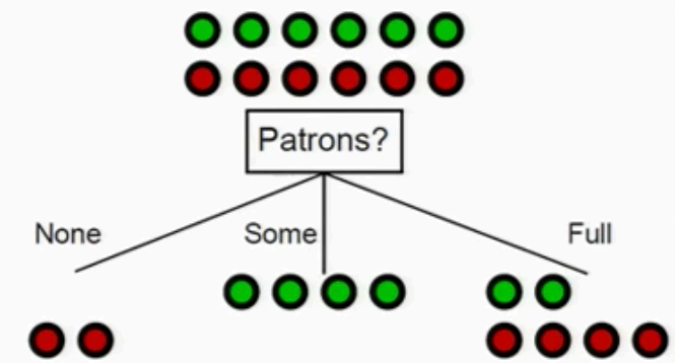
$$H(s) = -\sum p_c log_2 p_c = -\frac{6}{12} log_2 \frac{6}{12} - \frac{6}{12} log_2 \frac{6}{12} = 1$$

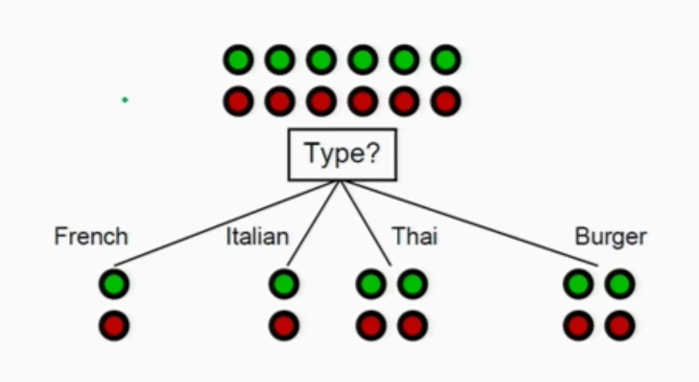
$$EH(s) = \sum_i^k \frac{p_i + n_i}{p+n} H(\frac{p_i}{p+n}, \frac{ni}{p+n}) = \frac{2}{12} H(0,1) + \frac{4}{12} H(1,0) + \frac{6}{12} H\left(\frac{2}{6}, \frac{4}{6}\right) = 0.459$$

$$IG = H(s) - EH(s) = 1 - 0.459 = 0.541$$

Classification Trees: Which feature ?

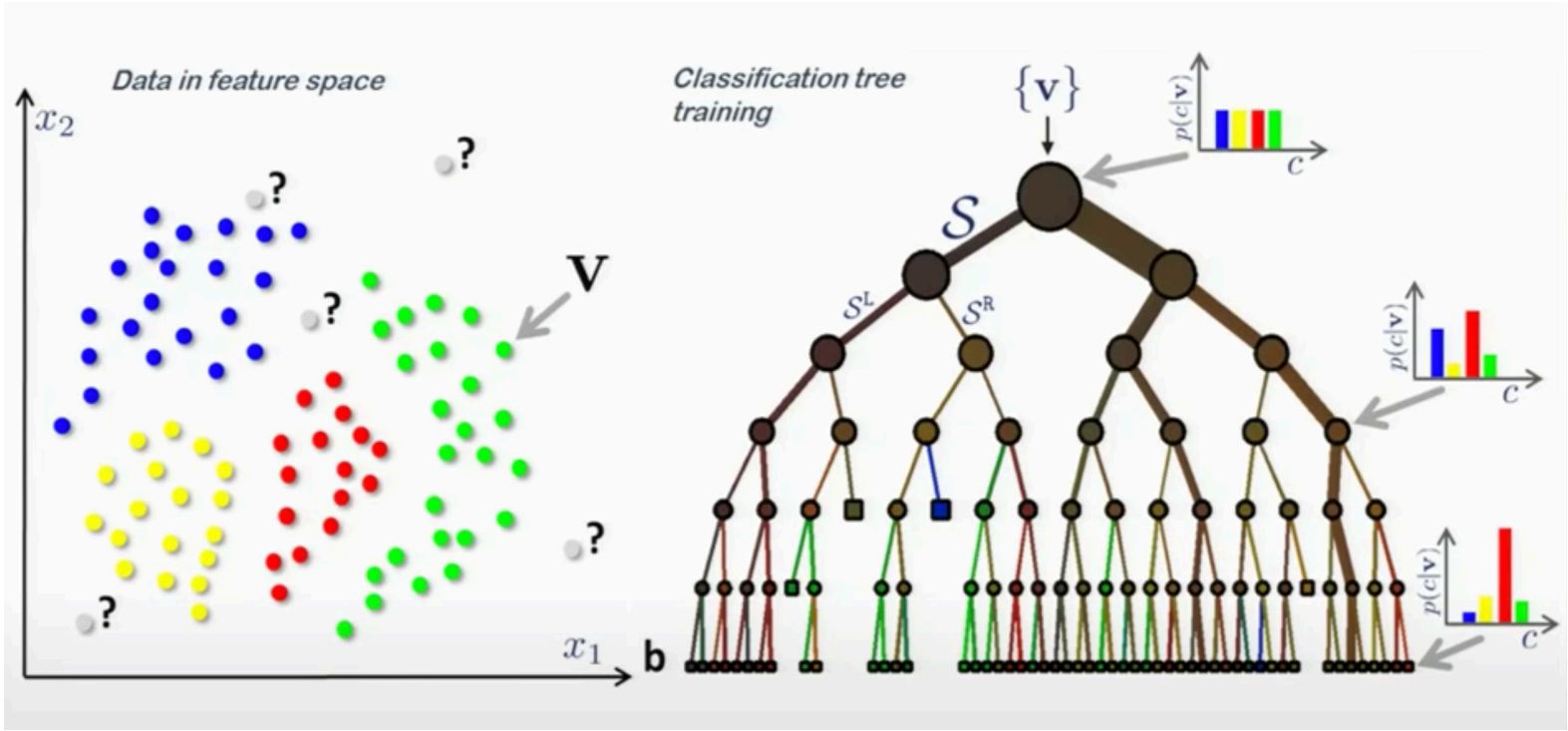
Entropy and Information gain





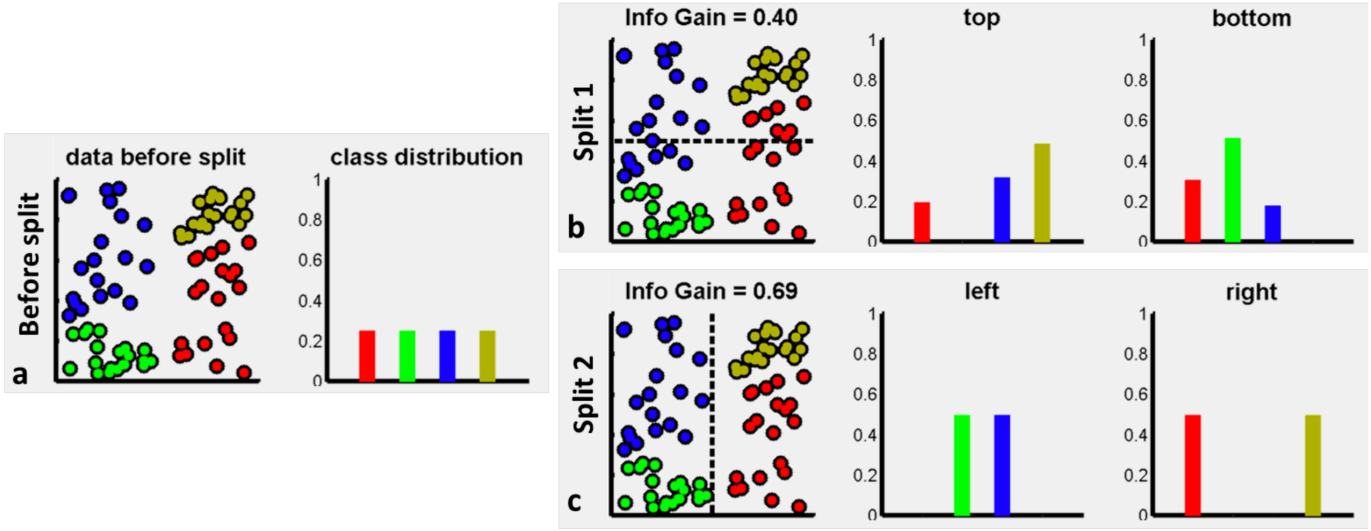
Type

Classification Trees: Construction ?



Classification Trees: Construction ?

1. **Start with empty tree**
2. **Select a feature to split the data**
3. **For each split:**
    1. **If “pure” subset or stopping condition: Stop**
    2. **Go to 2 and recurse.**



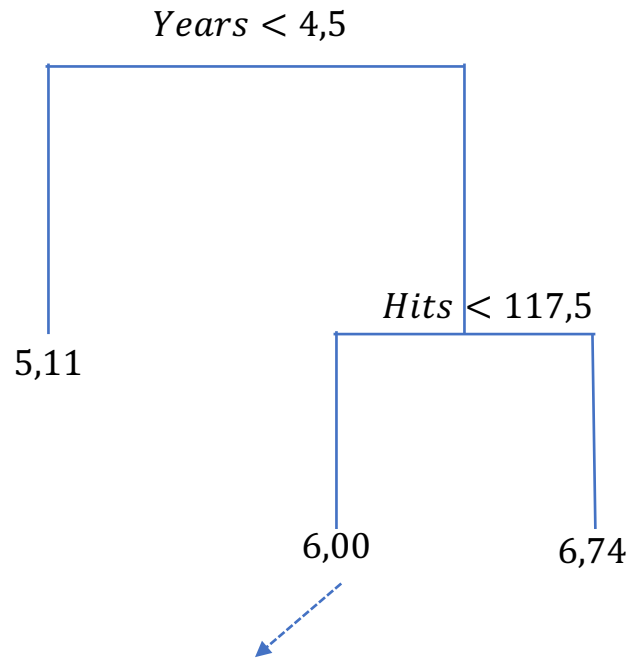
Regression Trees (CART)

A simple example

Data: *hitters* (https://rdrr.io/cran/ISLR/man/Hitters.html)

The figure shows a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.

At a node, the label (of the form $x_j < t$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $x_j \geq t$.



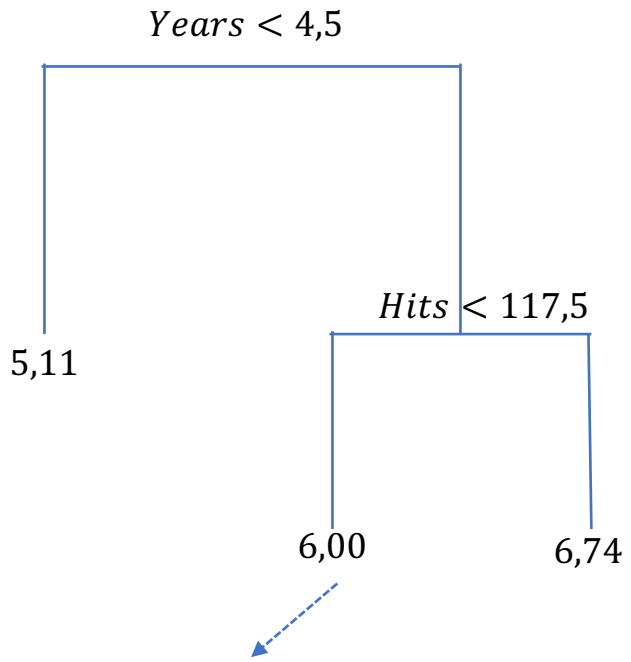
Regression Trees (CART)

A simple example

Data: *hitters* (https://rdrr.io/cran/ISLR/man/Hitters.html)

The figure shows a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.

At a node, the label (of the form $x_j < t$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $x_j \geq t$.



Regression Trees (CART)

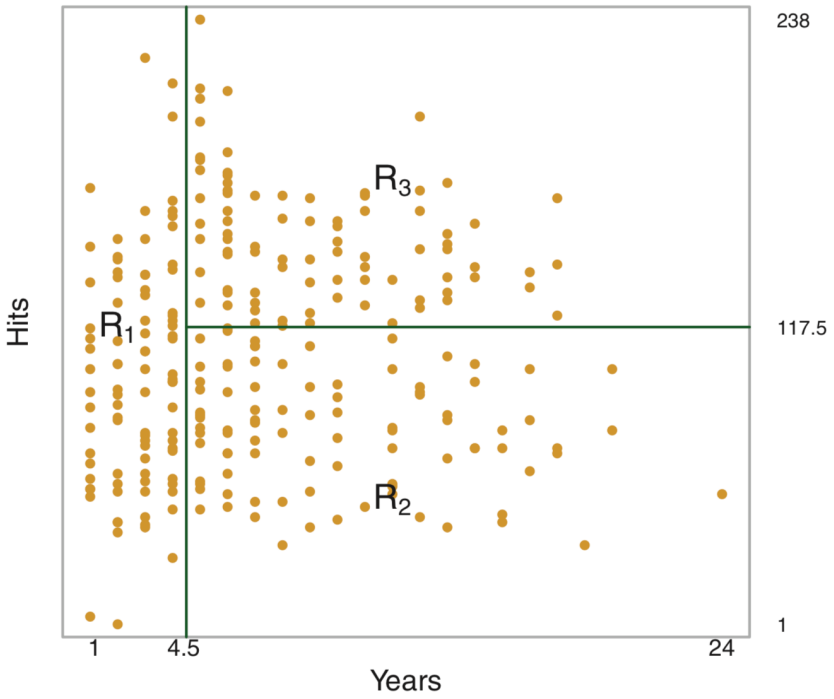
Graphical representation

The tree stratifies or segments the ***players*** into three regions of predictor space: players who have played for four or fewer years, players who have played for five or more years and who made fewer than 118 hits last year, and players who have played for five or more years and who made at least 118 hits last year. These three regions can be written as

$$R_1 = \{x_j | years < 4.5\}$$

$$R_2 = \{x_j | years \geq 4.5, hits < 117.5\}$$

$$R_3 = \{x_j | years \geq 4.5, hits \geq 117.5\}$$

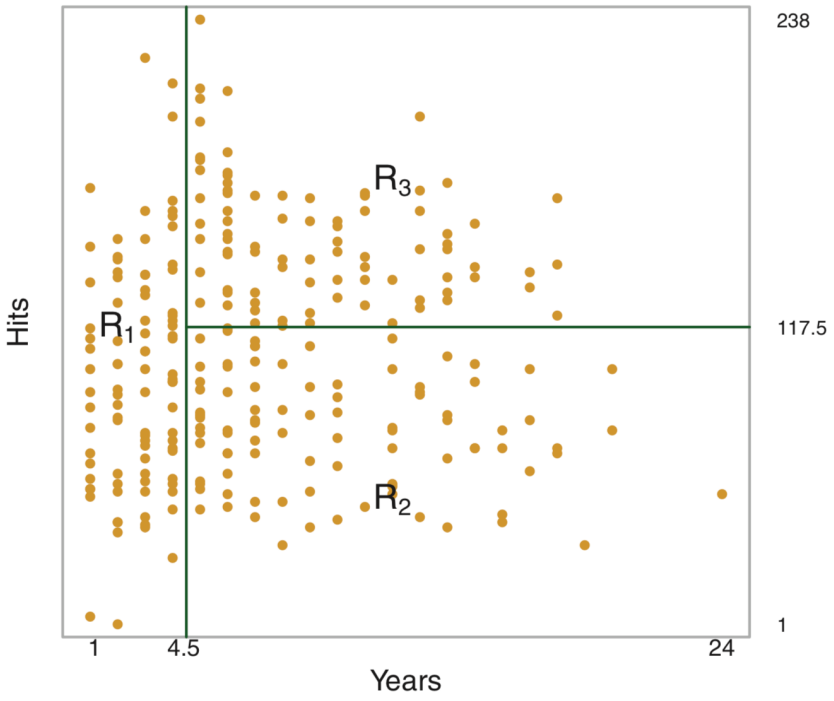


Regression Trees

In order to build a regression tree:

1. Divide the feature space into ***M*** distinct and non-overlapping regions:

2. For every observation, $x^{(i)}$, that falls into the region R_m , we make the same prediction, which is simply the mean of the response values for the training observations in R_m ($\hat{y}_{R_m}$)



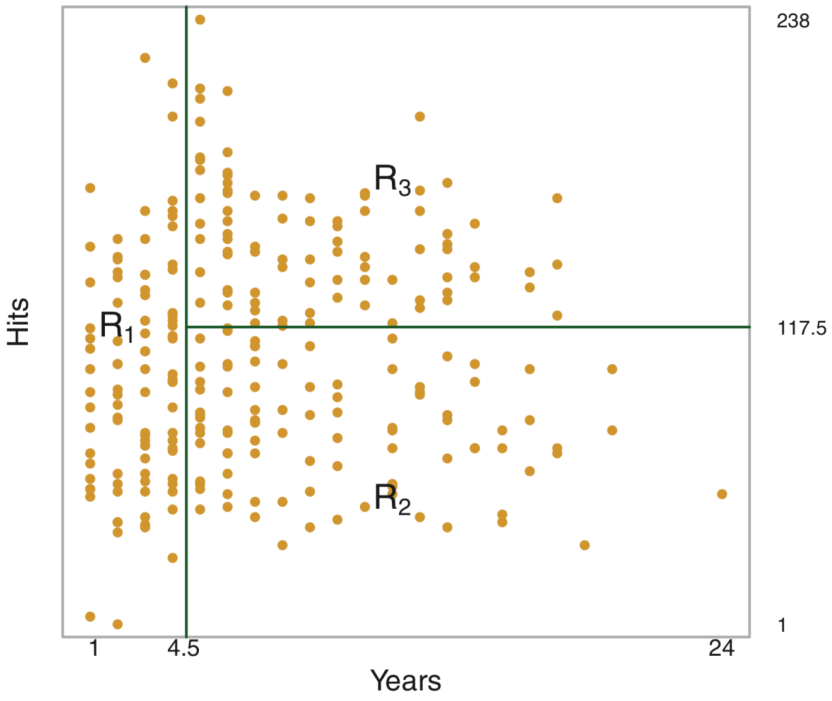
Regression Trees

How do we construct the regions R_m?

The goal is to find regions $R_1, R_2, R_3, \cdots, R_M$ that minimize the squared error, given by

It is computationally infeasible to consider every possible partition of the feature space into M regions

$$\sum_{m=1}^{M} \sum_{i \in R_m} \left(y^{(i)} - \hat{y}_{R_m}\right)^2$$



Recursive binary splitting

This top-down, greedy approach, begins at the top of the tree(a single region) and then successively splits the feature space.

It is **greedy** because at each step of the tree-building process, the best split is made at that particular step.

Repeat the process, in order to split the data further so as to minimize the error within each of the resulting regions.

The process continues until a stopping criterion is reached

Select the feature x_j and the cutpoint t such that splitting the feature space into the regions

$$R_1(j,t) = \{x | x_j < t\}, \quad R_2(j,t) = \{x | x_j \geq t\}$$

To seek the value of j and t that minimize the equation

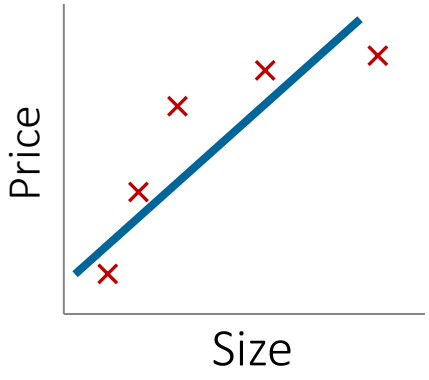
$$\sum_{i \in R_1} \left(y^{(i)} - \hat{y}_{R_1}\right)^2 + \sum_{i \in R_2} \left(y^{(i)} - \hat{y}_{R_2}\right)^2$$

Remember: **Bias**-Variance tradeoff

A learning algorithm is **biased** for a particular input (x) if, when trained on different data sets, it is systematically incorrect when predicting the correct output for x.

Therefore, the **bias** is an error from erroneous assumptions in the learning algorithm.

High bias can cause an algorithm to miss the relevant relations between features and target outputs (***underfitting***).



$$\theta_0 + \theta_1 x$$

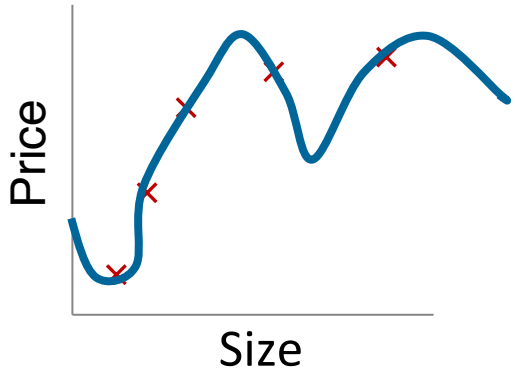
Bias
(Underfitting)

Remember: Bias-**Variance** tradeoff

A learning algorithm has high **variance** for a particular input x if it predicts different output values when trained on different training sets.

The **variance** is an error from sensitivity to small fluctuations in the training set.

High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (***overfitting***).



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Variance
(Overfitting)

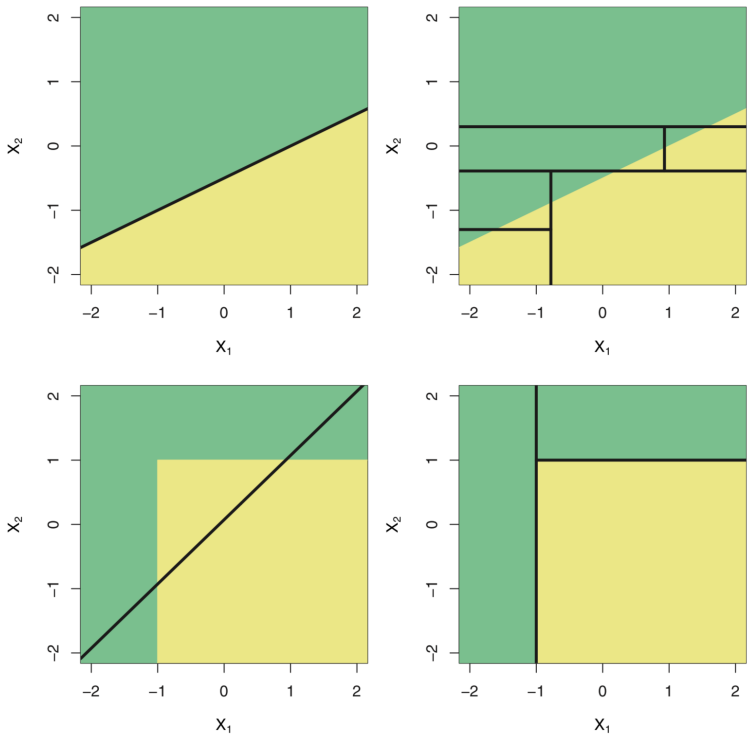
Trees vs Linear Models

$$h_\theta(x) = g\left(\sum_{j=0}^{n} \theta_j x_j\right)$$

$$h_t(x) = \hat{y}_{R_m} \cdot 1\{\mathrm{x} \in R_m\}$$

Which model is better? It depends on the problem at hand:

- Linear vs Non-linear feature space



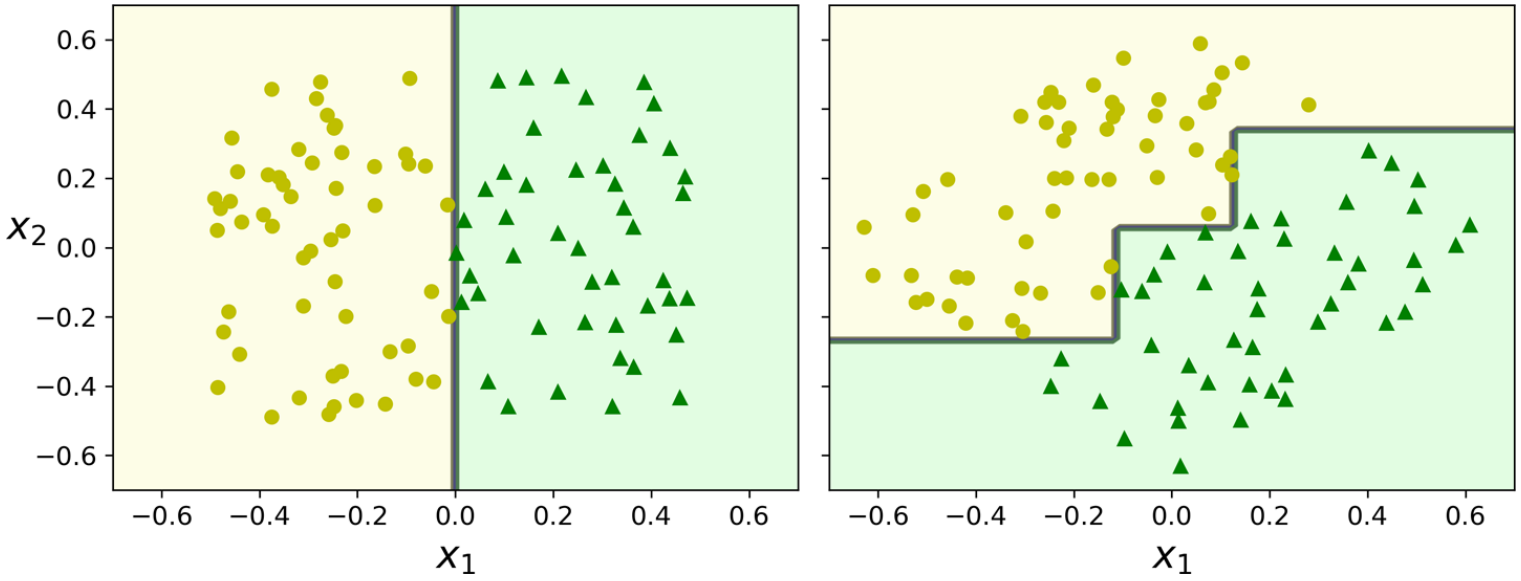
CART: Advantages and Disadvantages

+ Explainable; Trees are very easy to explain to people
+ Human: decision trees more closely mirror human decision-making
+ Machine performance: fast learning & prediction algorithms (Real-time)
− Bias: Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree
− High Variance: Tends to overfiting and they can be unstable because small variations in the data might result in a completely different tree being generated

Instability

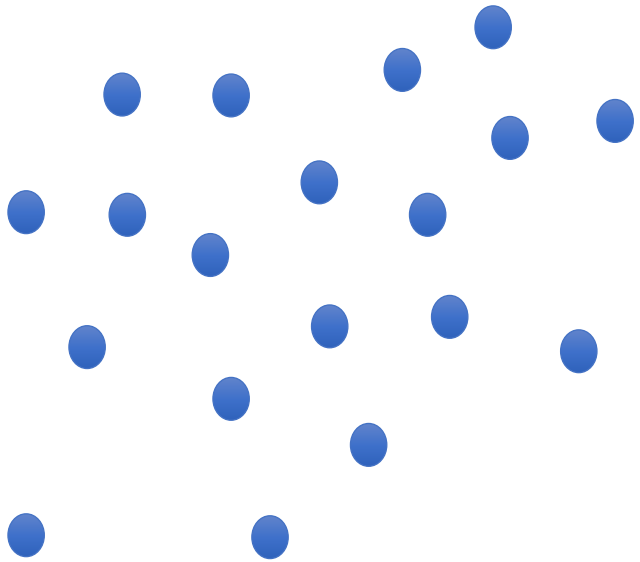
Decision Trees love orthogonal decision boundaries (all splits are perpendicular to an axis), which makes them sensitive to training set rotation.

For example, figure below shows a simple linearly separable dataset: on the left, a Decision Tree can split it easily, while on the right, after the dataset is rotated by 45°, the decision boundary looks unnecessarily convoluted



Ensemble methods

How many balls ?

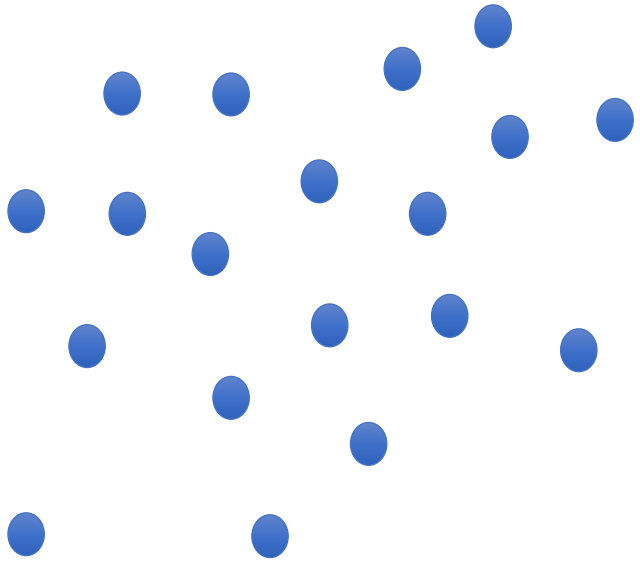


Ensemble methods

How many balls ? 19

Wisdom of the crowd

Similarly, if you **aggregate the predictions of a group of predictors** (such as classifiers or regressors), you **will often get better predictions than with the best individual predictor.**
A **group of predictors** is called an ***ensemble***; thus, this technique is called *Ensemble Learning*, and an Ensemble Learning algorithm is called an *Ensemble method*.



Bagging (bootstrap aggregating)

CART suffers of high variance… however by means of **bootstrapping** we can reduce the variance by taking many training sets from the dataset, build a separate model using each training set, and average the resulting predictions.

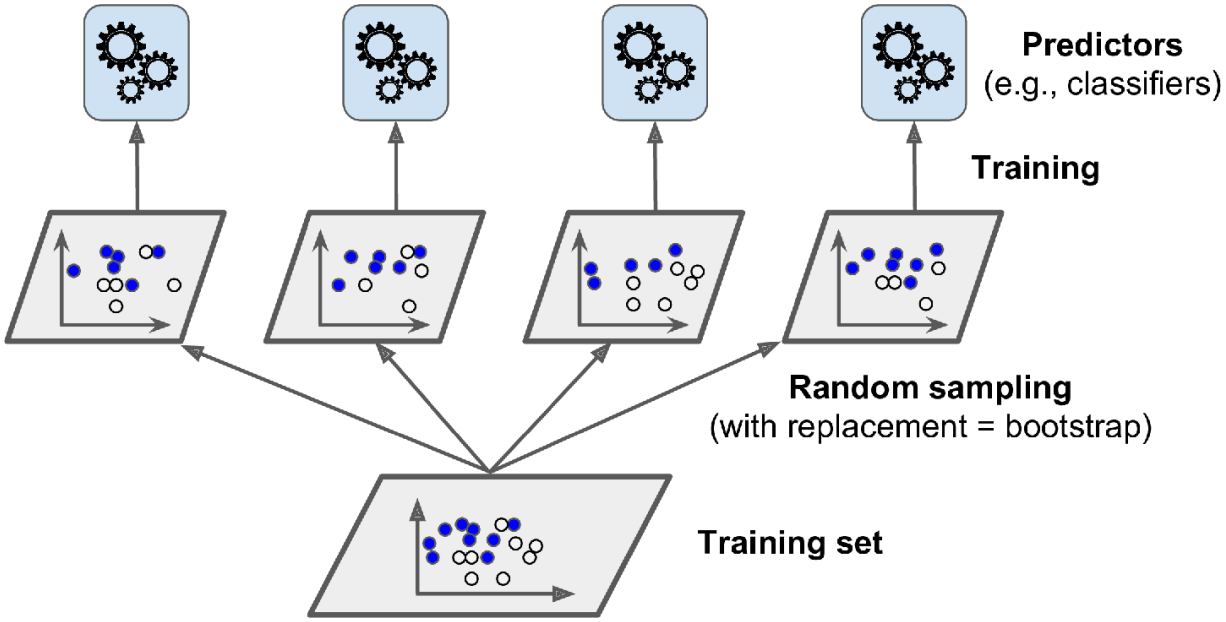
To apply **bagging** to regression trees, we simply construct *T* regression trees using *T* bootstrapped training sets and average the resulting predictions. **These trees are grown deep and are not pruned. Hence each individual tree has high variance, but low bias.** Averaging these *T* trees reduces the variance.

Calculate, $h_1(x), h_2(x), \ldots, h_T(x)$ using *T* separate training sets, and average them in order to obtain a single low-variance statistical learning model

$$h(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

Quiz: how it works for classification trees?

Bagging (bootstrap aggregating)



Bagging (bootstrap aggregating)

Ensemble methods work best when the predictors are as independent from one another as possible.

Why ?

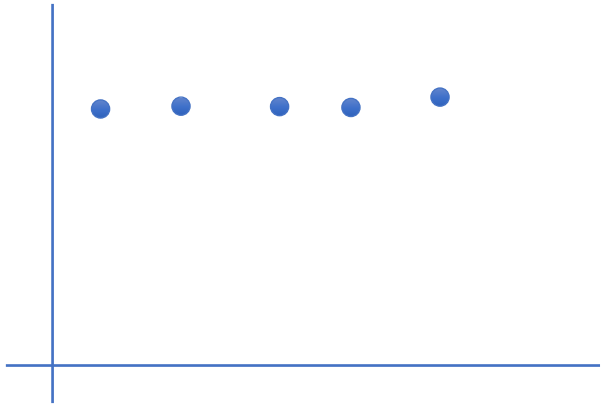
Tree 1: Bias very low - Variance very high

.

.

.

Tree n: Bias very low – Variance very high

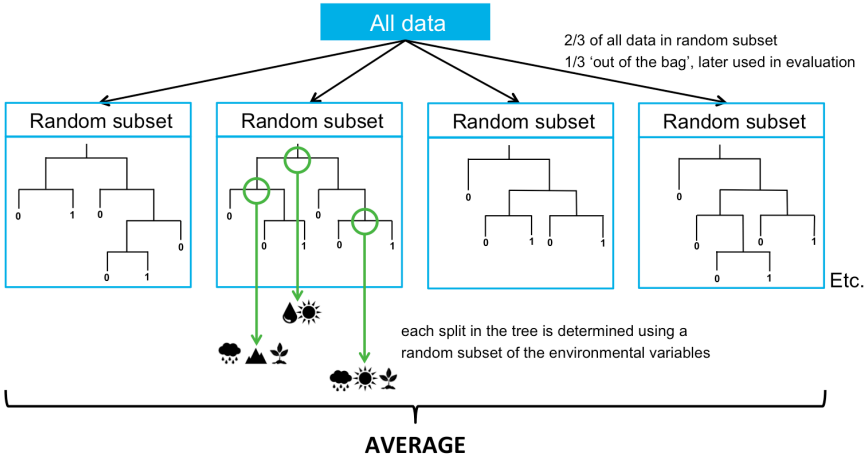


Averaging a set of observations reduces variance.

Random Forest

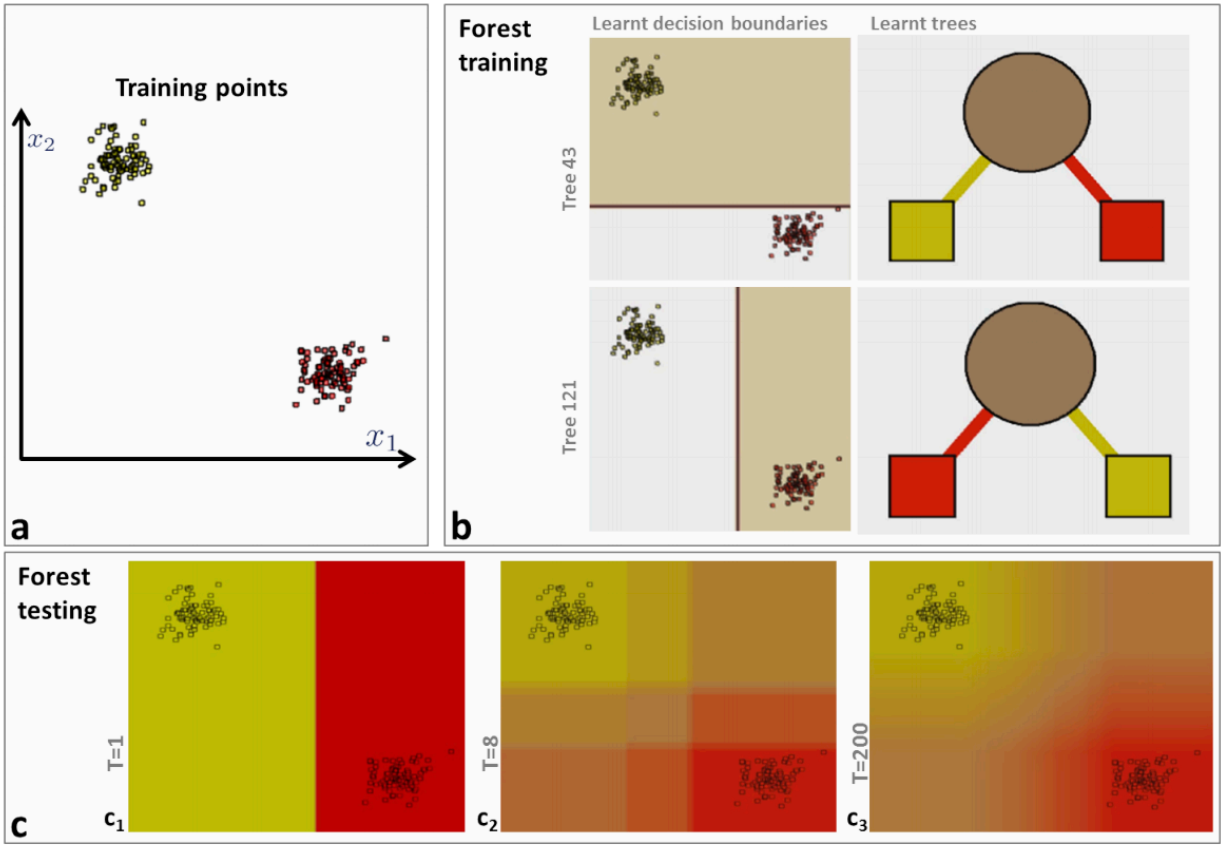
Random Forest provide an improvement over bagged trees by way of a small tweak that decorrelates the trees:

> "when building the decision trees in a bagging approach, each time a split in a tree is considered, a *random sample of features* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those features. A fresh sample of features is taken at each split (typically we choose $\sqrt{n}$)"



> find the set of predictor variables that produce the strongest classification model

the effect of forest size



Different training trees produce different partitions and thus different leaf predictors.

The colour of tree nodes and edges indicates the class probability of training points going through them

In testing, increasing the forest size T produces smoother class posteriors.

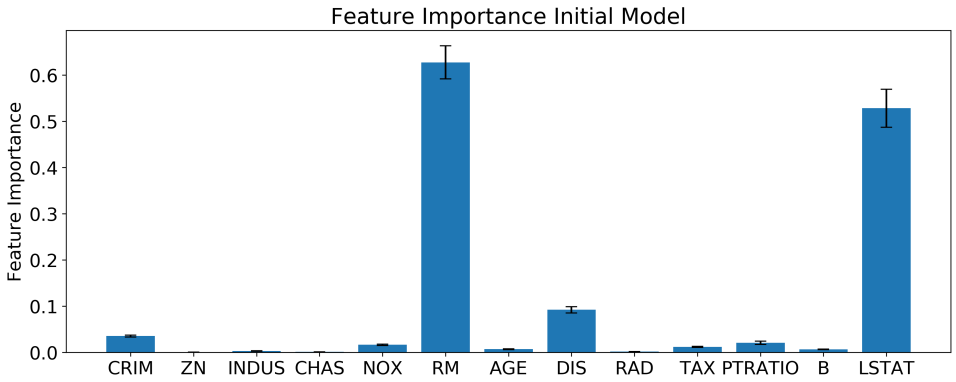
Random Forest: Feature Importance

Random Forest

Feature importance

They make it easy to measure the relative importance of each feature.

Scikit-Learn measures a **feature's importance by looking at how much the tree nodes that use that feature reduce impurity on average** (across all trees in the forest).



Boosting

Boosting

To combine several weak learners into a strong learner.

The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor

Most popular

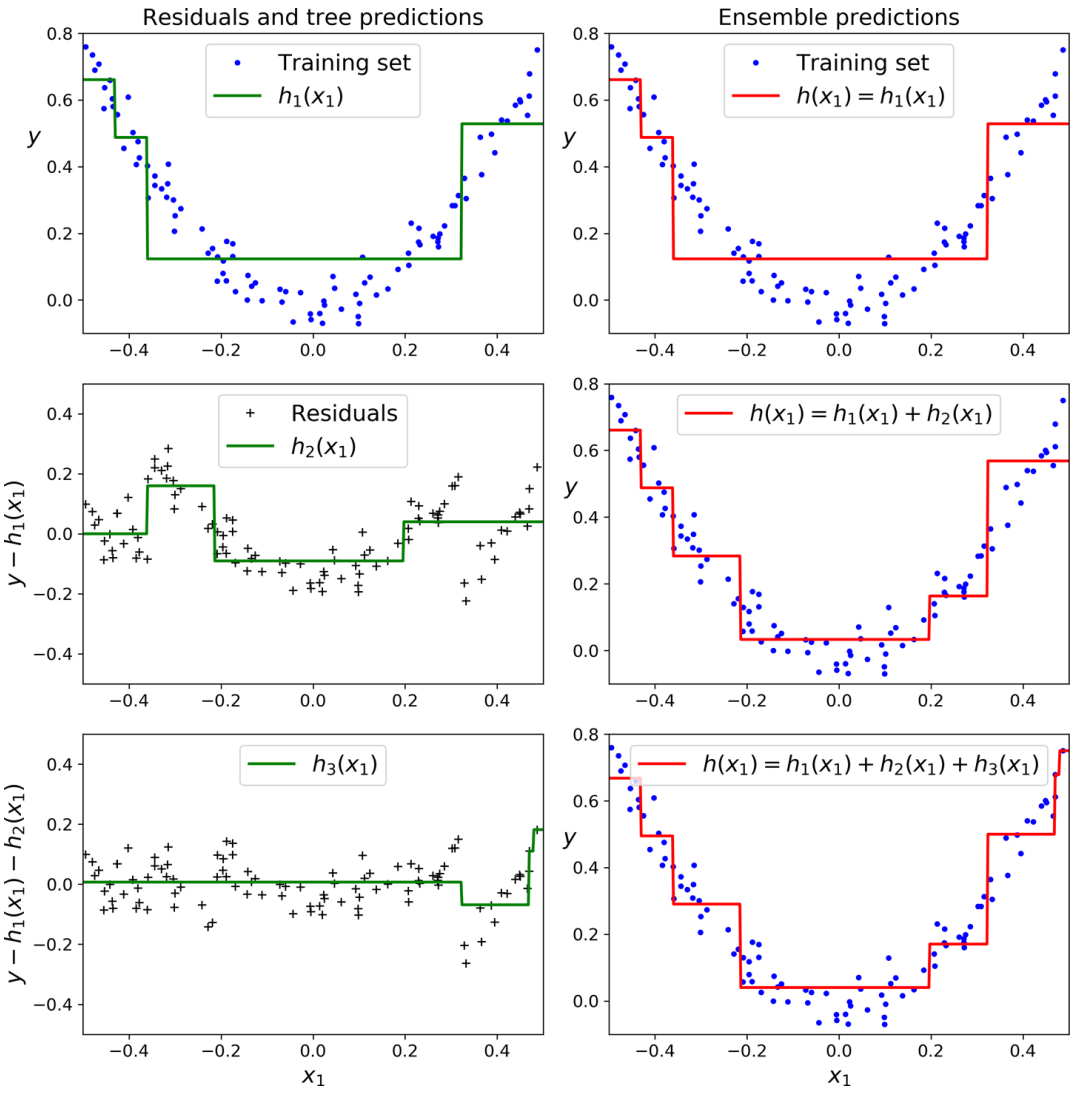
Gradient boosted trees

XGBoost → here

Gradient Boosting works by sequentially adding predictors trying to fit the new predictor to the *residual errors* made by the previous predictor.

Prone to overfitting!

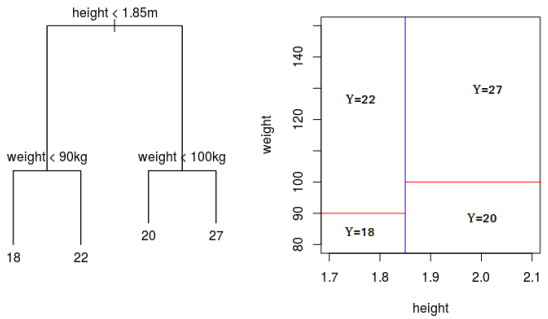
Boosting


Residuals and tree predictions
Ensemble predictions
Training set
$h_1(x_1)$
$h(x_1) = h_1(x_1)$
Residuals
$h_2(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1)$
$h_3(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$
y
$y - h_1(x_1)$
$y - h_1(x_1) - h_2(x_1)$
x_1

Summary

CART

Classification & regression trees



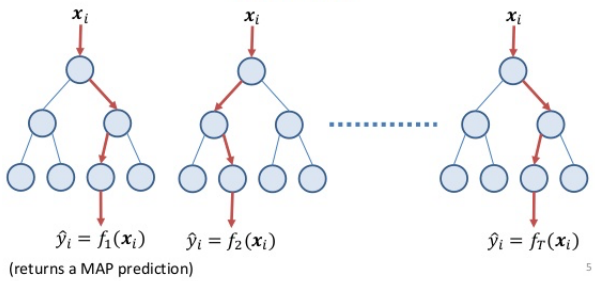
Rec. bin splitting

Error measures for each node split decision.

- *Squared error*
- *Gini index*
- *Entropy*

Random forest

Bagging plus feature sampling



Going further

Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning – Criminisi et al 2011

https://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf. – pages 531 - 558