Tree-based methods

La Serena Data School

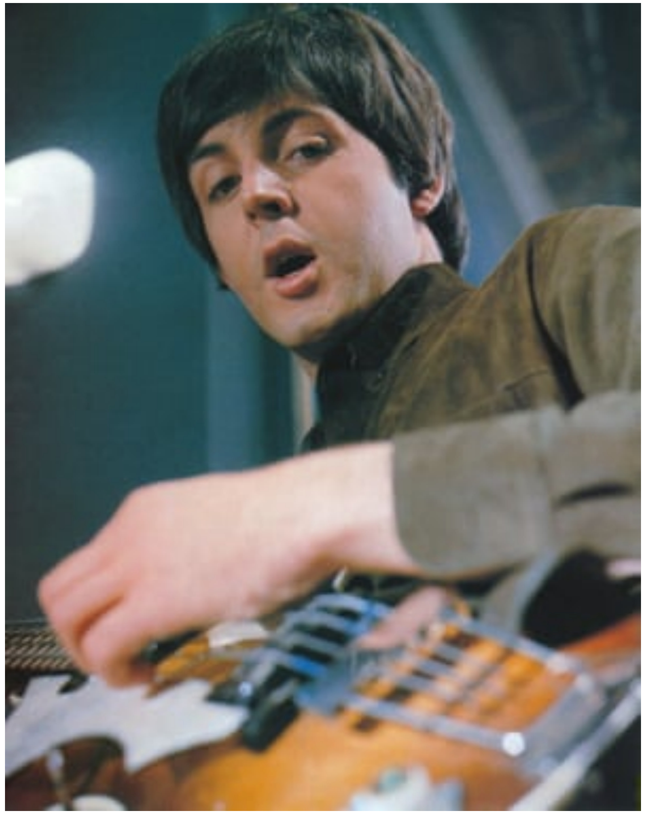
Jocelyn Dunstan
University of Chile

August 2019

Announcement: Saturday’s session is on natural language processing

First time in LSSDS!

Teaching tree-based methods in LSSDS might feel like playing the bass guitar in a band...



But trees can be as worthy as Paul McCartney playing the bass!

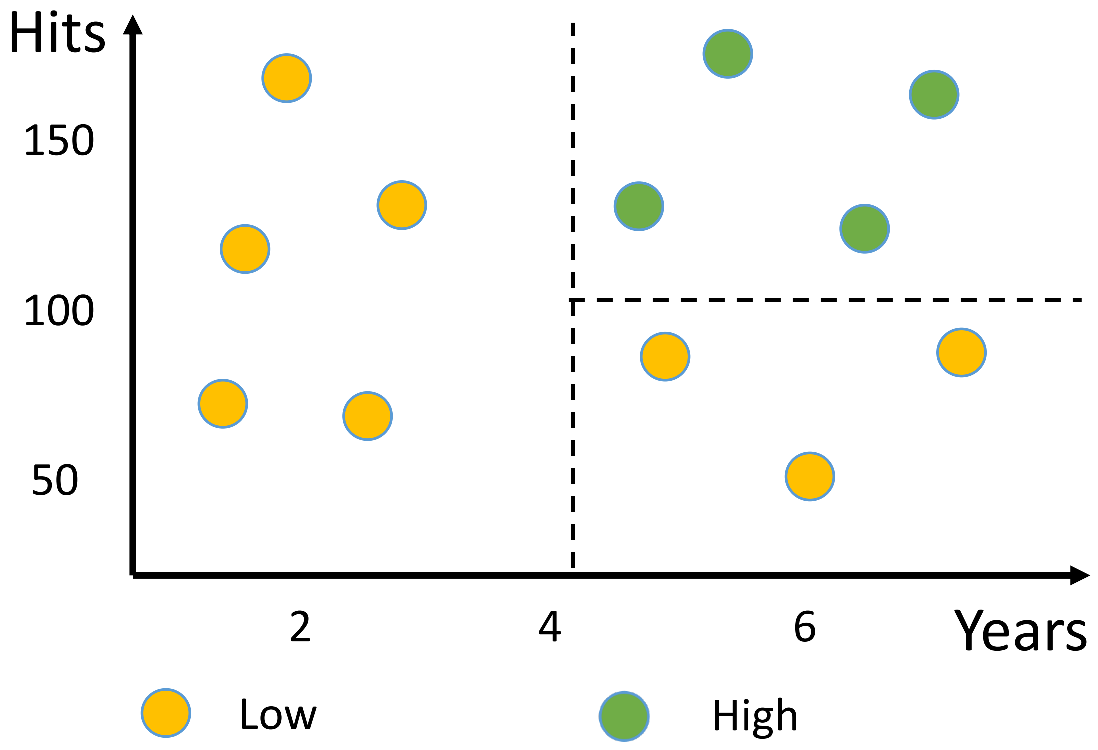
Also Pavlos Protopapas used to teach this lecture!

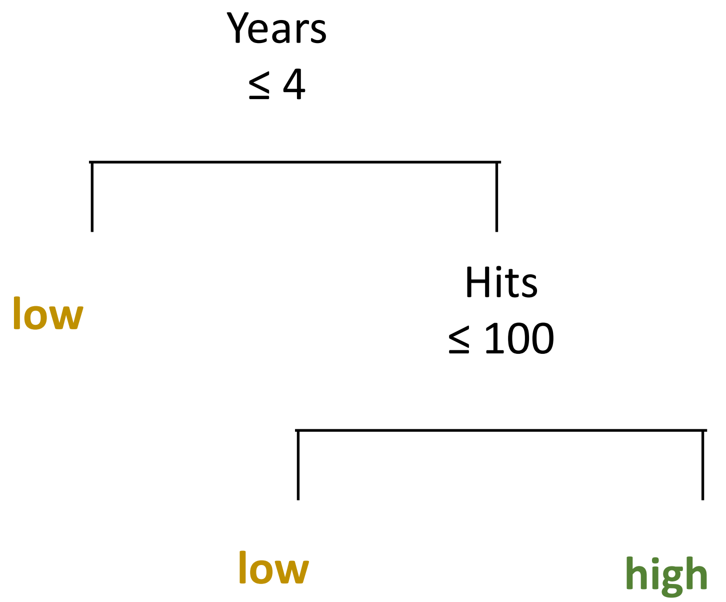
https://harvard-iacs.github.io/2018-CS109A/

So let’s have a plan to learn trees!

- Separation of the predictor’s space
- Tree structure and special features
- Aggregating trees
- XGBoost and AdaBoost
- Hands-on

Idea behind trees:
Segmentation of predictor space





Adapted from Introduction to Statistical Learning by James, Witten, Hastie & Tibshirani

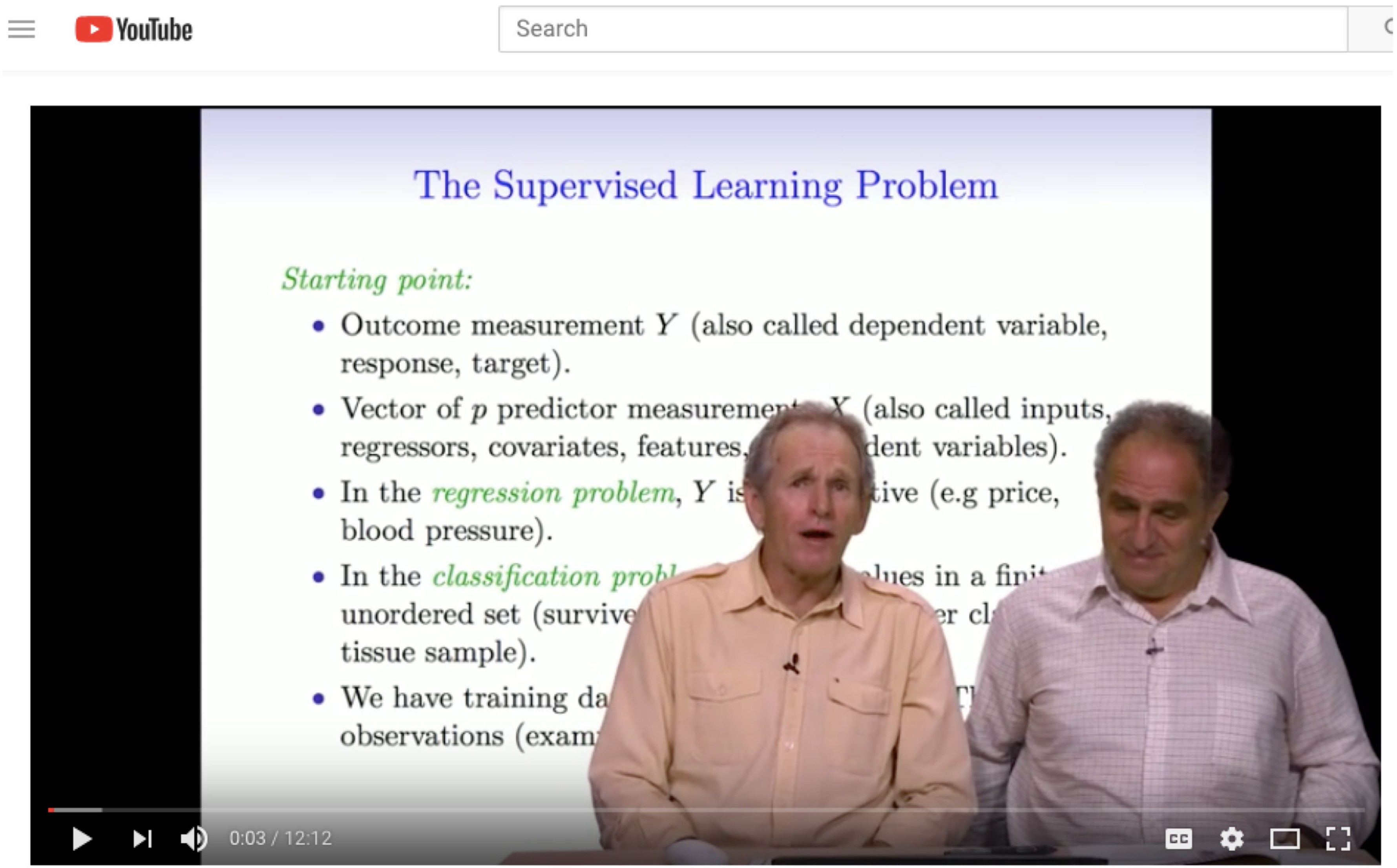
Springer Texts in Statistics

Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

An Introduction to Statistical Learning

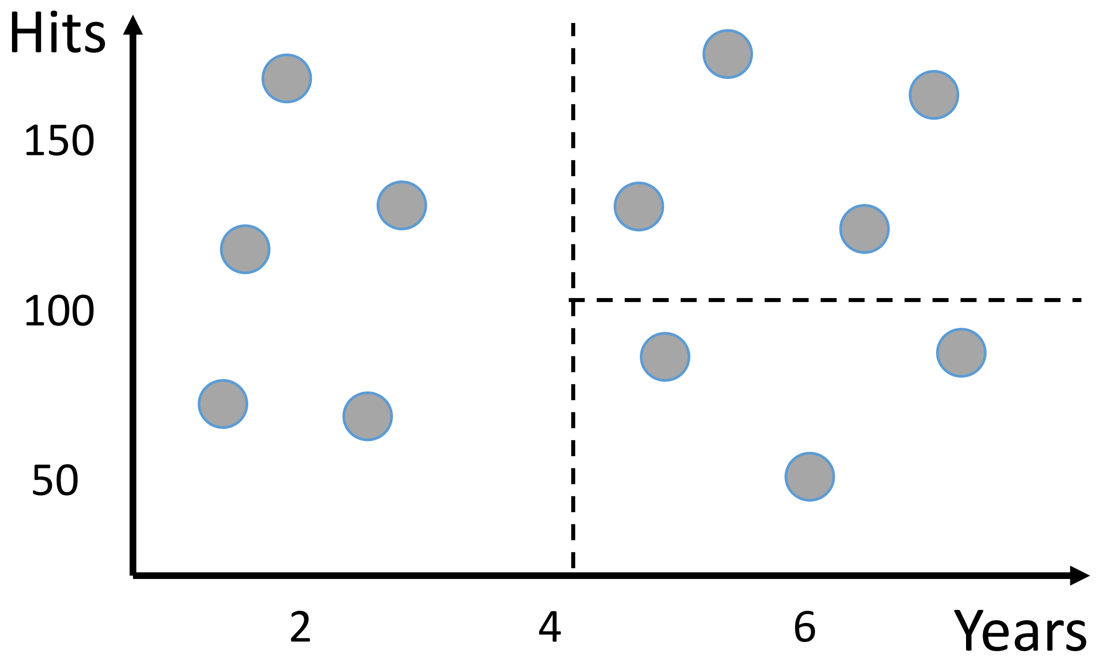
with Applications in R

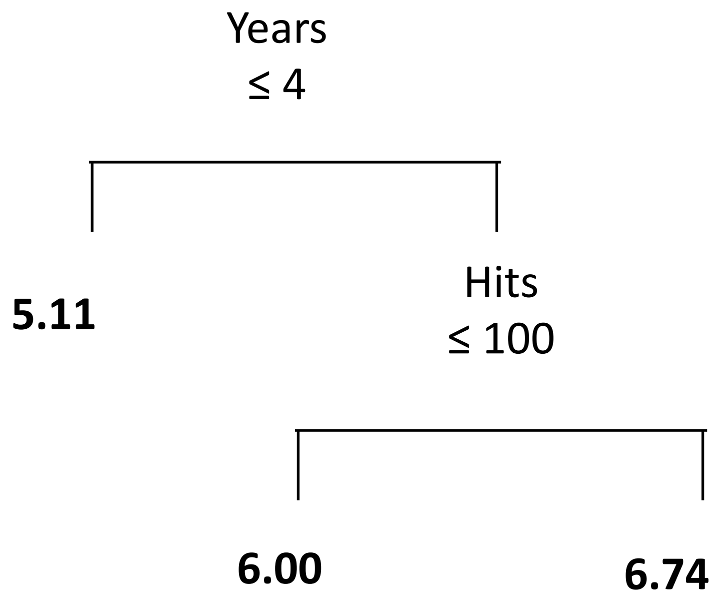
Springer



https://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/

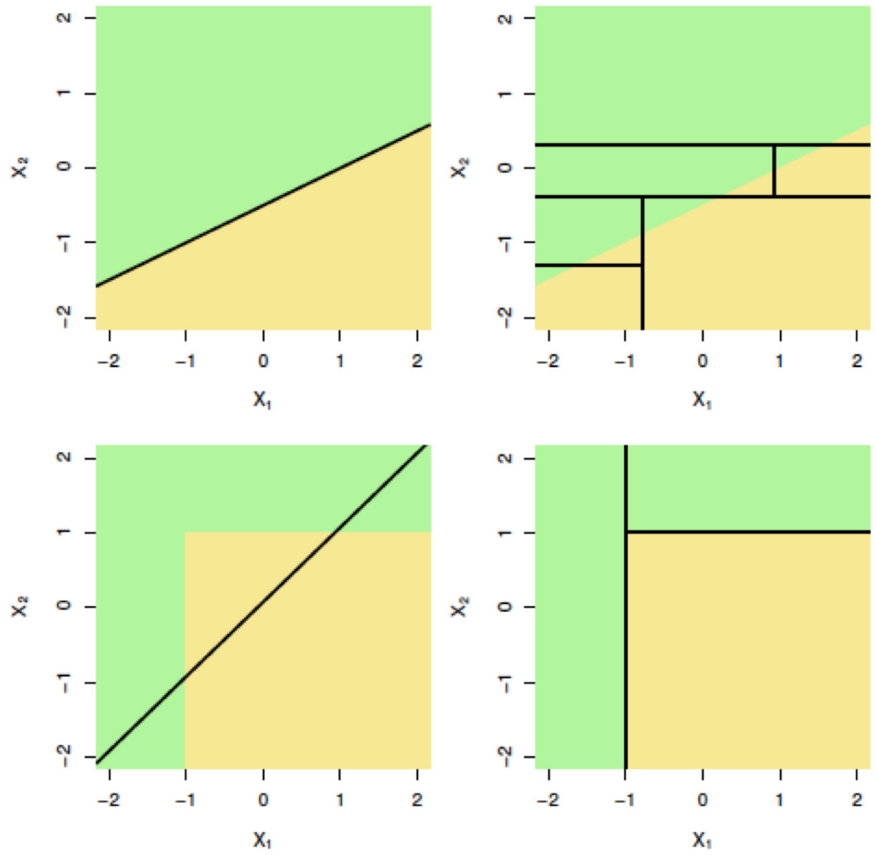
And the same intuition is valid for regression trees





Adapted from Introduction to Statistical Learning by James, Witten, Hastie & Tibshirani

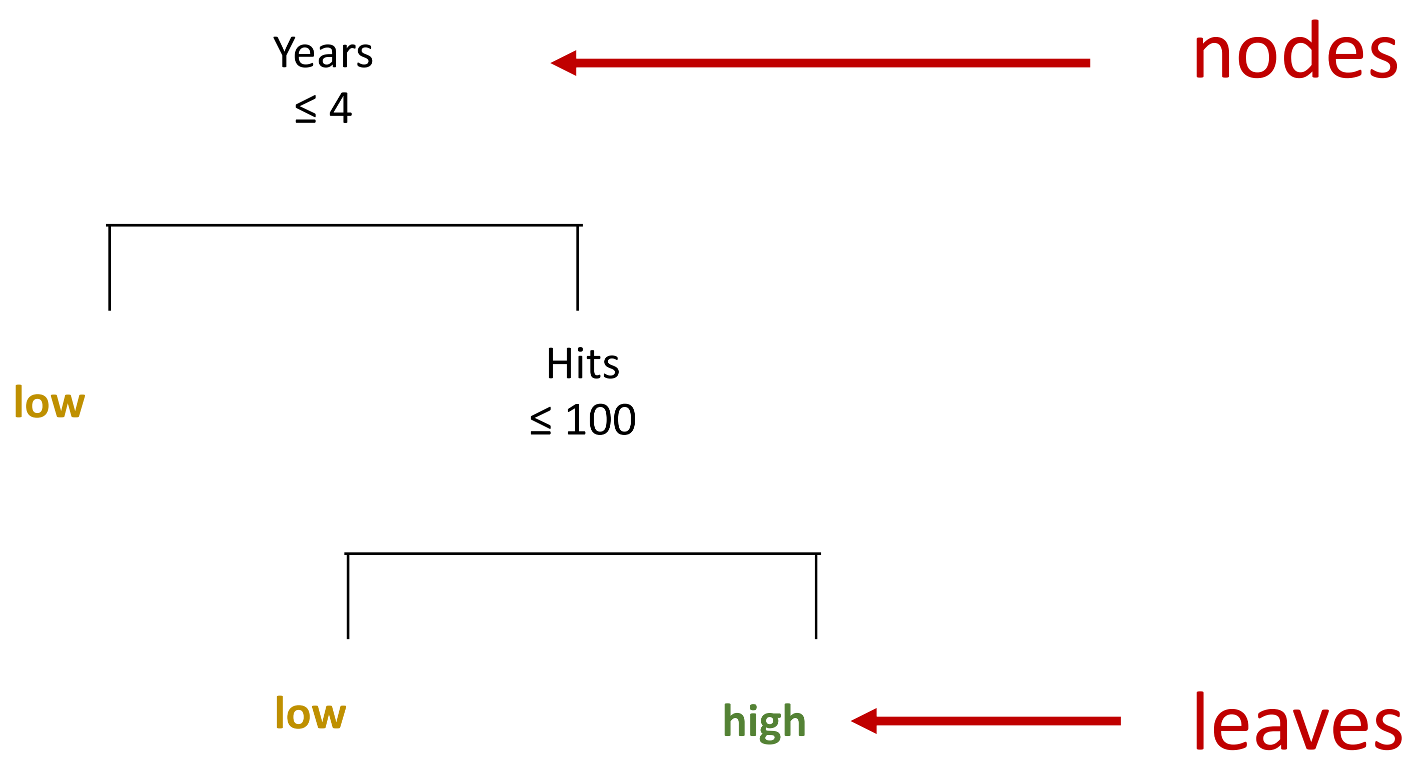
Tree-based compared to linear models



From Introduction to Statistical Learning

The problem is that usually you don't know how your data looks when plotted in a high dimensional space

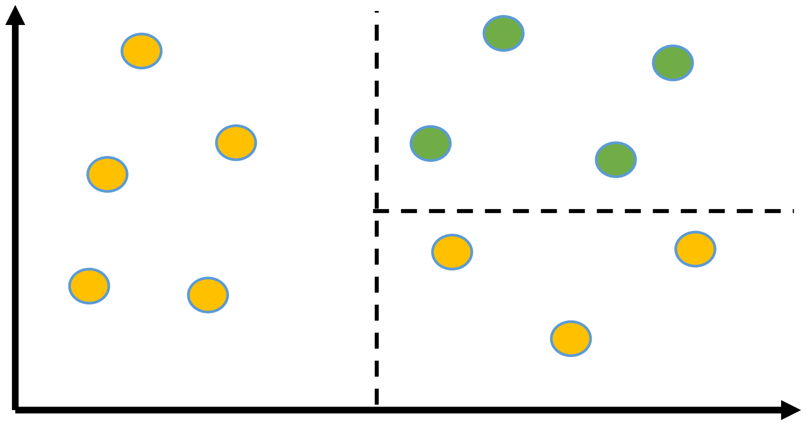
Tree structure



Upside down tree!

Building a tree

- In general, the problem of creating N boxes with different sizes from the data is unfeasible!



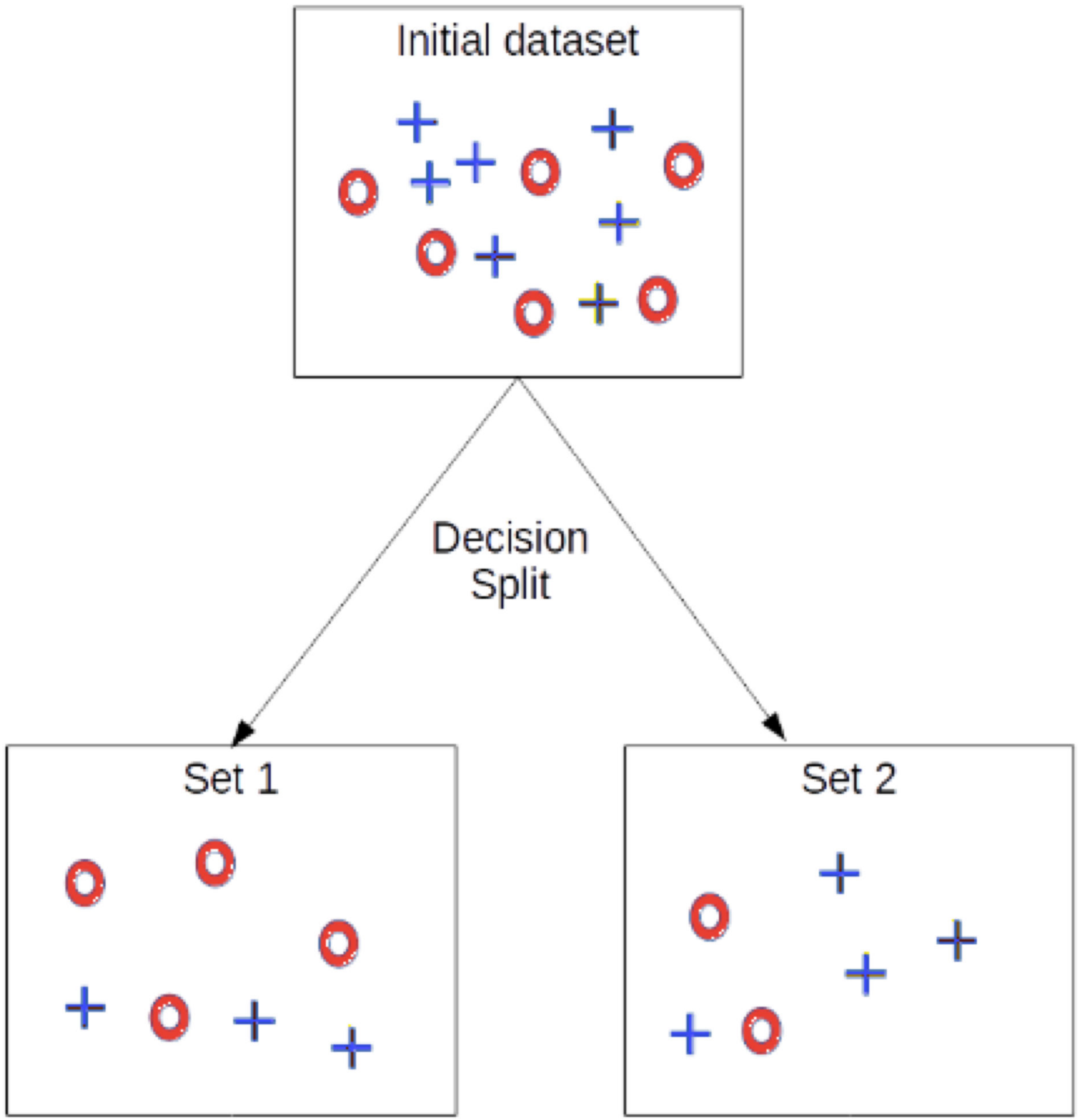
- Trees act **locally**: for a given predictor, find the splitting point

Steps: classification trees

Pick a predictor → Find a splitting point that reduce **entropy** or **impurity** (Gini)

Ways to stop splitting:

- Maximum depth
- Certain function less than a value
- Number of samples in each terminal node


Initial dataset
Decision
Split
Set 1
Set 2

Gini index

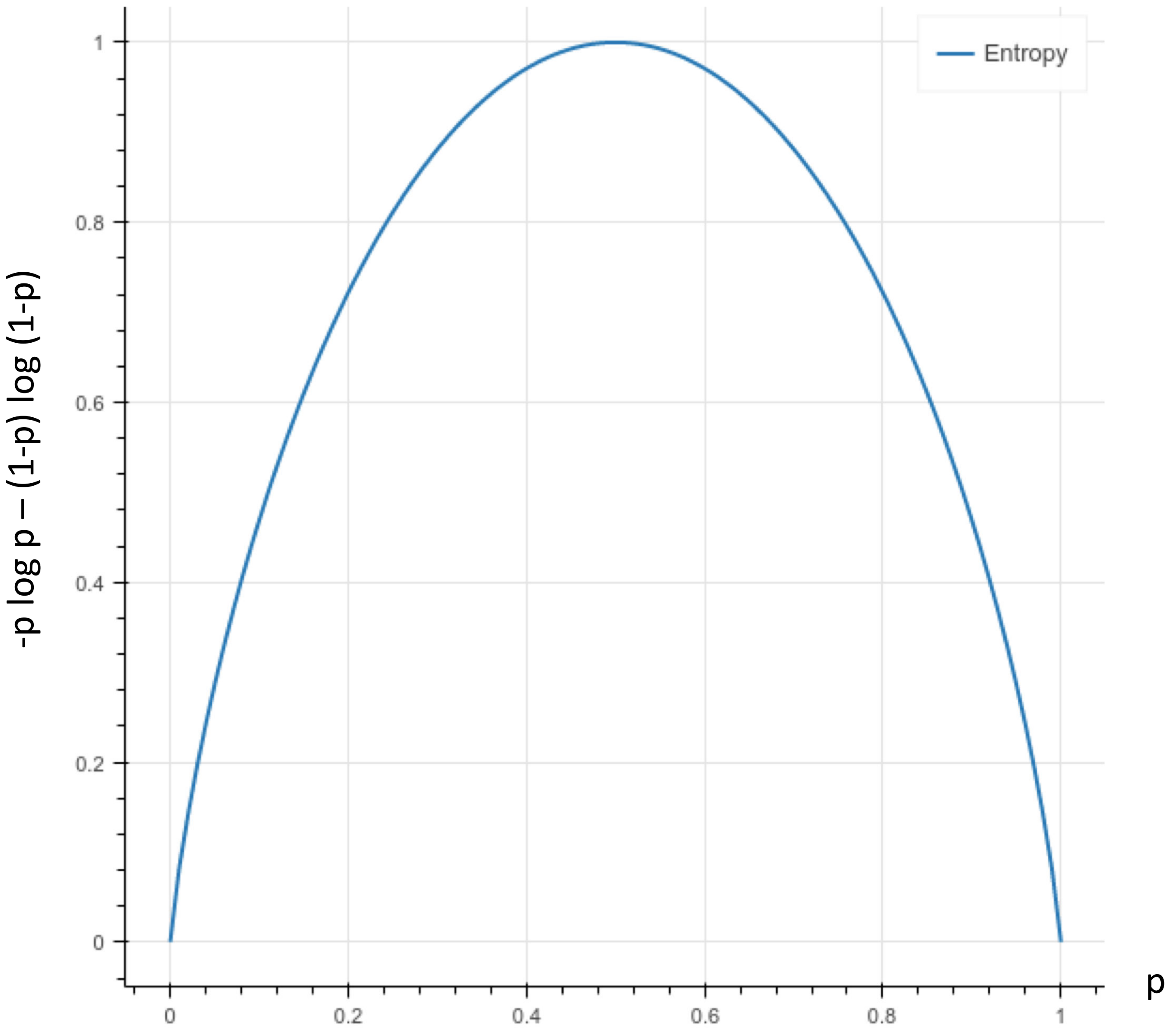
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Where p_{mk} is the proportion of training observations in the m^{th} region from the k-class

Shannon’s entropy

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information gain



-p log p – (1-p) log (1-p)
p
Entropy
0
0.2
0.4
0.6
0.8
1

	Class 1	Class 2	Entropy$(i\|j, t_j)$
R_1	0	6	$-(\frac{6}{6}\log_2\frac{6}{6} + \frac{0}{6}\log_2\frac{0}{6}) = 0$
R_2	5	8	$-(\frac{5}{13}\log_2\frac{5}{13} + \frac{8}{13}\log_2\frac{8}{13}) \approx 1.38$

https://harvard-iacs.github.io/2018-CS109A/

Evaluation of classification

Confusion matrix

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive,** Type I error
	Predicted condition negative	**False negative,** Type II error	**True negative**

https://en.wikipedia.org/wiki/Confusion_matrix

Evaluation of classification

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR}$$

accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

F1 score

is the harmonic mean of precision and sensitivity

$$F_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

https://en.wikipedia.org/wiki/Confusion_matrix

Evaluation of classification

		Actual class		
		Cat	Dog	Rabbit
Predicted class	Cat	**5**	2	0
	Dog	3	**3**	2
	Rabbit	0	1	**11**

		Actual class	
		Cat	Non-cat
Predicted class	Cat	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

https://en.wikipedia.org/wiki/Confusion_matrix

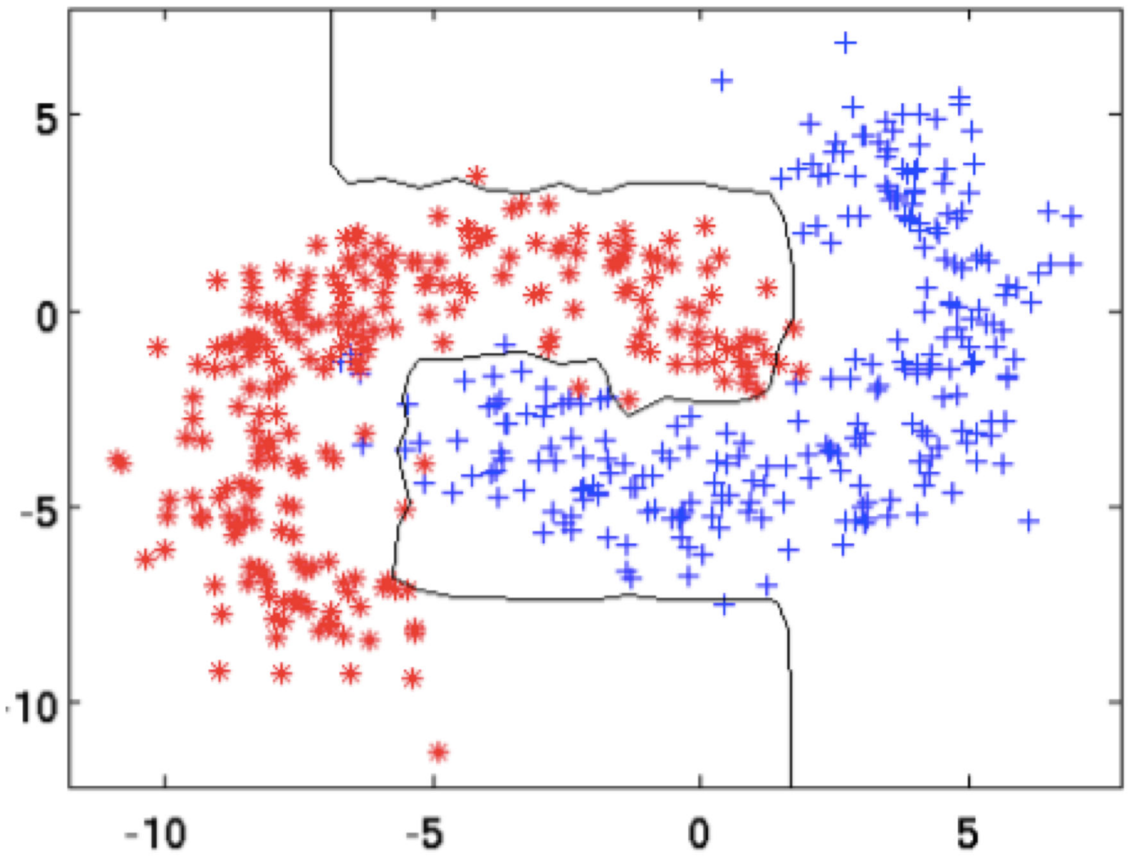
Evaluation of regression

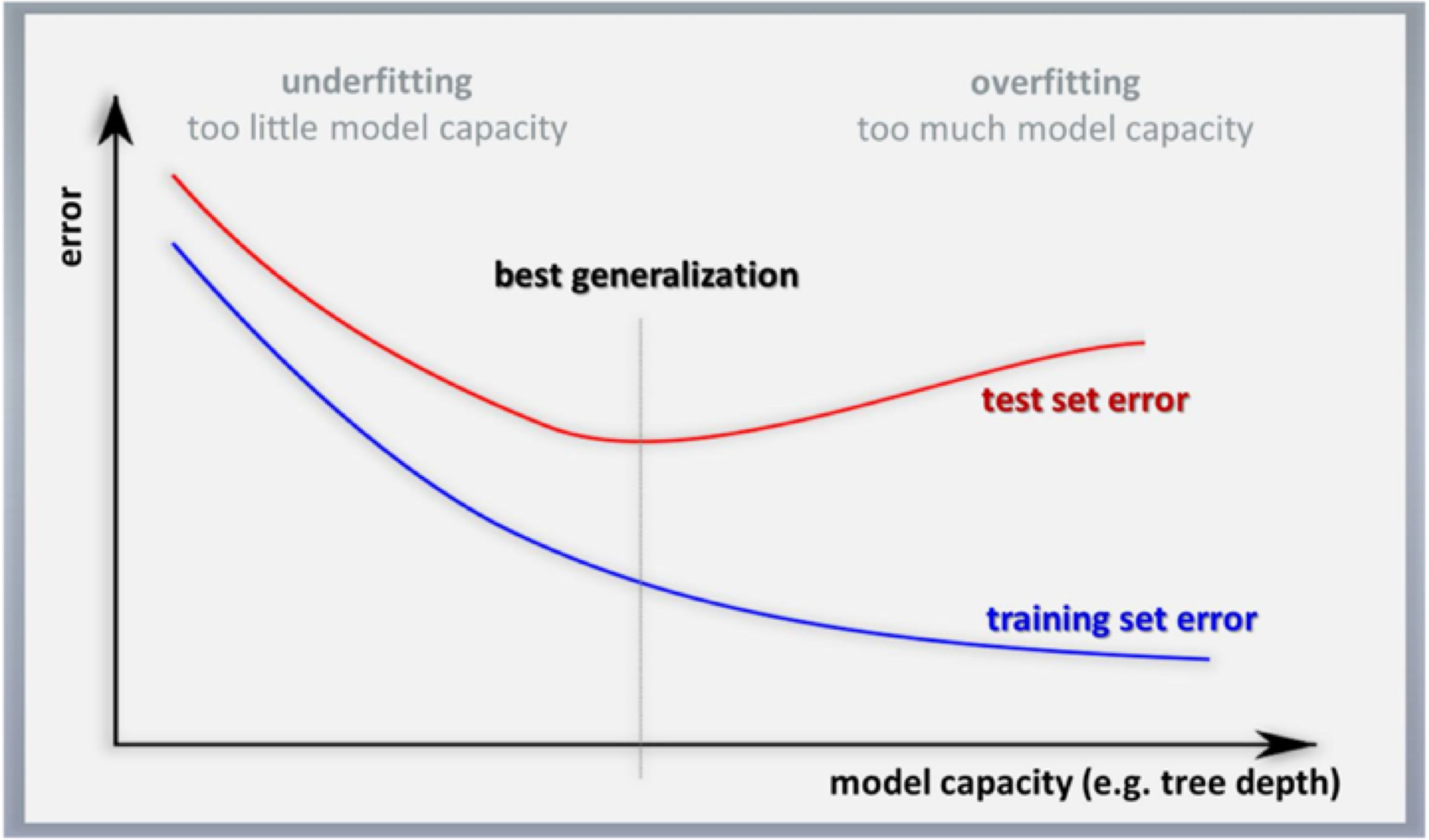
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Minimize root-square error (RSS), where R_J is the region J

Pruning the tree

Trees with too many branches are likely to overfit the data




underfitting
too little model capacity
overfitting
too much model capacity
error
best generalization
test set error
training set error
model capacity (e.g. tree depth)

Tree methods special features

- They work well with small data
- No need to normalize your data before, but you do if you are going to compare with other methods
- Are easy to explain
- Competitive performance, specially when many are averaged.
- Allow inference and dimensionality reduction

Examples of tree-based methods

- Decision trees
- Random Forest
- Xtreme Gradient Boosting (XGBoost)
- AdaBoost

Random Forest

- Decorrelates trees by picking a random selection of m predictors each time (m <n)
- Tuning parameters:

  - Number of trees
  - m predictors
  - When to stop? P-value, entropy, depth

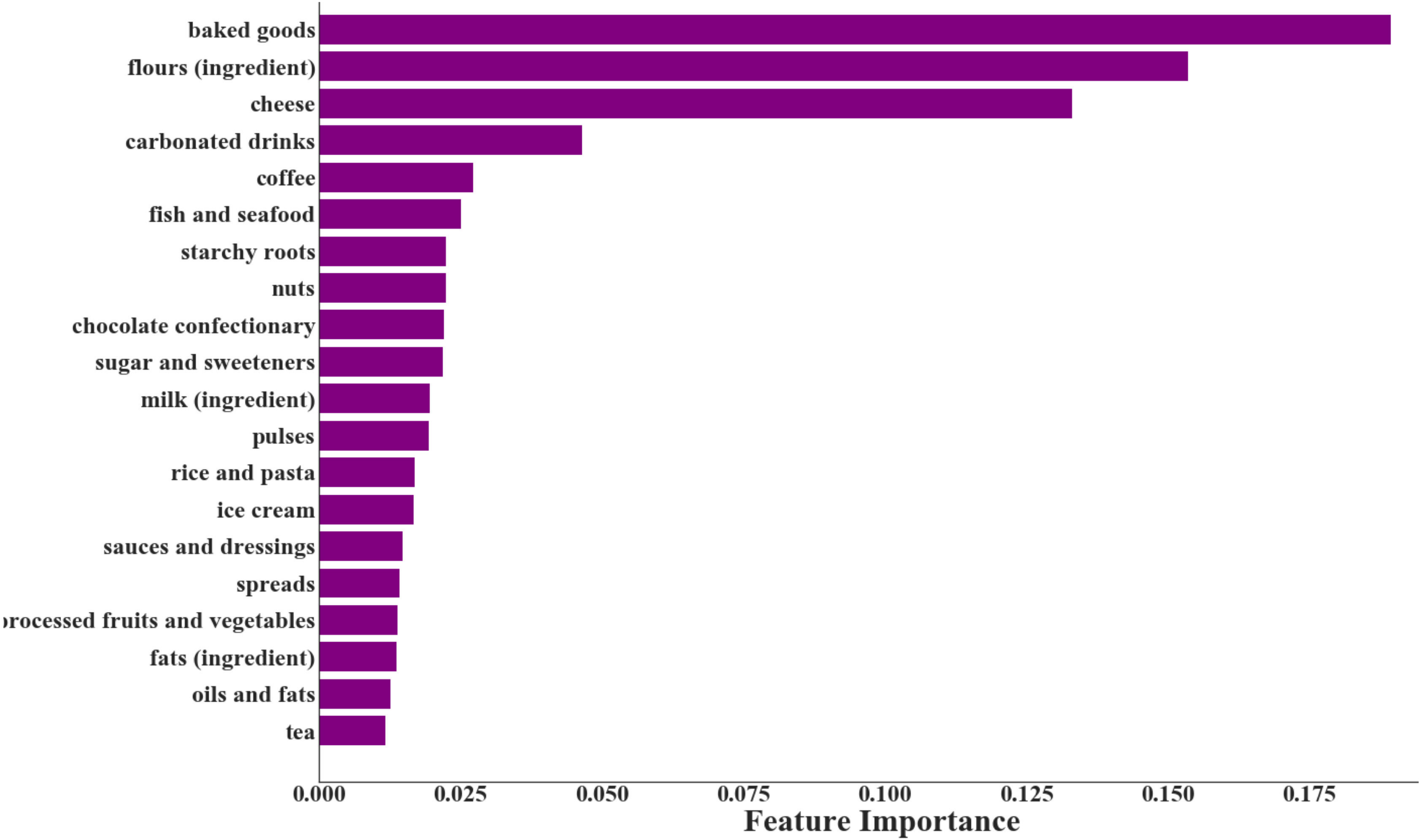
- Variable importance list

Random Forest

- There are default parameters, but in general they should be tuned for the specific training data.
- For example, Breiman's recommendation for m is sqrt(n) for classification and n/3 for regression, but is a parameter that should be explored.

Variable importance list

Is a measure of the decrease accuracy, averaged over all trees, when one predictor is left aside in the model



https://doi.org/10.1177/1460458219845959

Gradient boosting

- Basic idea: we can have a better model by adding single models
- The method works iteratively, adding a single model if compensates weaknesses of the current model

https://harvard-iacs.github.io/2018-CS109A/

Gradient boosting

1. Fit a simple model $T^{(0)}$ on training data, T <- $T^{(0)}$
2. Calculate residuals for T
3. Fit a simple model $T^{(1)}$ for the current residuals
4. T <- T+ λ $T^{(1)}$, where λ is called learning rate (turning parameter)
5. Continue the iteration until stopping condition is met.

https://harvard-iacs.github.io/2018-CS109A/

Gradient boosting

- When we realize that gradient boosting is an example of gradient descent we can import a bunch of knowledge
- For example knowing that for an appropriate choice of λ, the iterative process will eventually converge if the function is convex
- In this case the function to minimize is the MSE

https://harvard-iacs.github.io/2018-CS109A/

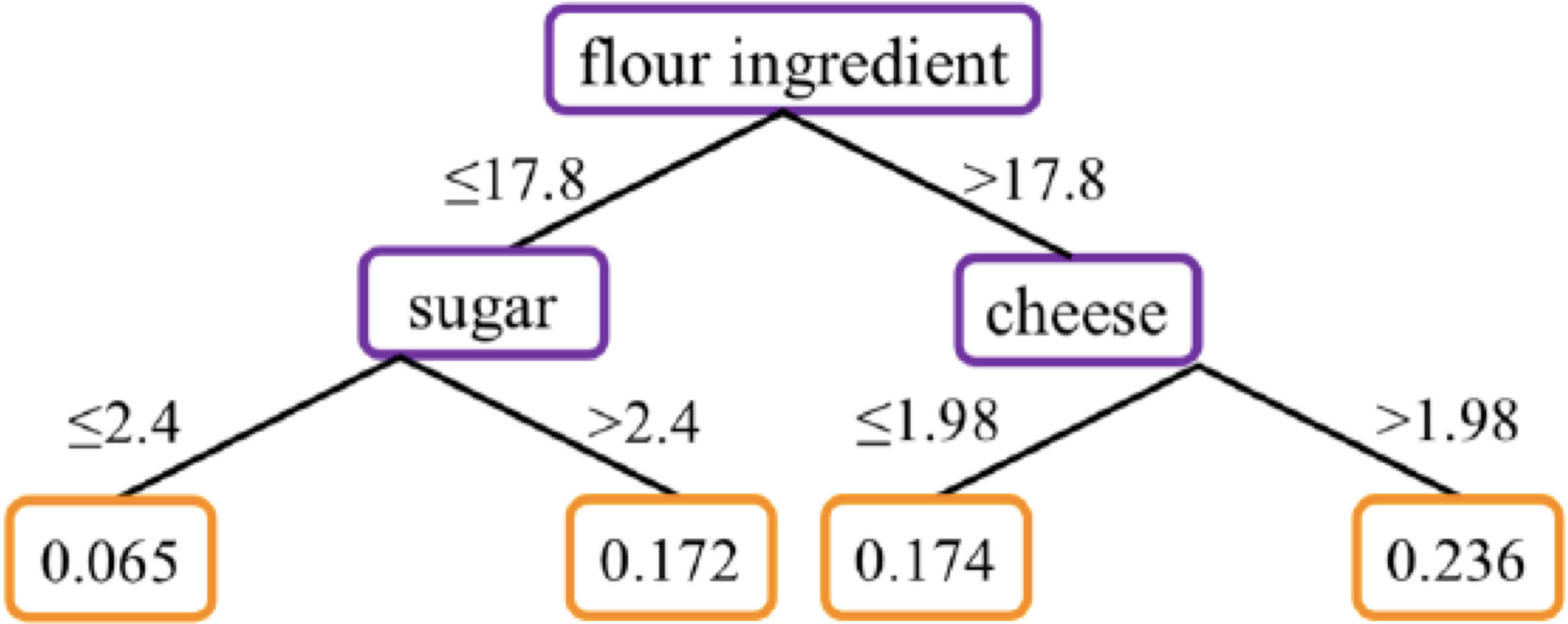
AdaBoost

- Can be seen as the analogy of gradient boosting for classification
- Since the function error in classification is not differentiable (is either 0 or 1), we can use the exponential loss:

$$\text{ExpLoss} = \frac{1}{n}\sum_{k=0}^{n} \exp(-y_n \hat{y}_n)$$

https://harvard-iacs.github.io/2018-CS109A/

Hands on!



flour ingredient
≤17.8
>17.8
sugar
cheese
≤2.4
>2.4
≤1.98
>1.98
0.065
0.172
0.174
0.236