INTRODUCTION TO MACHINE LEARNING

INTRODUCTION

Andreu Arderiu

NORTHIN SUMMER SCHOOL

Barcelona, July 2022

Supervised learning: overview

- **Supervised**: Models are trained with input/output pairs (X, y) which we relate via a function $y = f(X)$. Model learns f to make predictions on new data inputs X.
  - **Classification:** predictions/outputs y are discrete (class labels)
  - **Regression:** y is continuous

- Tasks:
  - Is this image a cat, dog, car?
  - Is this email spam?
  - What would be the price of this house?

Supervised learning: algorithms

- k-NN (k nearest neighbours)
- Naive Bayes
- Linear + logistic regression
- Support vector machines
- Random forests
- Supervised neural networks
- etc…

Today!

Bayes rule overview

- P(A|B): "**Posterior**", probability that A happens given that the evidence B already happened
- P(B|A): **"Likelihood",** probability that B happens given that A already happened
- P(A): "**Prior**", probability that A will happen
- P(B): probability that B happens

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes rule quizz

★ $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

We are planning a picnic but the morning is cloudy. **What is the chance of afternoon rain?**

- From historical weather data we know that:
  - P(Clouds|Rain) = 0.5: 50% of the rainy days were cloudy in the morning
  - P(Rain) = 0.1 : 10% of the days it rained in the afternoon
  - P(Clouds) = 0.4: 40% of the days are cloudy

$P(Rain|Clouds) \overset{★}{=}$?

Bayes rule quizz

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

We are planning a picnic but the morning is cloudy. **What is the chance of afternoon rain?**

- From historical weather data we know that:
  - P(Clouds|Rain) = 0.5: 50% of the rainy days were cloudy in the morning
  - P(Rain) = 0.1 : 10% of the days it rained in the afternoon
  - P(Clouds) = 0.4: 40% of the days are cloudy

$$P(Rain|Clouds) = \frac{0.1 \times 0.5}{0.4} = 0.125$$ ⇨ **12.5%** chance of rain!

Naive Bayes: Golf example

- Naive Bayes: **classification** algorithm based on Bayes rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
- Dataset (X, y) samples, where
  - ➢ y: class variable (play golf yes/no)
  - ➢ X: features or parameters
- **Given** a new sample $\boldsymbol{X}$ we want to **predict** class $\boldsymbol{y}$

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes

Naive Bayes: Golf example

- $X = (x_1, x_2, x_3, x_4)$, where x_1= outlook, x_2= temperature, …
- $P(y|x_1,\ x_2,\ x_3,\ x_4) \overset{\star}{=} \frac{P(x_1|y)P(x_2|y)P(x_3|y)P(x_4|y)P(y)}{P(x_1)P(x_2)P(x_3)P(x_4)}$
- $y = argmax_y P(y) \sum_{i=1}^{4} \frac{P(x_1|y)}{P(x_1)}$

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes

★

We asume $x_1, x_2, x_3,$ x_4 are independent. e.g. Temperature does not afect humidity

→ “Naive”

Naive Bayes: Pros and cons

- Pros
  - **Easy** to use and **computationally cheap**
  - **If** assumption of **independent features** holds, performs **well** with **few data**

- Cons
  - Features are **rarely independent**
  - **Zero-frequency**: If test set has a categorical variable or category not observed in the training test, the Naive Bayes model Will always (erronously) assign a 0 probability

K-nearest neighbours (k-nn)

- Supervised algorithm that can be used for **classification** or **regression** tasks.
- **Classification**: Given X, compute $y =$ **majority class** of the k nearest neighbours.
- **Regression**: Given X, compute $y =$ **average value** of the k nearest neighbours.

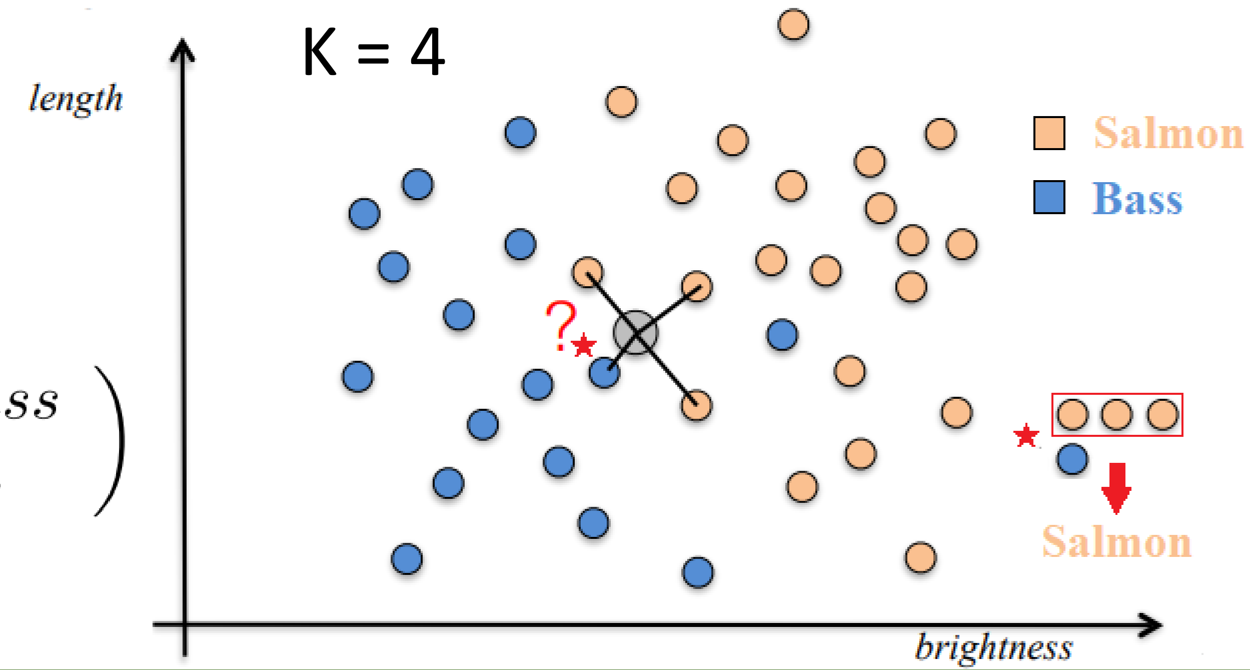
K-nearest neighbours: classification

1. Given a new x, find its **k nearest neighbours** according to some distance measure
2. **Classify** the point according to the **majority of labels** of its nearest neighbours

Is it a bass or a salmon?

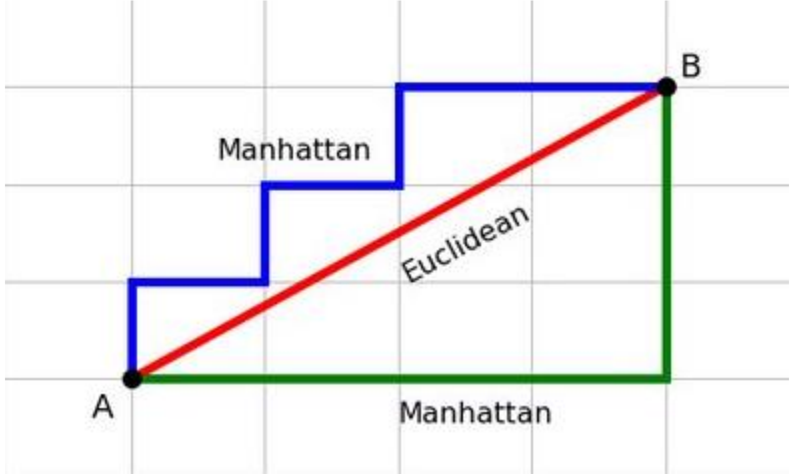
$$\xrightarrow{\textit{Some algorithms}} \quad \mathbf{x} = \begin{pmatrix} brightness \\ length \end{pmatrix}$$

Credit: Bob West, EPFL



k-NN similarity/distance measures

- **Euclidean** distance: Simplest, fast to compute
$$d(x, y) = ||x - y||$$
- **Cosine** distance: Good for documents, images
$$d(x, y) = 1 - \frac{x \cdot y}{||x|| \, ||y||}$$
- **Manhattan** distance: Coordinate-wise distance
$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$
- ...



Source: Omni calculator

Quizz: k-NN, choosing k

- We have a bias/variance tradeoff
- When k increases how does bias and variance change?

- Small k ➡
- Large k ➡

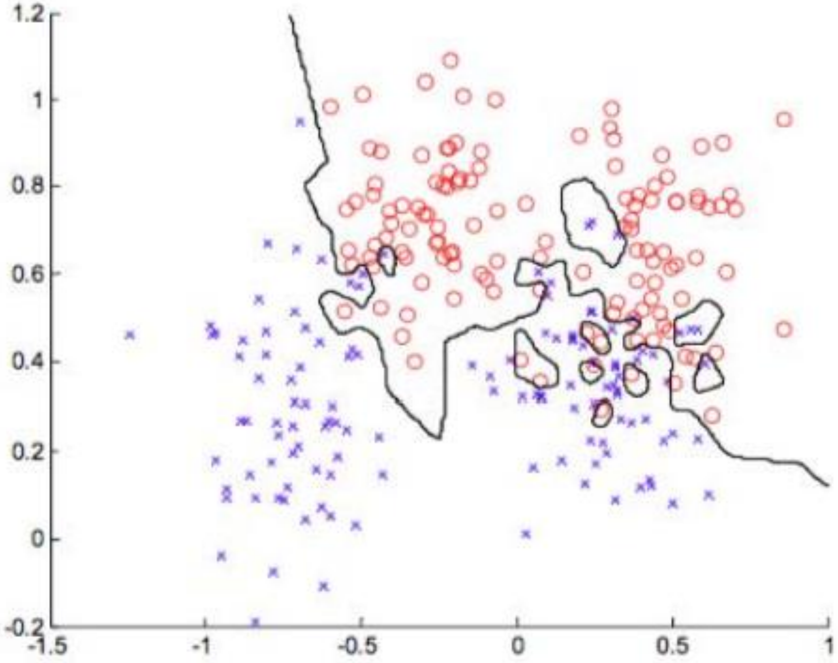
Quizz: k-NN, choosing k

- We have a bias/variance tradeoff
- When k increases how does bias and variance change?

- Small k → low bias, high variance
- Large k → high bias, low variance

k-NN: choosing k, bias-variance tradeoff

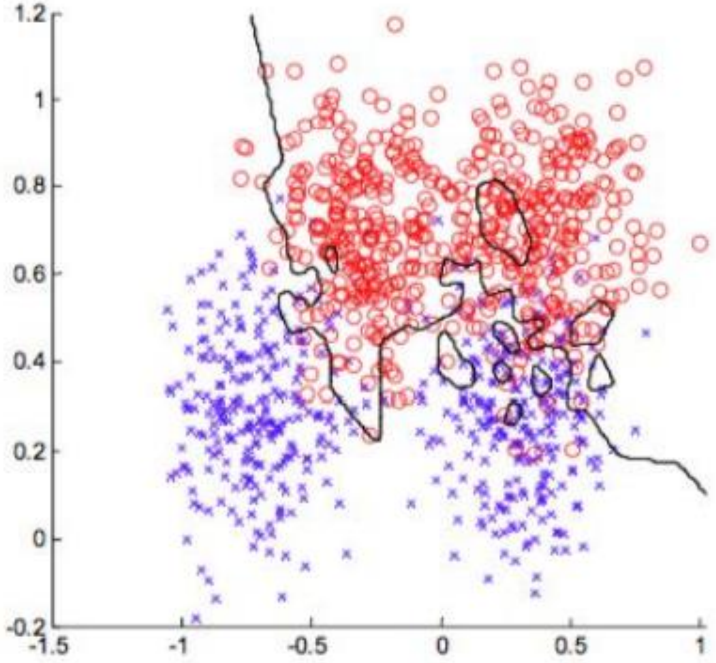
Classification error with k=1 (low bias,high variance)



Training set

error = 0.0

Credit: Bob West, EPFL

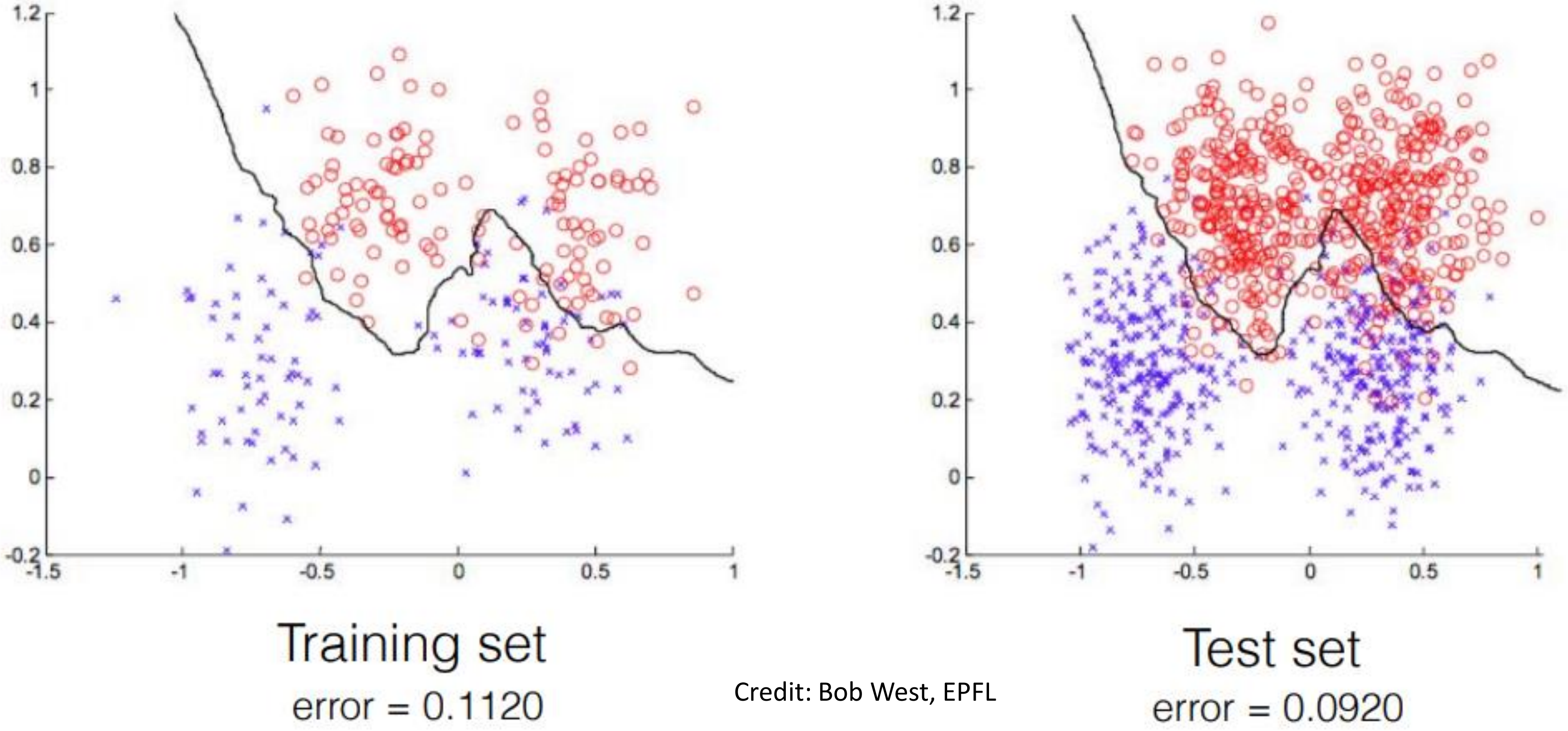


Test set

error = 0.15

K-NN: choosing k, bias-variance tradeoff

Classification error with optimal k=21.



Credit: Bob West, EPFL

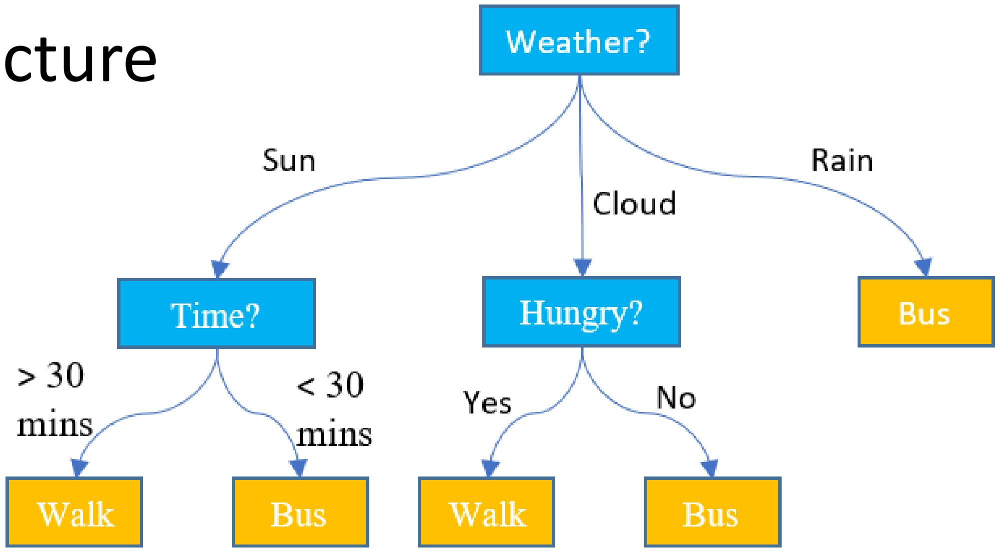
K-NN: pros and cons

- Pros:
  - **Intuitive** and **simple**
  - Used for **classification** (binary and multi-class) and **regression**
  - Just **one hyperparameter** k, **no** need to **train** a model, the data is the model!

- Cons:
  - **Slow**: as dataset size grows, computational cost grows very fast
  - Curse of **dimensionality**: as number of variables grows, performance declines
  - Sensitive to **outliers**

Decision trees: overview

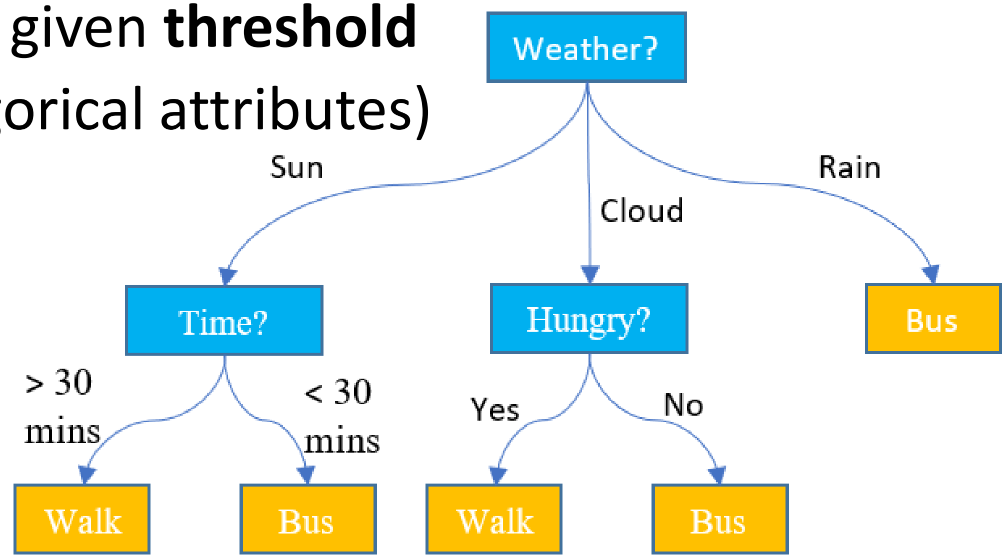
- Supervised algorithm that can be used for **classification** or **regression** tasks.
- The model follows a flow-chart tree structure
  - ➢ Nodes are tests on a single attribute
  - ➢ Branches are attribute values
  - ➢ Leaves are class labels (classification) or output values (regression)



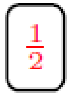
Source: SQLshack

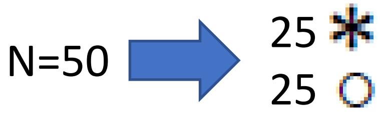
Decision tree induction

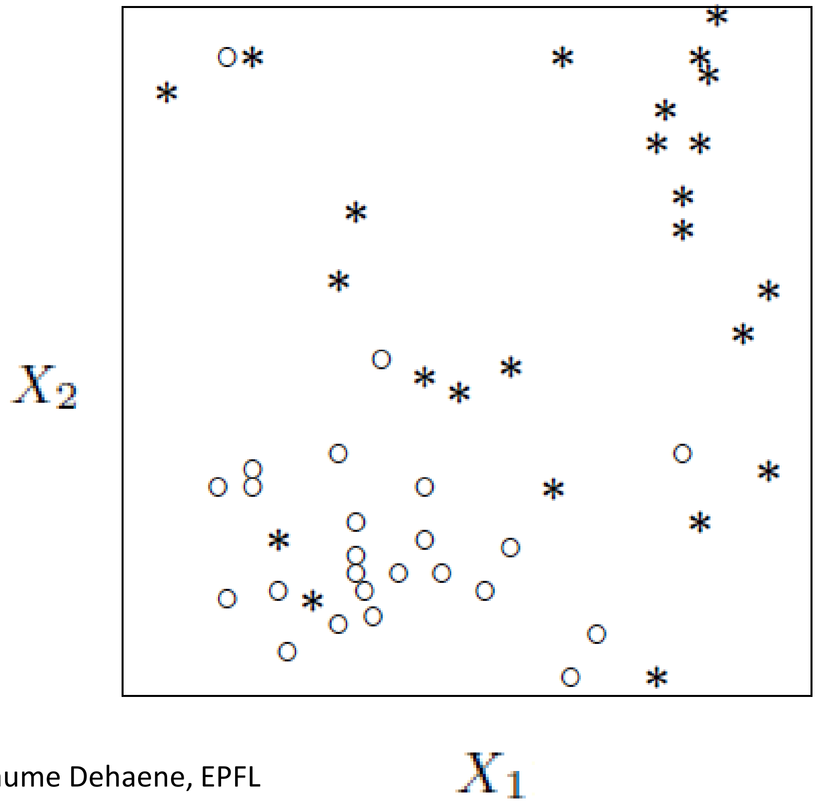
1. Examples are partitioned recursively based on "most discriminative" features
2. Partitioning **stops** if
   - **All** simples belong to the **same class** (classification)
   - **Number** of simples in the partition is **below** a given **threshold**
   - There are no attributes left for splitting (categorical attributes)
   - Maximum depth is reached
3. Leaf predictions
   - Class majority vote (classifications)
   - Local average (regression)



Decision tree: example

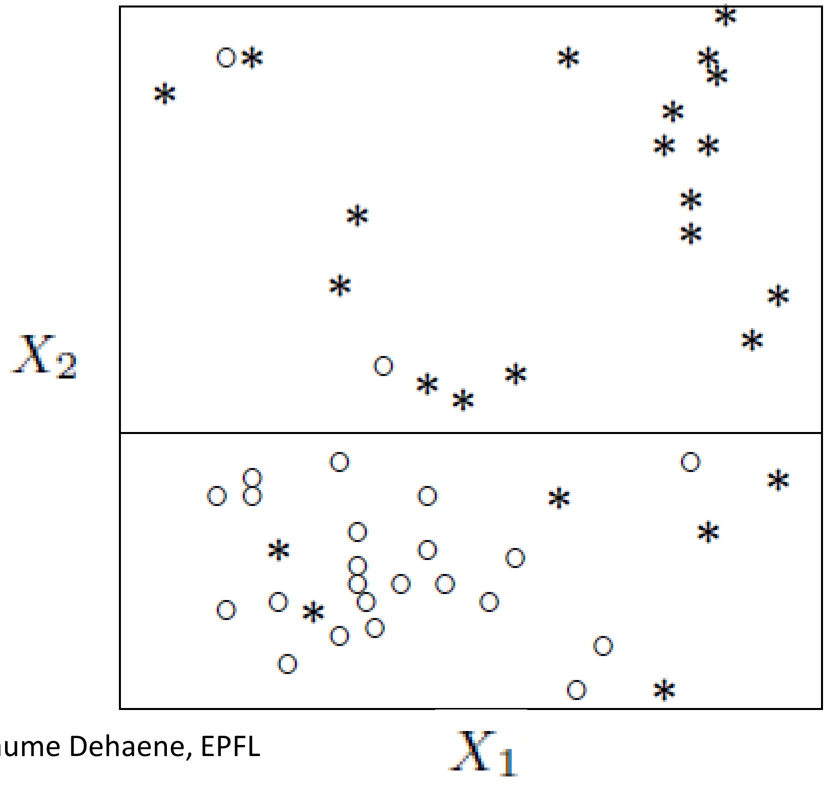




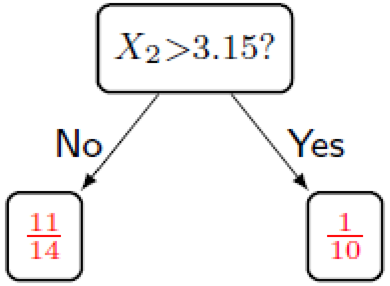


Credit: Guillaume Dehaene, EPFL

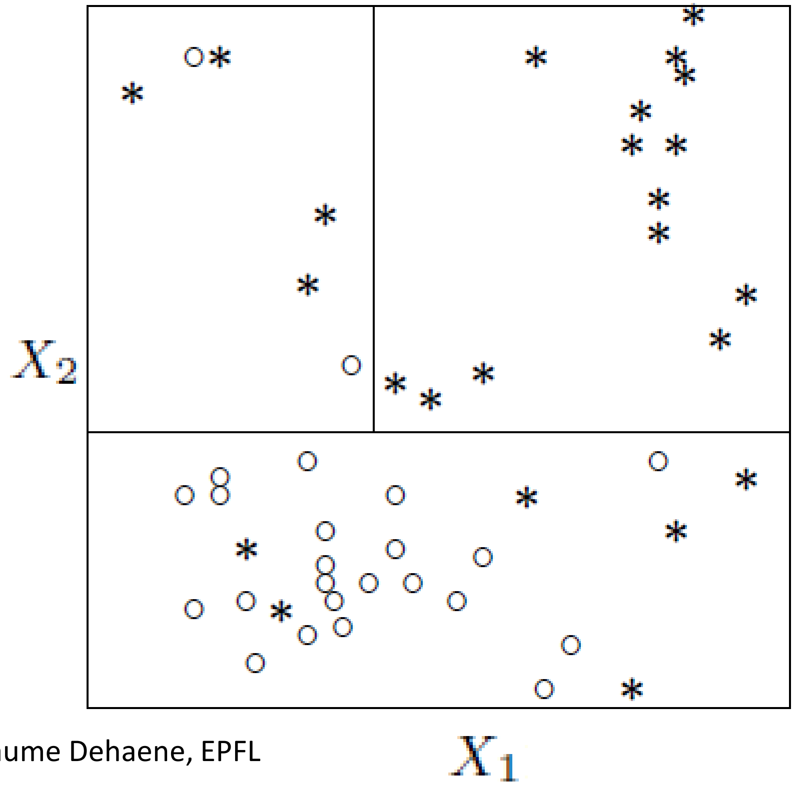
Decision tree: example



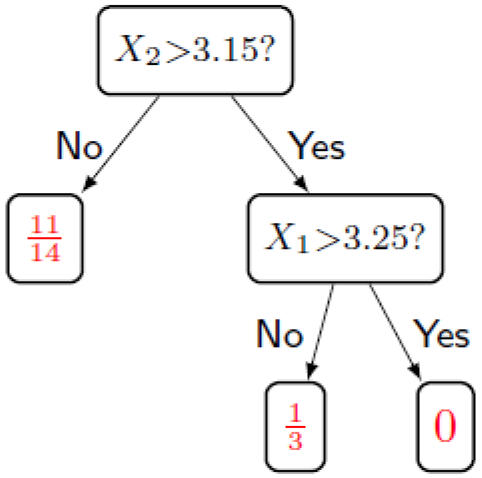
Credit: Guillaume Dehaene, EPFL



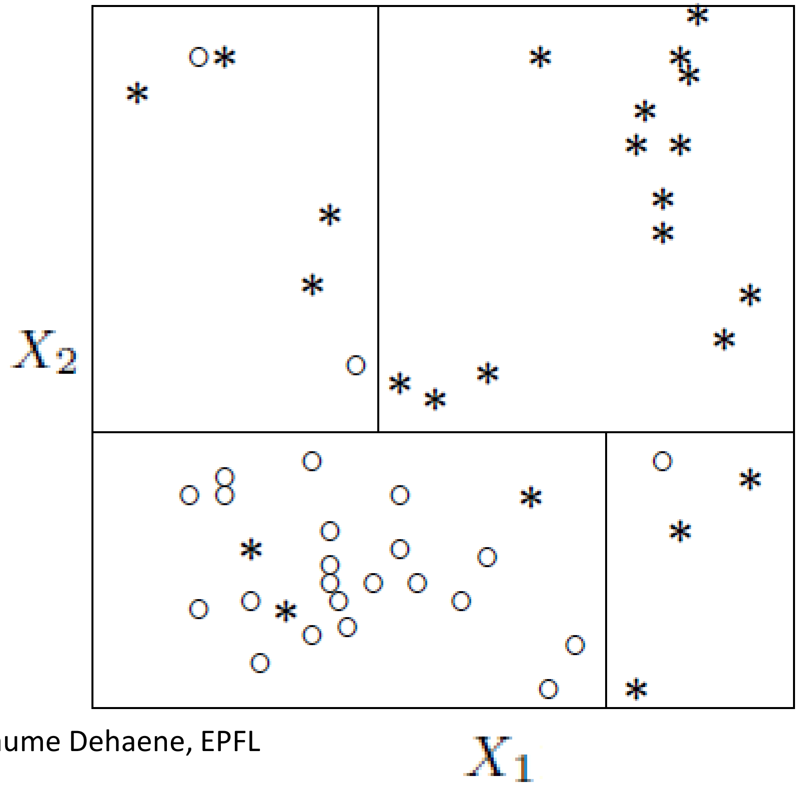
Decision tree: example

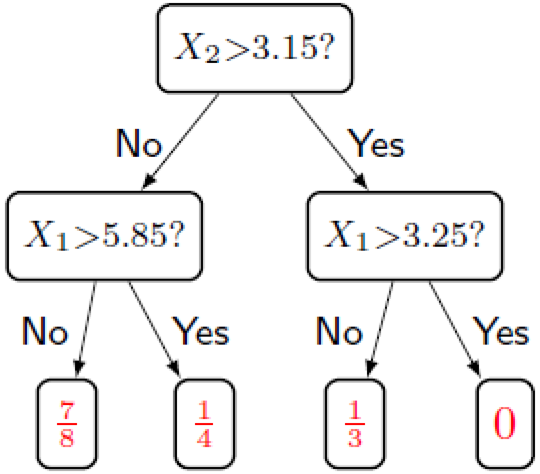


Credit: Guillaume Dehaene, EPFL



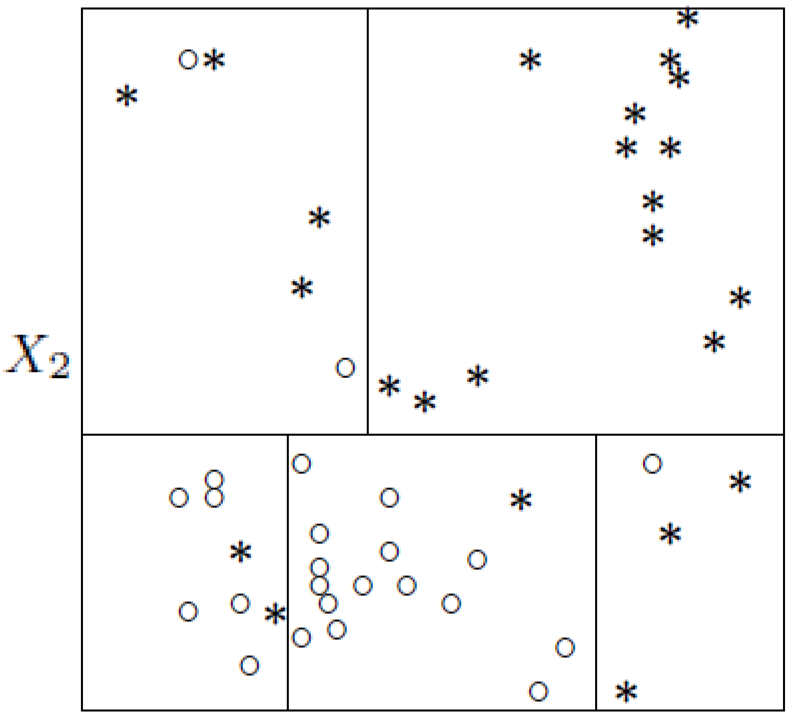
Decision tree: example

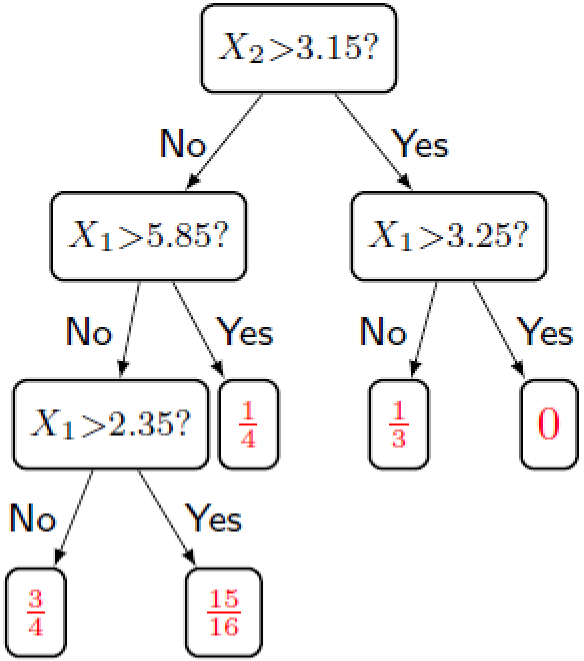




Credit: Guillaume Dehaene, EPFL

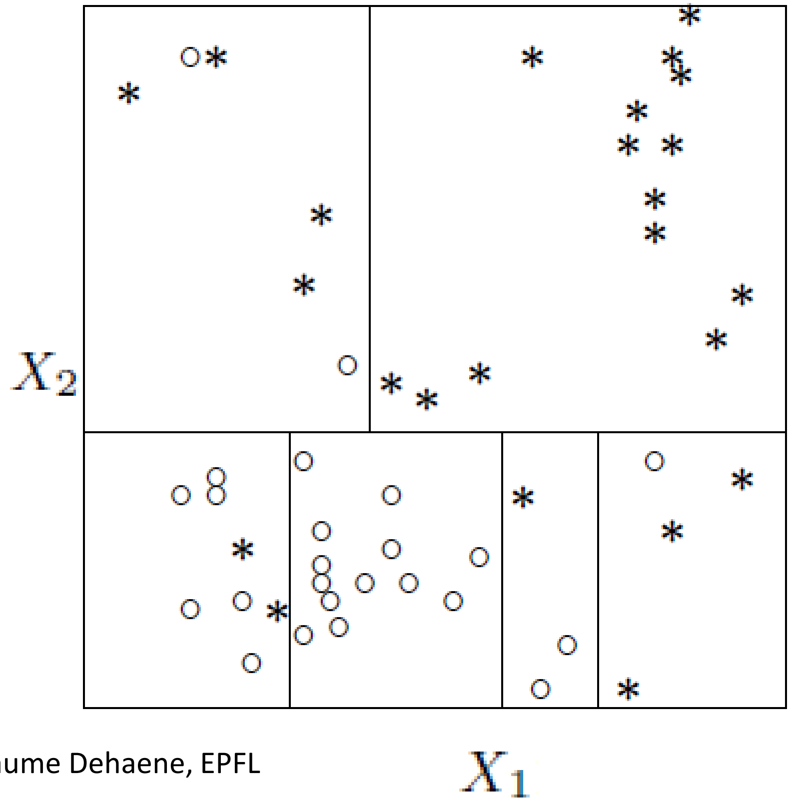
Decision tree: example

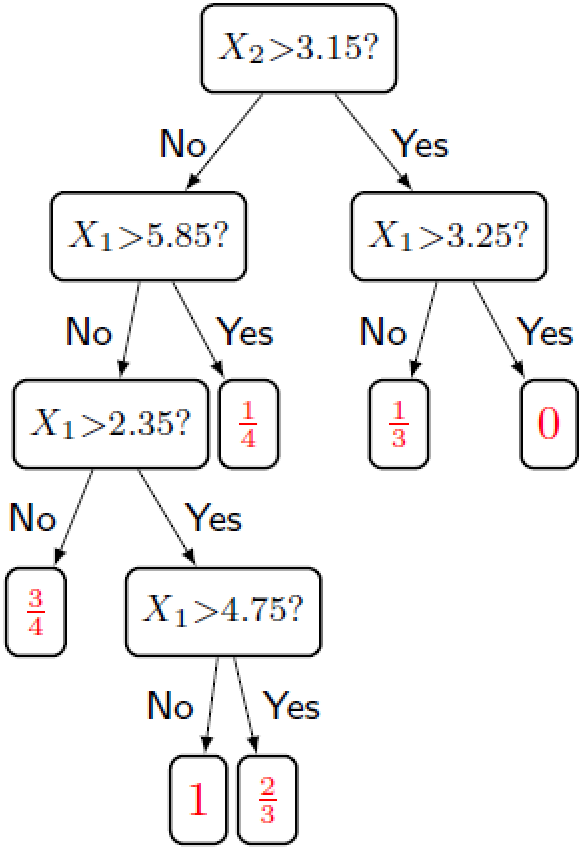




Credit: Guillaume Dehaene, EPFL

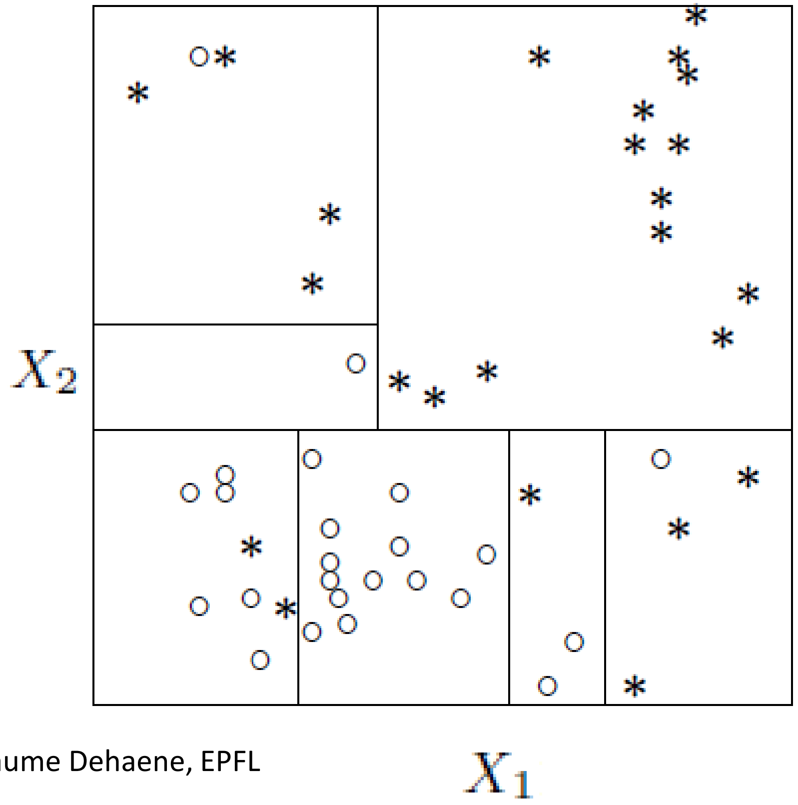
Decision tree: example



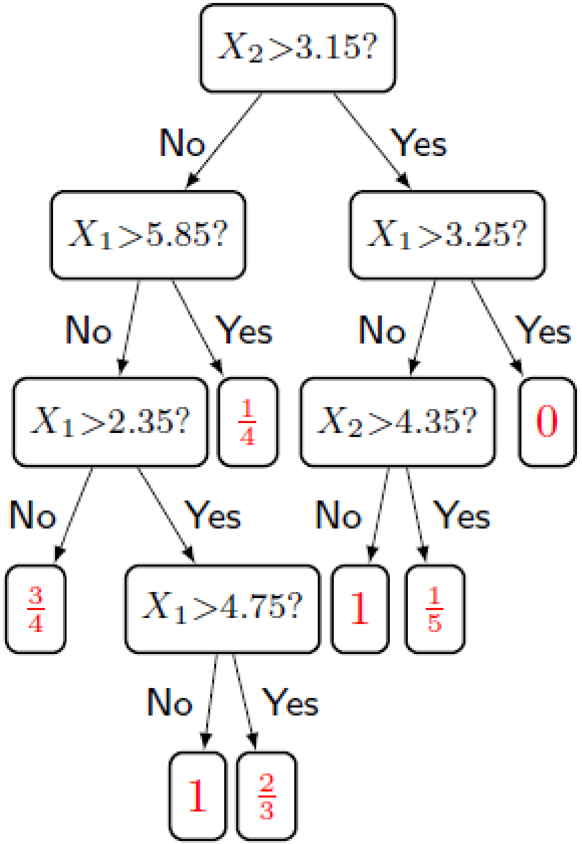


Credit: Guillaume Dehaene, EPFL

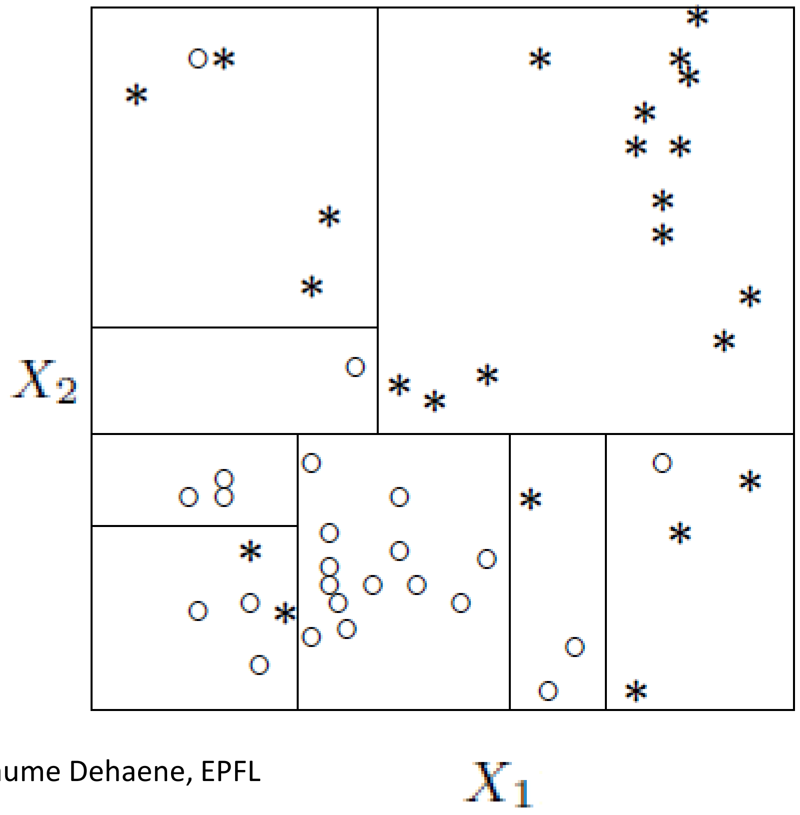
Decision tree: example

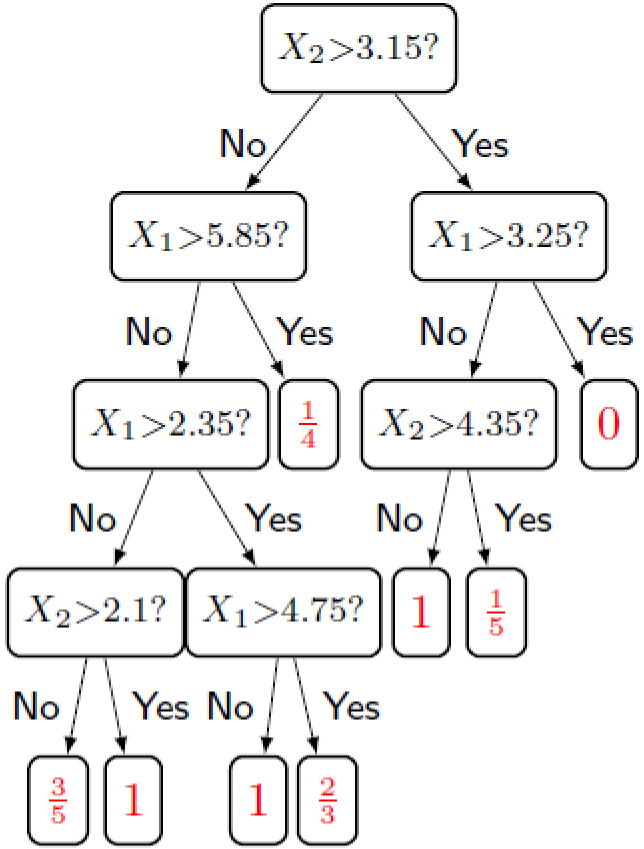


Credit: Guillaume Dehaene, EPFL



Decision tree: example





Credit: Guillaume Dehaene, EPFL

Quizz: bias-variance

- As tree Depth increases, how do bias and variance change?

Quizz: bias-variance

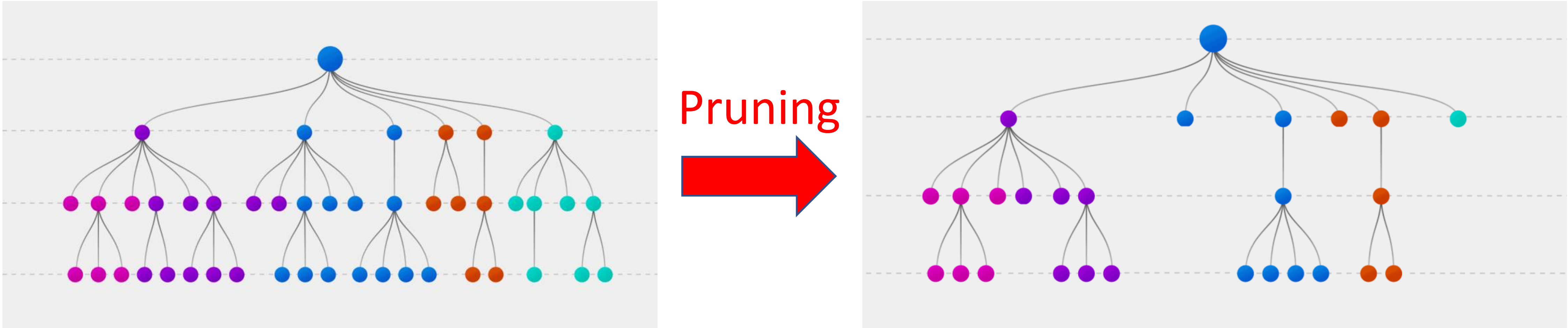
- As tree Depth increases, how do bias and variance change?

Bias decreases
with tree depth

Variance increases
with tree depth

Decision trees: pruning

- **Reduce** the **complexity** of the tree by removing **branches**, **reduce overfitting**!
- Replace nodes with leaves and keep the change if accuracy in **validation set** does not decrease



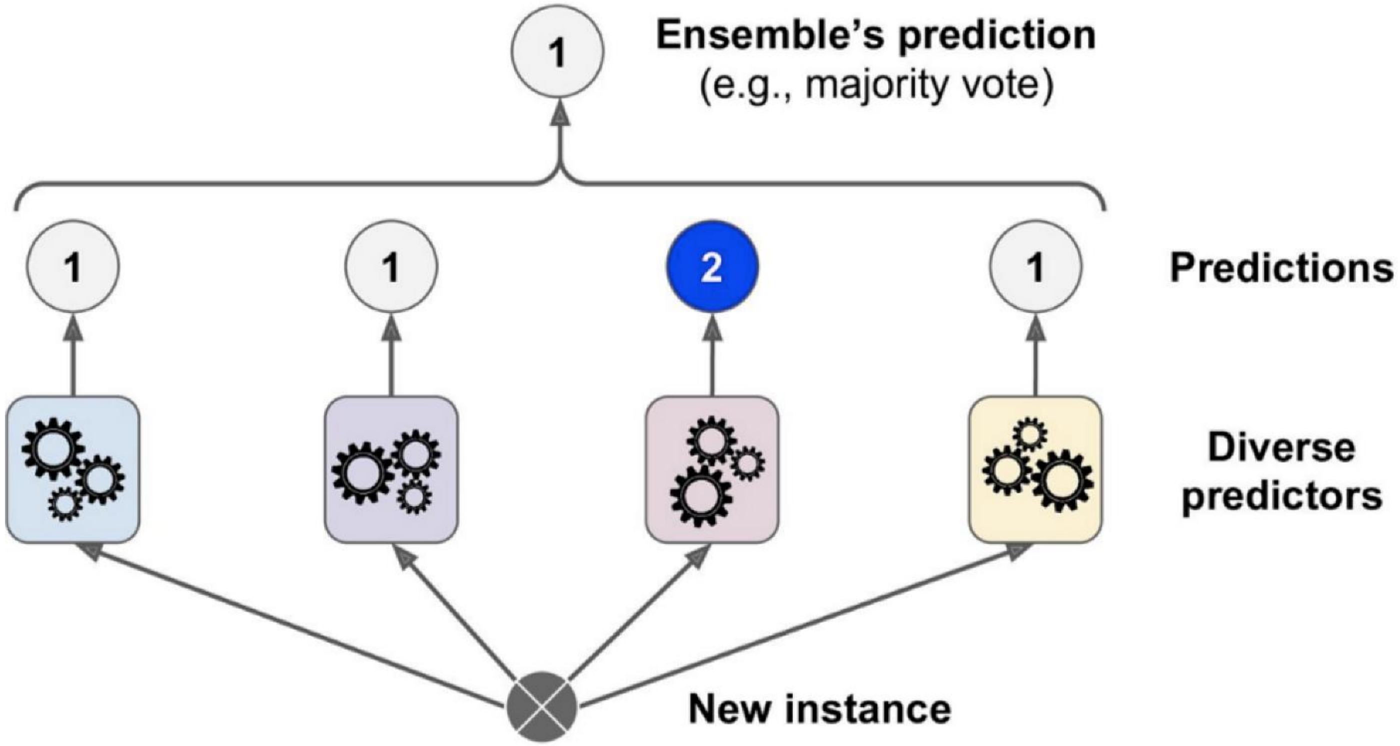
Decision trees: pros and cons

- Pros
  - **Easy** to explain and **human-interpretable**
  - Handles both **classification** and **regression**
  - **Fast** and makes **no asssumptions** about the shape of the data

- Cons
  - Splits have high **variance**, they tend to **overfit**!
  - **Poor performance**

Ensemble methods

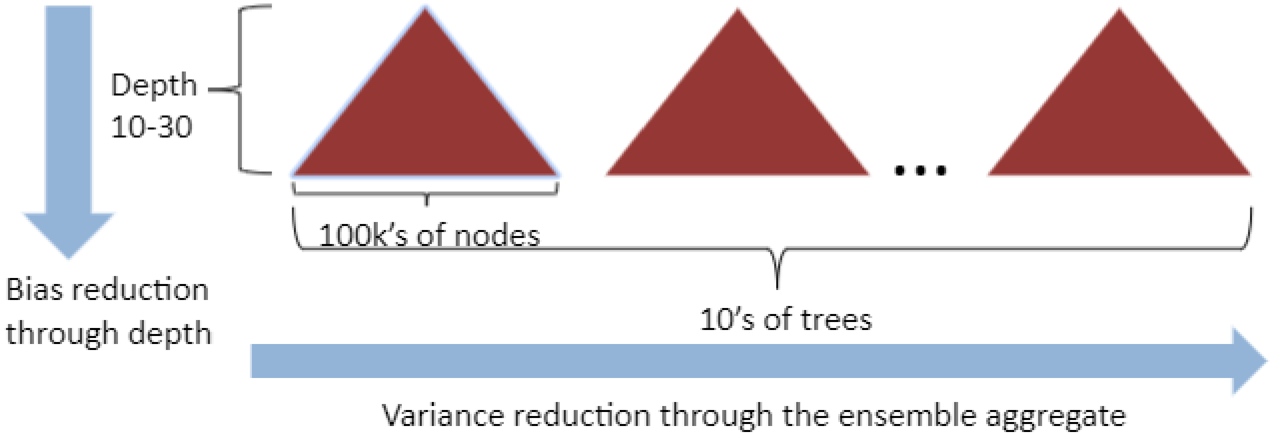
- **Combine** results of **weak learners** to make a **single better learner**
- Popular methods:
  - ➢ Random forests
  - ➢ Boosted trees



Source: "Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow", Chapter 7.

Random forests

- Grow K trees on datasets **sampled** from the original dataset (size N, p features) with replacement (Bootstrap simples)
  1. Drak K Bootstrap simples of size N
  2. Grow each decisión tree by selecting a **random subset m of features** at each node, and choosing the best feature to split on
  3. **Aggregate** predictions of the trees (average, most popular vote,…)



Credit: Robert West, EPFL