Martin Mundt, Dr. Iuliia Pliushch, Prof. Dr. Visvanathan Ramesh

Pattern Analysis & Machine Intelligence Praktikum: MLPR WS-19/20

Week 3: Random Forests

Recap: Python arrays

```python
a = np.ones(5)
print(a.shape)
print(a)
b = np.ones((5,1))
print(b.shape)
print(b)
```

```
(5,)
[1. 1. 1. 1. 1.]
(5, 1)
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

```python
M = np.ones((3,5))
ra = M@a
rb = M@b
print(ra.shape)
print(ra)
print(rb.shape)
print(rb)
```

```
(3,)
[5. 5. 5.]
(3, 1)
[[5.]
 [5.]
 [5.]]
```

```python
print(ra+b)
print(rb+np.ones((3,1)))
print(rb+b)
```

```
[[6. 6. 6.]
 [6. 6. 6.]
 [6. 6. 6.]
 [6. 6. 6.]
 [6. 6. 6.]]
[[6.]
 [6.]
 [6.]]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-38-f6d85db3c1b9> in <module>()
      1 print(ra+b)
      2 print(rb+np.ones((3,1)))
----> 3 print(rb+b)

ValueError: operands could not be broadcast together with shapes (3,1) (5,1)
```

Recap: Vectorization

- An alternatve to the **for-loop**: compute a function for all training examples at once

$$h_\theta(x^{(i)}) = \theta^T x^{(i)} = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \dots \\ x_n^{(i)} \end{bmatrix}$$

$$h_\theta(x) = X\theta = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & & x_n^{(2)} \\ \vdots & \vdots & \dots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

https://towardsdatascience.com/vectorization-implementation-in-machine-learning-ca652920c55d

Recap: Logistic regression

L

$$z_i = W x_i$$

$$a_i = \sigma(z_i)$$

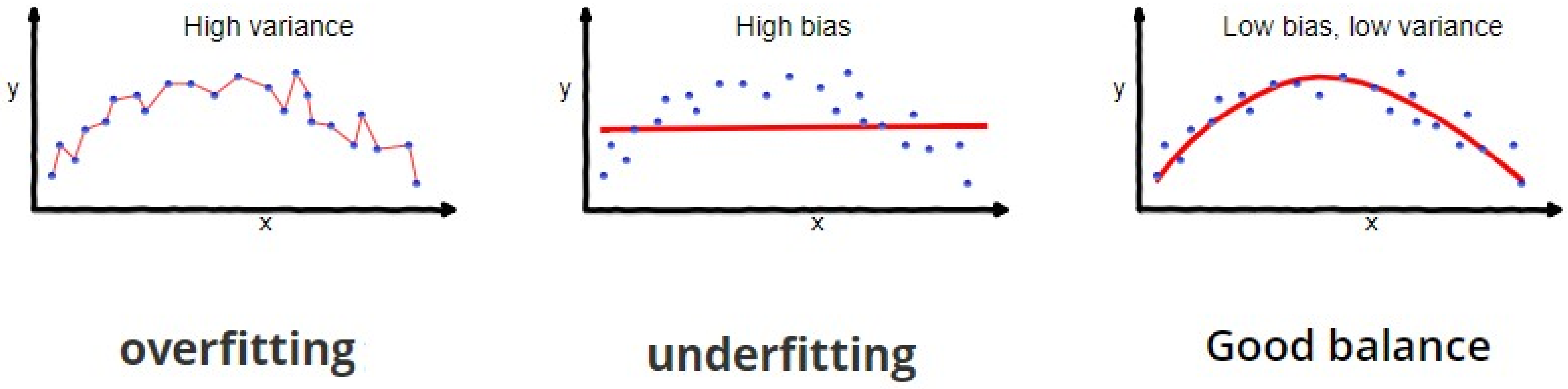
$$\frac{\delta L}{\delta W} = \sum_i \frac{\delta L}{\delta a_i}\frac{\delta a_i}{\delta z_i}\frac{\delta z_i}{\delta W} = \frac{1}{N}\sum_i -\frac{y_i - a_i}{a_i(1-a_i)} a_i(1-a_i) x_i =$$

$$\frac{1}{N}\sum_i -(y_i - a_i) x_i$$

$$\frac{\partial}{\partial w_i}\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log\left(P(y_n = 1|x_n, w) + (1-y_n)\log\left(1 - P(y_n = 1|x_n, w)\right)\right]\right. = 0$$

Models: bias-variance trade-off

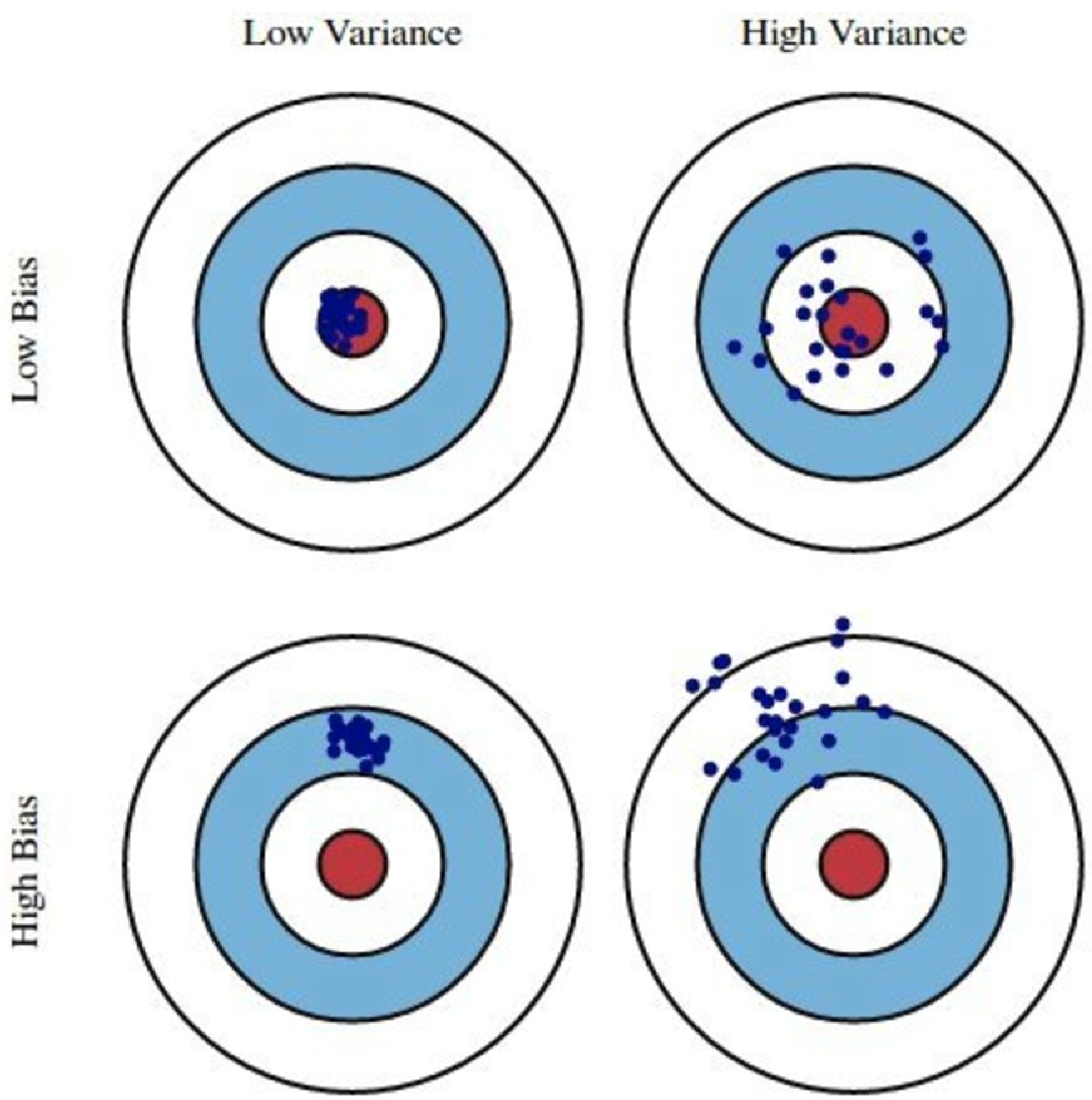
L



https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Models: bias-variance trade-off

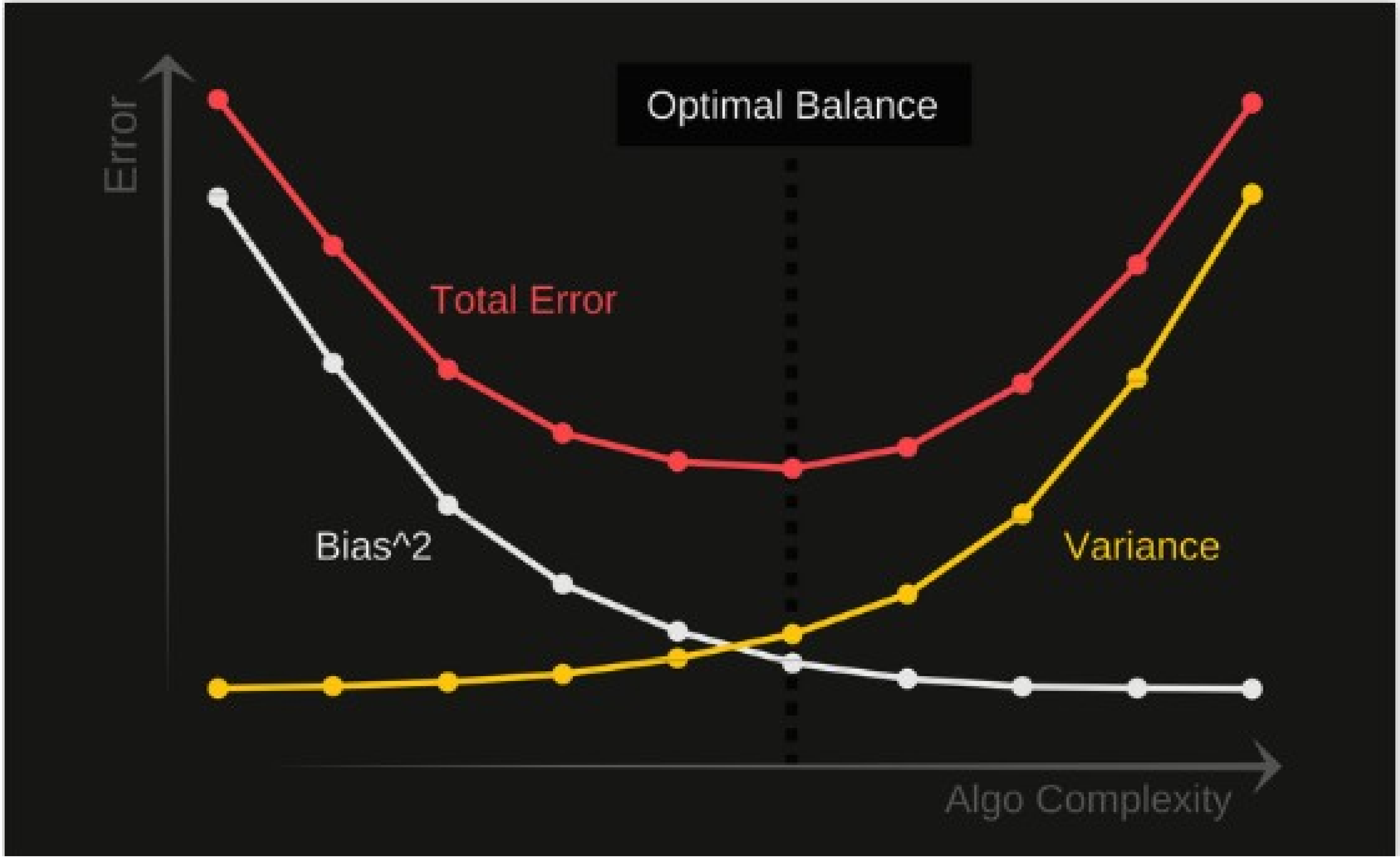
L



https://becominghuman.ai/machine-learning-bias-vs-variance-641f924e6c57

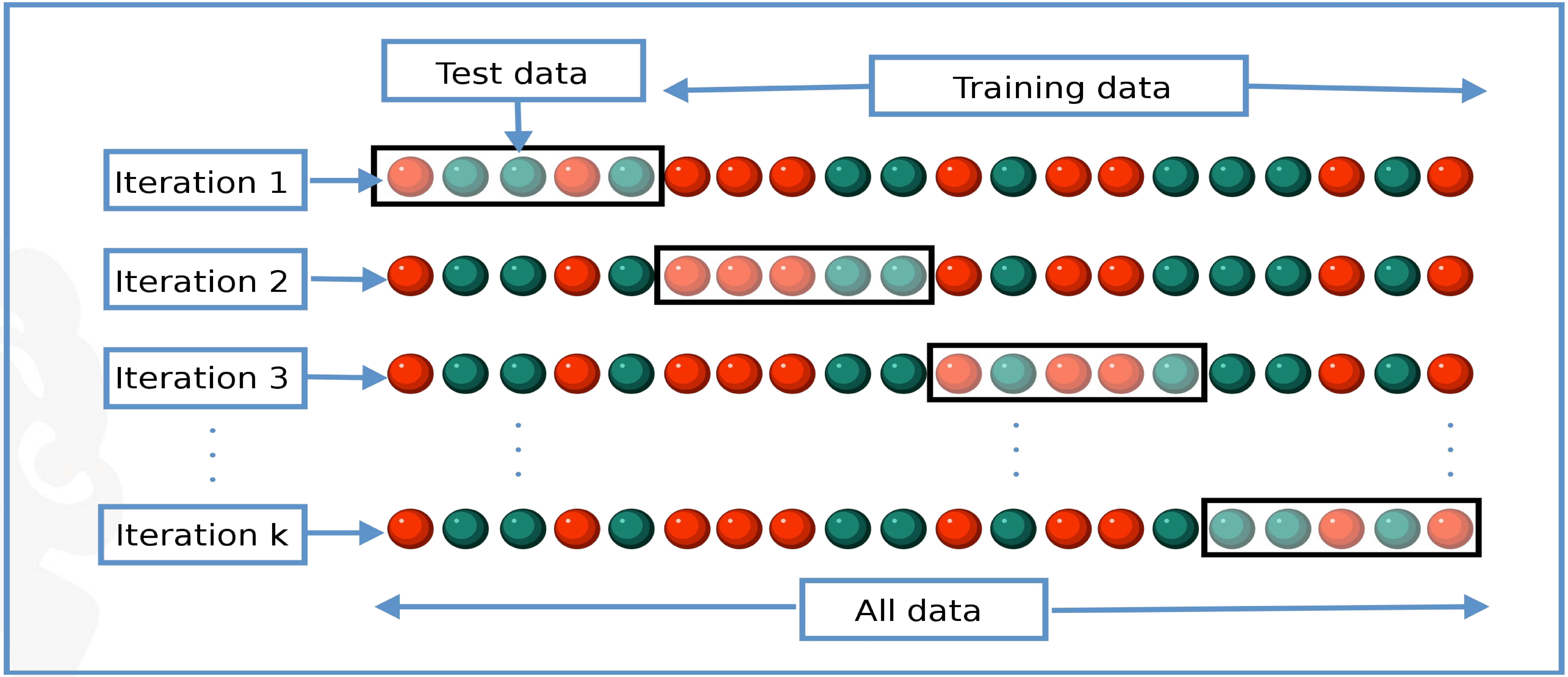
Models: bias-variance trade-off

L



https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

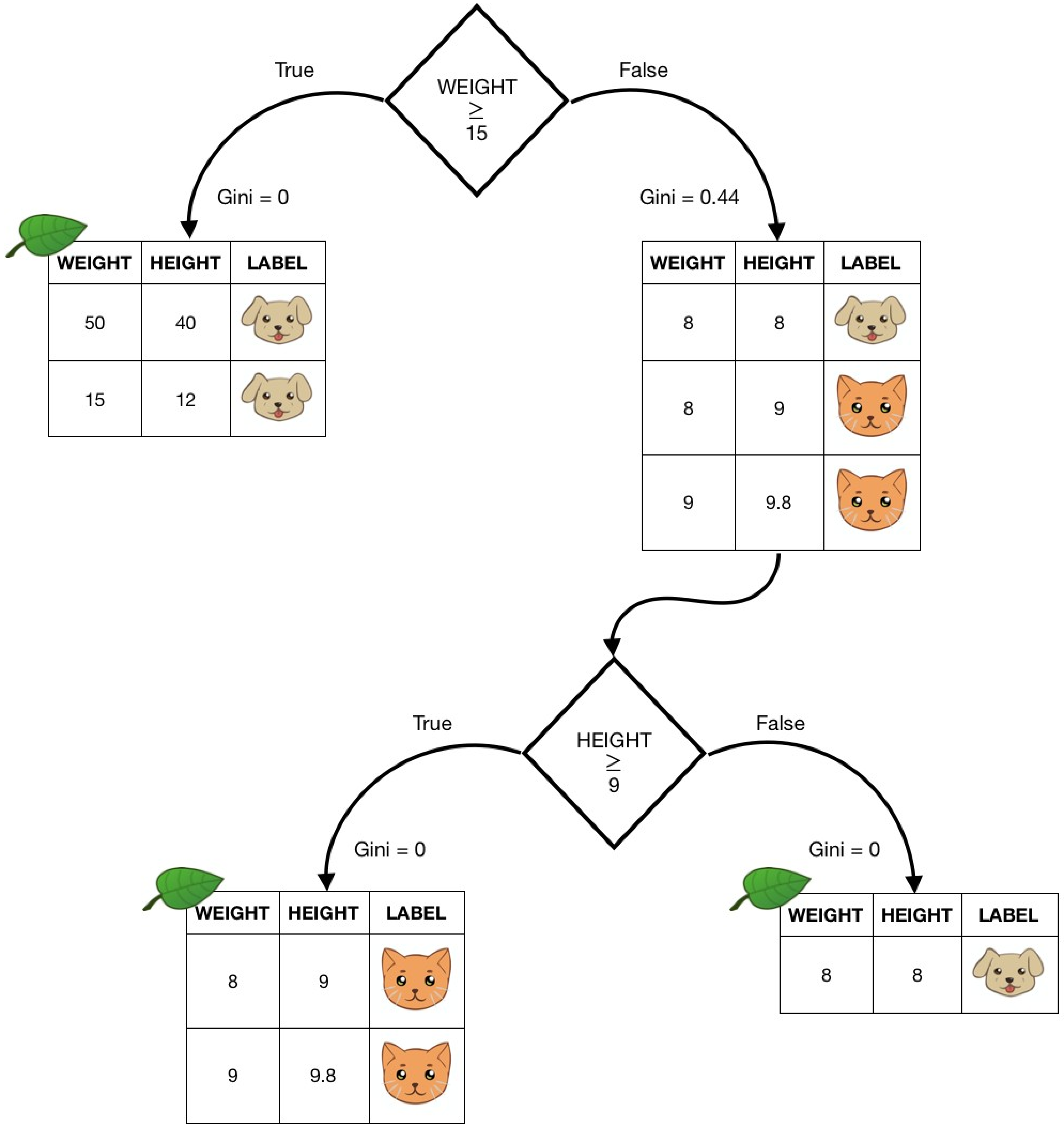
K-fold cross-validation



Decision trees and random forests

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- **Decision tree** is a machine learning algorithm for classification and regression
- **Random forests** is an **ensemble** learning algorithm which uses **multiple** decision trees for classification and regression

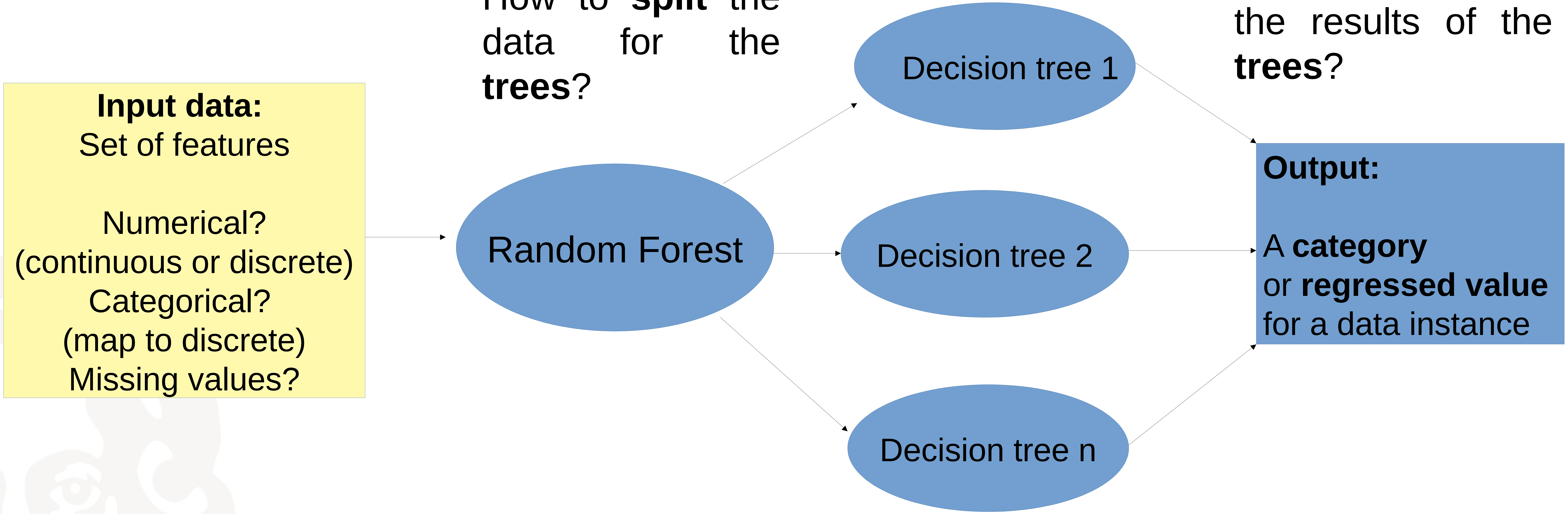


https://www.ke.tu-darmstadt.de/lehre/ws-18-19/mldm/dt.pdf

https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6

Decision trees and random forests

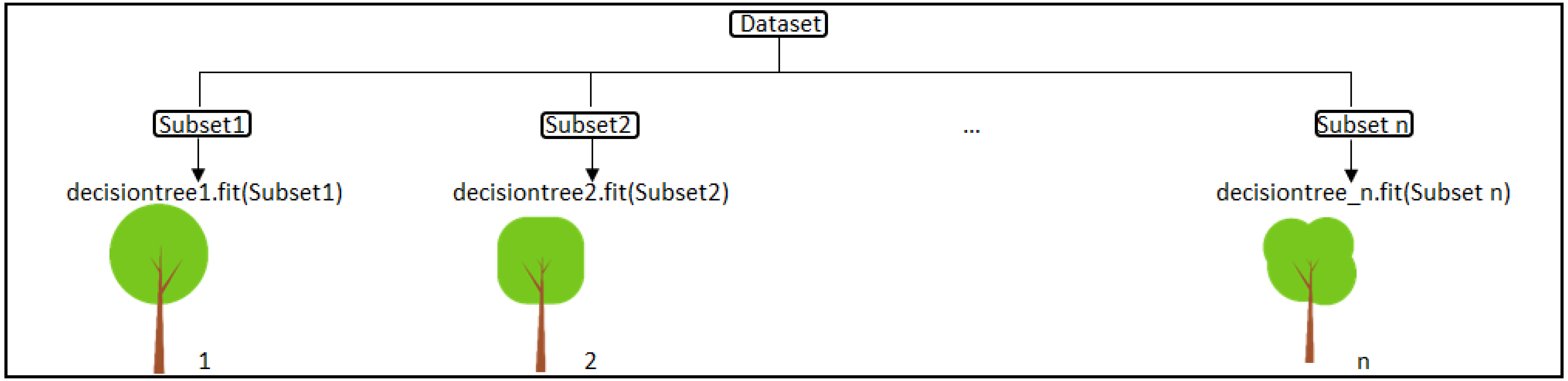
- How to **split** the data for the **trees**?
- How to **combine** the results of the **trees**?



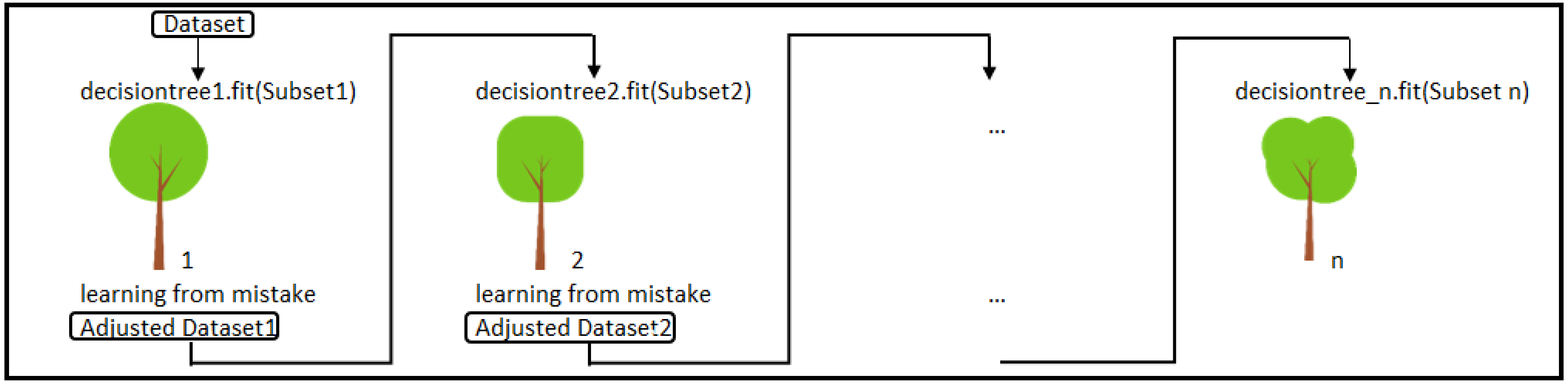
- **Pre- (while growing) and Postpruning** of trees as a means to avoid **overfitting**

Ensemble methods

Bagging

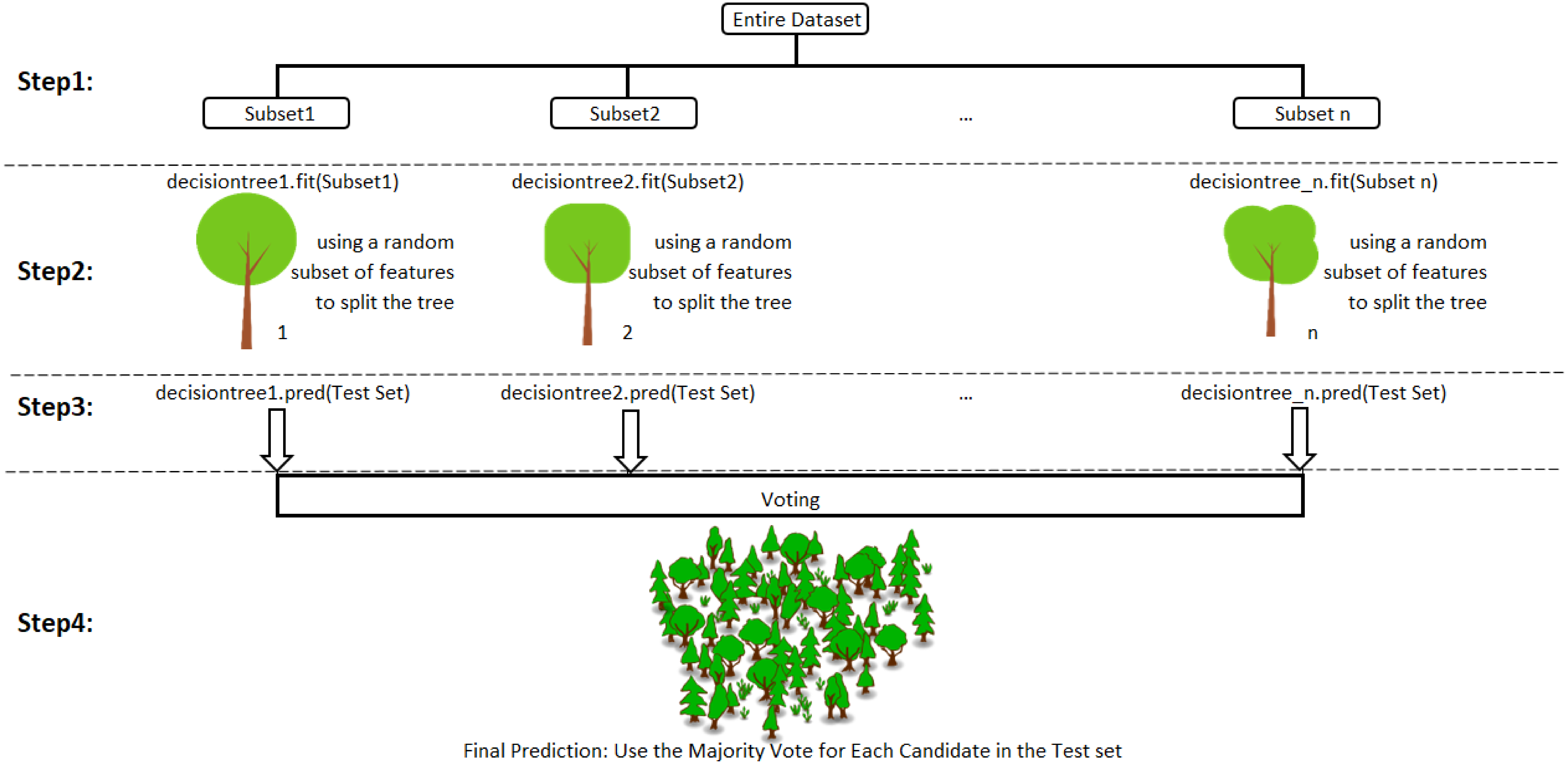


Boosting



https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725

Random forest (bagging)



https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725

Decision tree algorithms

- **ID3** (developed in 1986 by Ross Quinlan):
  - **categorical** features and targets
  - splitting criterion: **Information Gain**

- **C.5** (Quinlan) – commercial version of C4.5

- **C4.5** (Quinlan, 1993):
  - partitions the **continuous** features into a **discrete** set of intervals
  - suports missing values
  - splitting criterion: **Gain Ratio**

- **CART** (Classification and Regression trees):
  - similar to C4.5
  - supports **numerical target** variables (regression)
  - splitting criterion: **Gini-Index** for Classification**,** **Sum-of-Squares** for Regression

https://www.ke.tu-darmstadt.de/lehre/ws-18-19/mldm/dt.pdf https://scikit-learn.org/stable/modules/tree.html

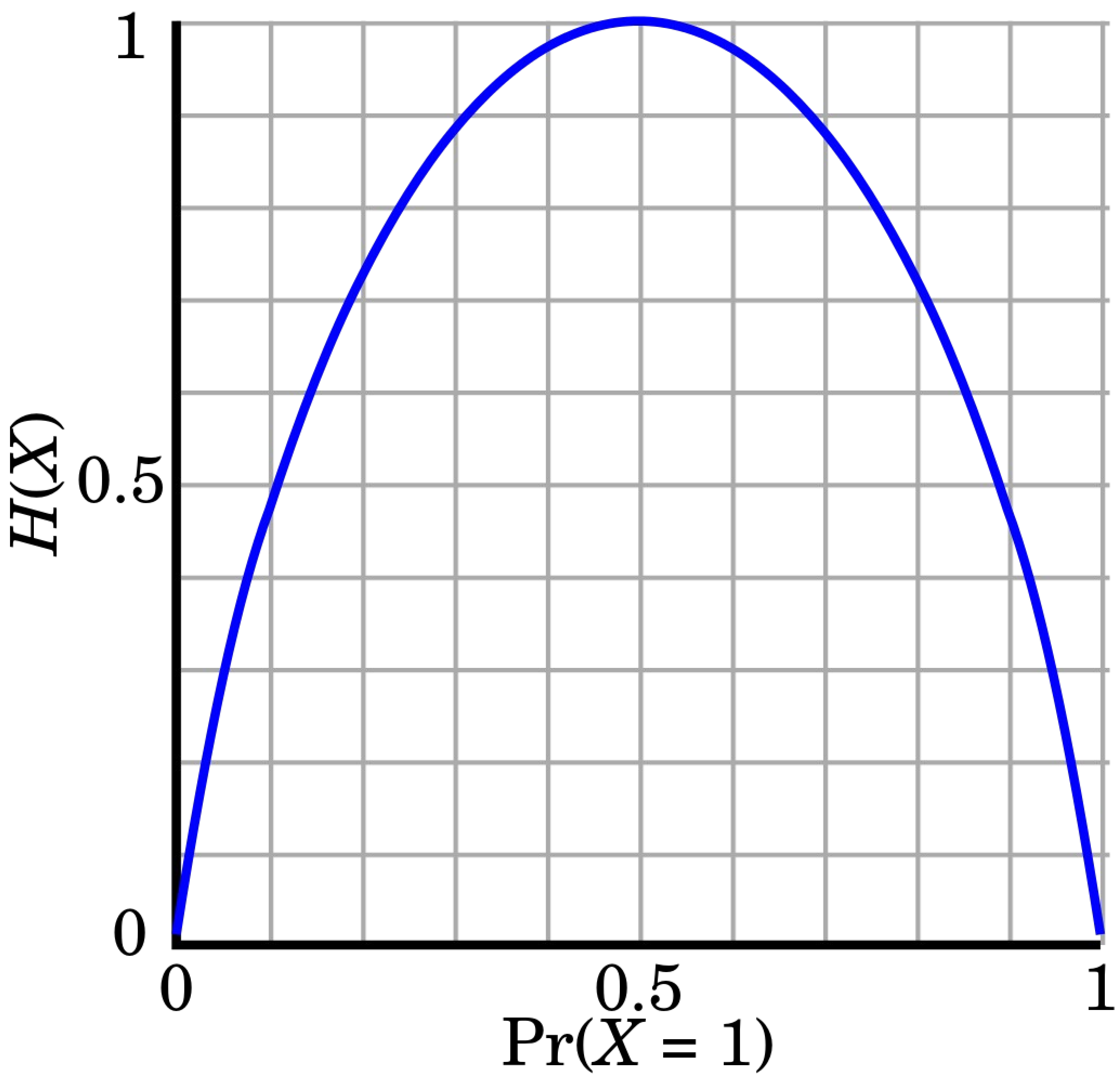
Splitting criteria: Entropy

- **Binary:**

$$-p_0 * \log_2(p_0) - p_1 * \log_2(p_1)$$

- **Multiclass:**

$$-\sum_{i \in Classes} p_i * \log_2(p_i)$$



By Brona and Alessio DamatoNewer version by Rubber Duck (☮ • ✍) - original work by Brona, published on Commons at Image:Binary entropy plot.png. Converted to SVG by Alessio Damato, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1984868

https://www.ke.tu-darmstadt.de/lehre/ws-18-19/mldm/dt.pdf

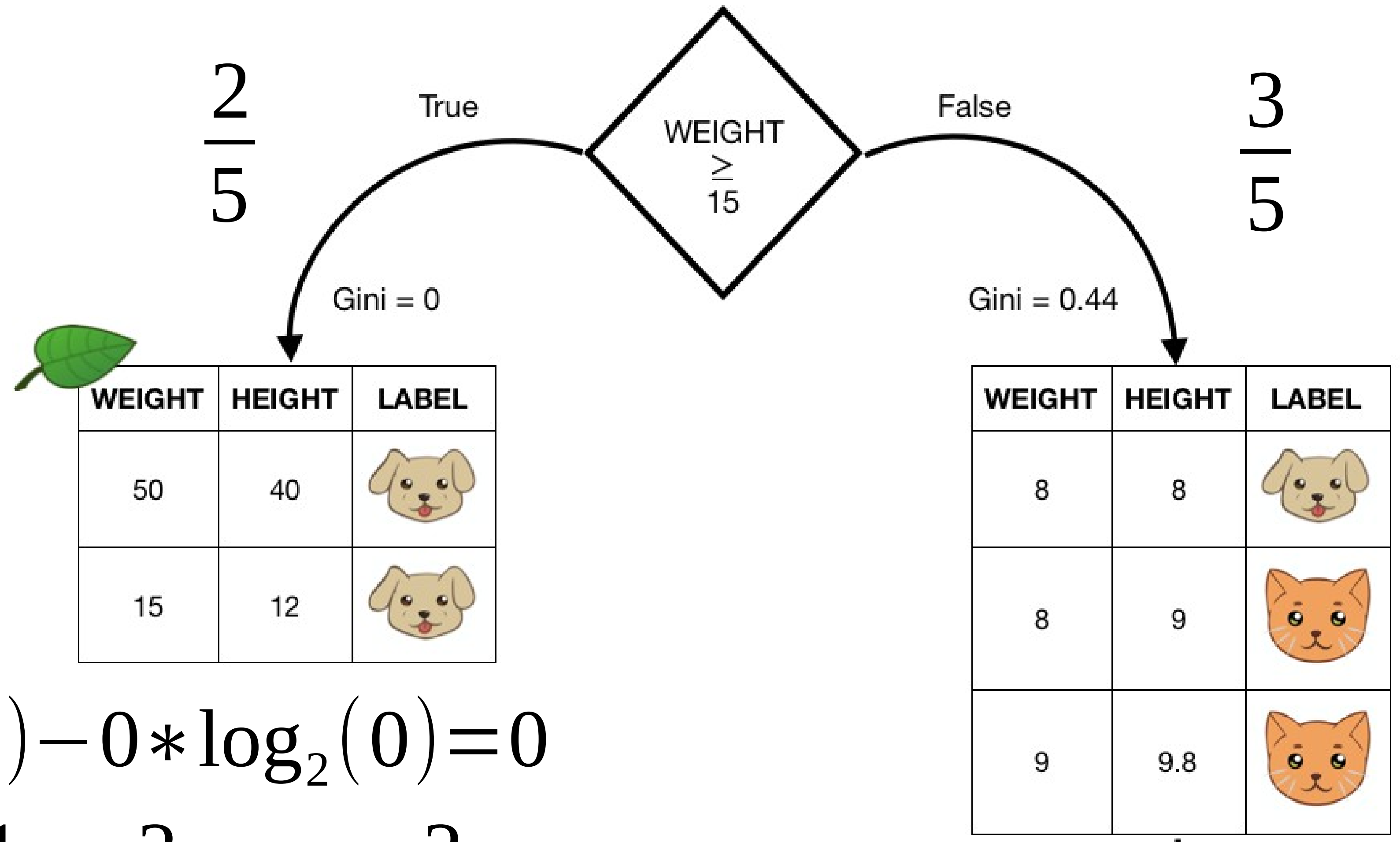
Splitting criteria

$$H(parent) = -\frac{2}{5} * \log_2\left(\frac{2}{5}\right) - \frac{3}{5} * \log_2\left(\frac{3}{5}\right) = 0.97$$

- **Entropy Gain:**

$$Gain(S,A) = H(S) - \sum_i \frac{|S_i|}{|S|} H(S_i)$$

$\frac{2}{5}$ $\frac{3}{5}$

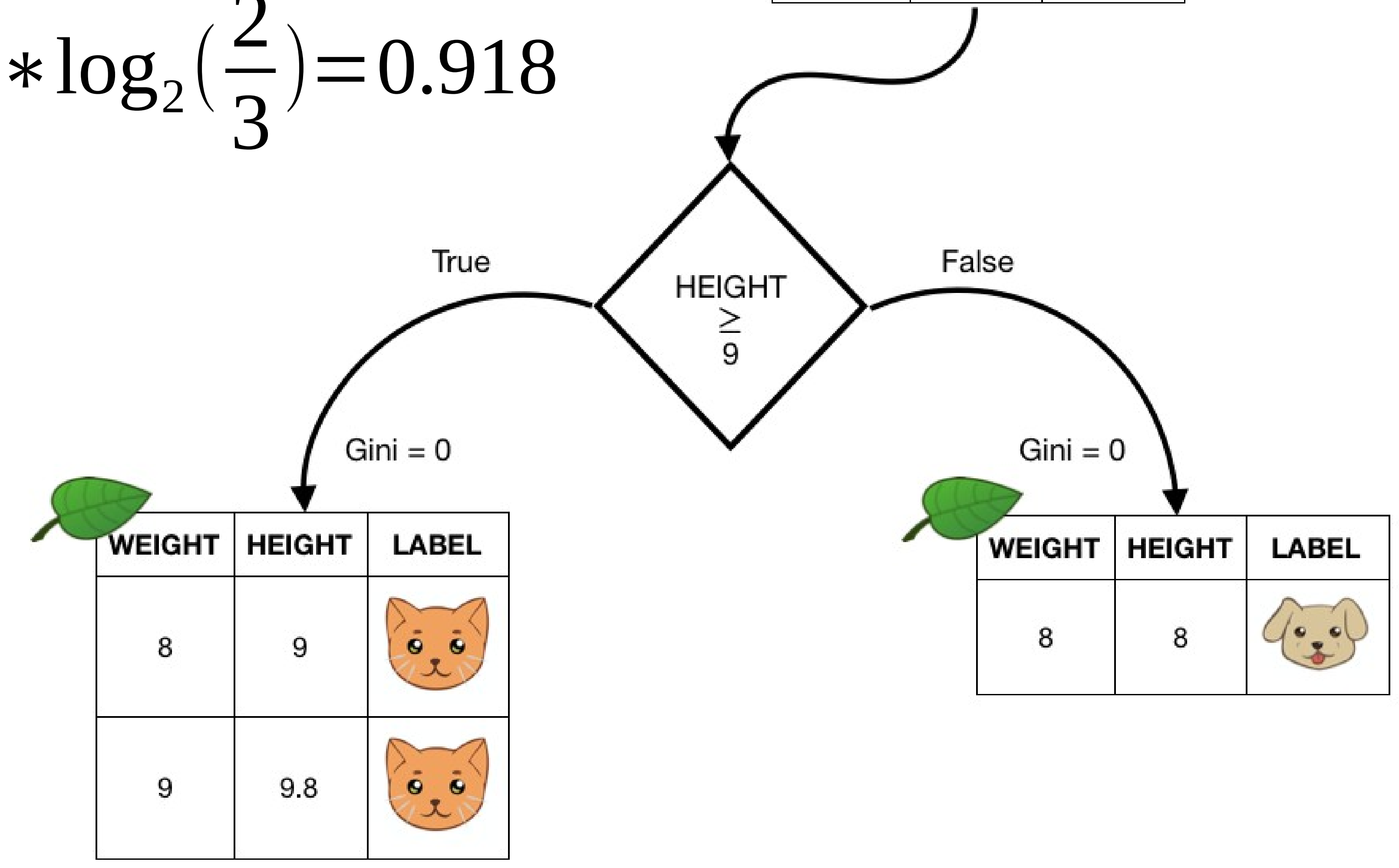


- **Intrinsic Information:**

$$H(leftchild) = -\frac{2}{2} * \log_2\left(\frac{2}{2}\right) - 0 * \log_2(0) = 0$$

$$H(rightchild) = -\frac{1}{3} * \log_2\left(\frac{1}{3}\right) - \frac{2}{3} * \log_2\left(\frac{2}{3}\right) = 0.918$$

$$IntI(S,A) = -\sum_i \frac{|S_i|}{|S|} \log_2\left(\frac{|S_i|}{|S|}\right)$$



- **Gain Ratio:** $\frac{Gain(S,A)}{IntI(S,A)}$

https://www.ke.tu-darmstadt.de/lehre/ws-18-19/mldm/dt.pdf

https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6

Splitting criteria (CART)

- **Gini (impurity measure)**
  – for classification

$$Gini(S)=1-\sum_{i\in Classes} p_i^2$$

$$Gini(S,A)=\sum_i \frac{|S_i|}{|S|} Gini(S_i)$$

- **MSE (Mean Squared Error)**
  – for regression

$$MSE(S)=\frac{1}{N}\sum_{i\in Ndata}(y_i-\mu_y)^2$$

$$Var(X)=E[(X-\mu)^2]=E[(X-E[X])^2]=E[X^2]-E[X]^2$$

https://www.ke.tu-darmstadt.de/lehre/ws-18-19/mldm/dt.pdf

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

https://web.stanford.edu/class/stats202/content/lec19.pdf

Decision tree

- Input: Set of features, class to predict

- 1. Create a (root) node
- 2. If termination criteria are met, make it a **leaf**
- 2. Select the best **feature** to split the data according to **criterion (loop over selected features)**
- 3. Split the **data** accordingly
- 4. Create subtrees for each **data subset (RECURSION!)**

Titanic dataset