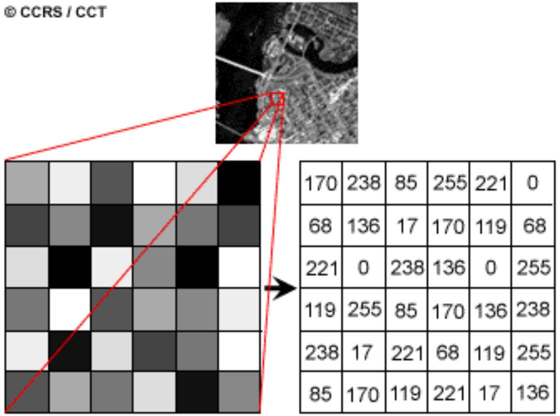
Machine Learning: writing the code

Maura Pintor

Pattern Recognition as a Classification Problem

- Pattern classification is about assigning class labels to patterns

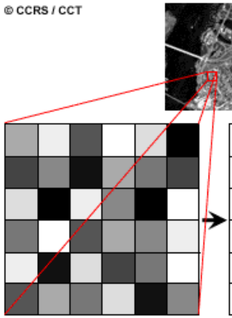
- Patterns are described by a set of measurements called *features* (or attributes)
  - For images, feature/input values could correspond to the brightness of each pixel



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Basic Concepts: Class and Features

- Each input sample is described by a feature vector with "d" elements: $\mathbf{x} = (x_1, x_2, \ldots, x_d)$.



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

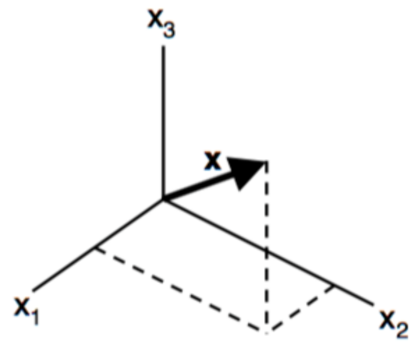
$$\mathbf{x} = (x_1, x_2, \ldots, x_d) = (170, 238, 85 \ldots 136)$$

- **Class**: intuitively, a class contains similar patterns, whereas patterns from different classes are dissimilar (e.g., dogs and cars)
  - In this course, we assume that there are *c* possible classes, normally denoted with $y_1 \ldots y_c$
  - Each sample belongs to one of the "c" classes - We say that each input sample has a **class label**

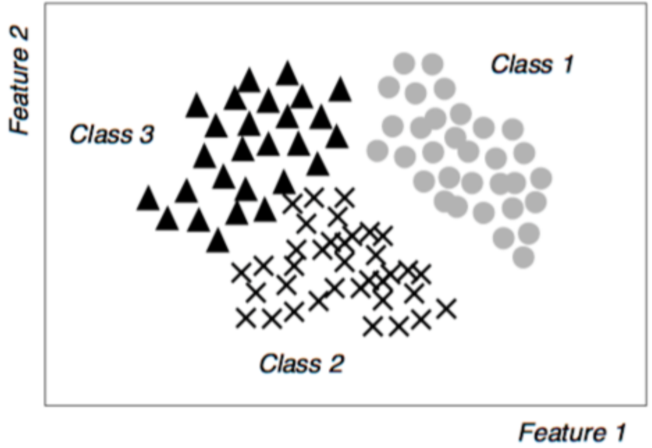
Basic Concepts: Feature Vector and Feature Space

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Feature vector



Feature space (3D)

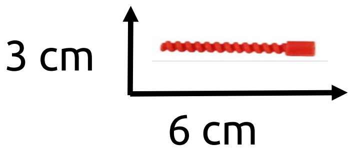


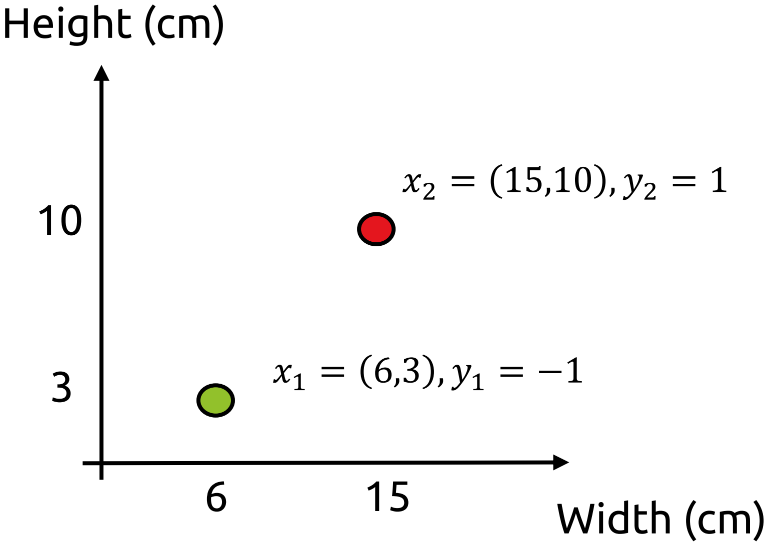
Scatter plot (2D)

Open in Colab https://colab.research.google.com/github/maurapintor/ai4dev/blob/main/AI4Dev_01_intro_numpy_lab.ipynb

Feature Extraction

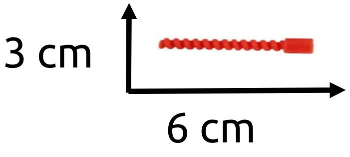
Now we can represent the two samples in the feature space

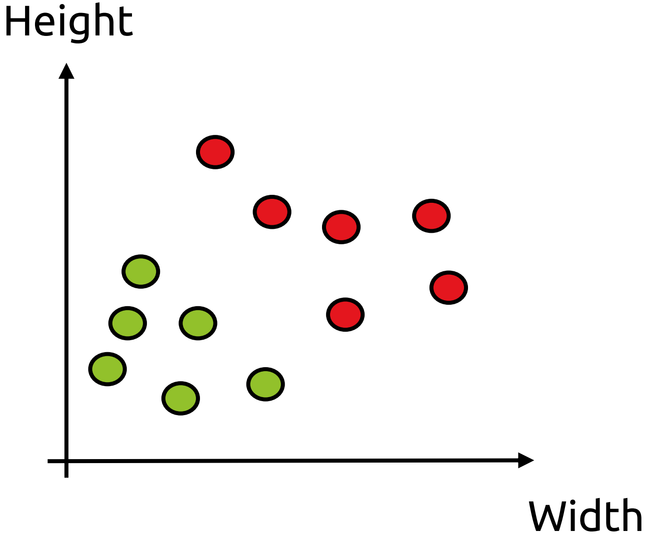




Feature Extraction

… and repeat for different screws and hammers…





Basic Concept: Design or Training Dataset

- The information to design a machine-learning model is usually in the form of a labeled data set $\boldsymbol{D}$ (called design or training set):

$$\boldsymbol{D} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$$

$$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \text{ i=1,..,n}$$

$\boldsymbol{x}_i$ belongs to one of the "c" classes ($\boldsymbol{x}_i$ belongs to y_j $j=1,\ldots,c$)

- In the previous example, $\boldsymbol{D}$ is the data set of screws and hammers that we collected…

Feature Scaling

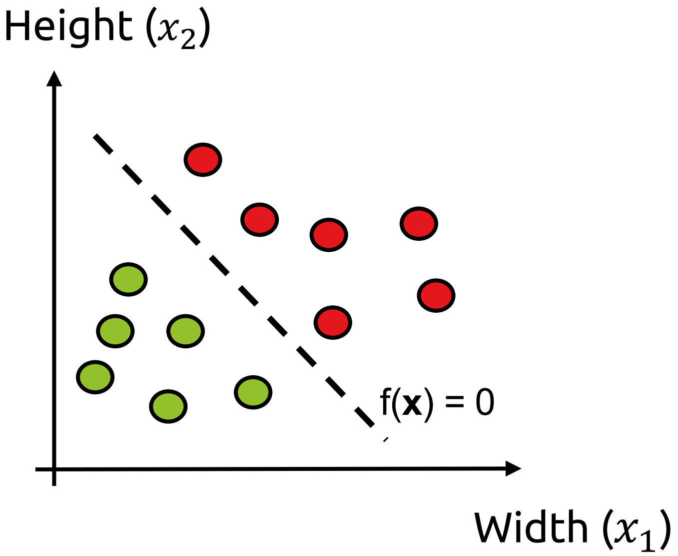
- Feature values are normally scaled within a bounded interval to facilitate model training

- **Min-max scaling:** $x' = \frac{x-m}{M-m}$, where x is the input feature, and
  - m and M are the min and max values of that feature over the whole training set

- **Z-score scaling:** $x' = \frac{x-\mu}{\sigma}$, where x is the input feature, and
  - μ and σ are the mean and standard deviation of that feature estimated from the training set

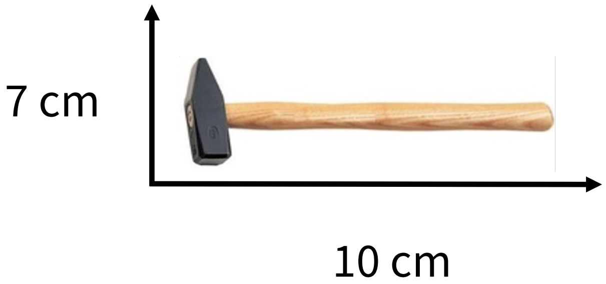
Model Training

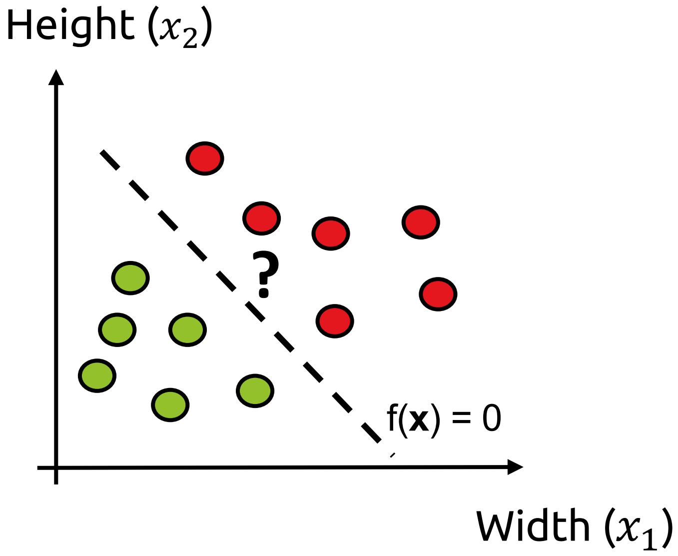
- Can you find one? Yes, of course…
- Let's pick $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = w_1x_1 + w_2x_2 + b$
- If $f(\boldsymbol{x}) \geq 0$ the model predicts «hammer» and «screw» otherwise
- Model training aims to estimate the model parameters $(\boldsymbol{w}, b)$ from the training data,
  - using either a probabilistic approach or solving an optimization problem
- This model makes zero errors on the training data. *Can we trust it?*



Model Evaluation

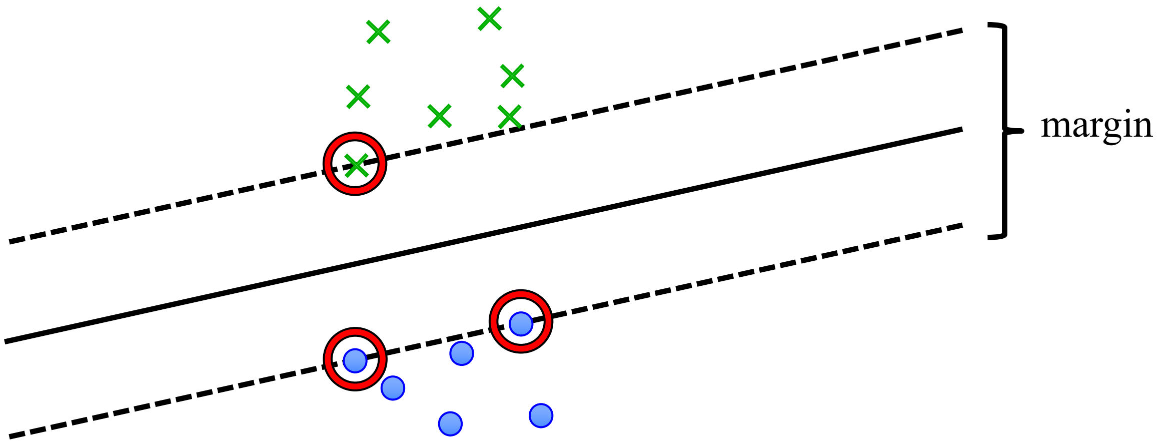
- Once the model is trained, it should be evaluated on never-before-seen input samples (a.k.a. *test* samples) to check if it can *generalize*…





Margin Maximization and Support Vectors

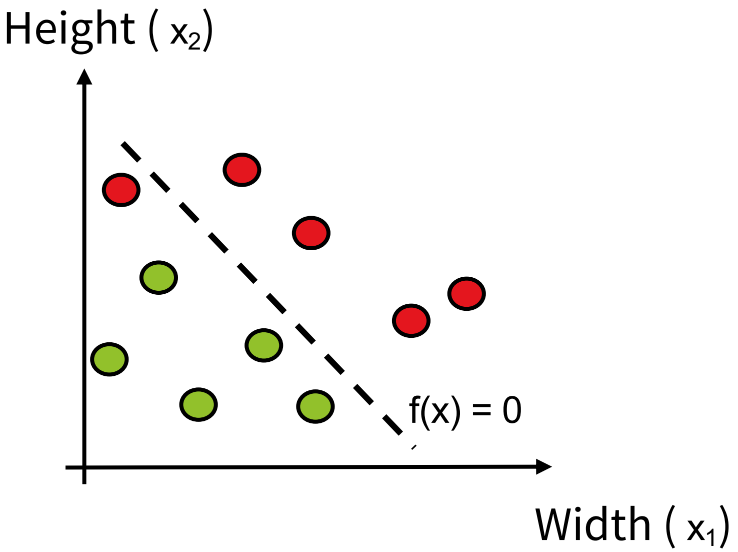
- The circled points are called ***support vectors***



- The decision hyperplane only depends on the SVs
  - the other points can move freely without violating the margin

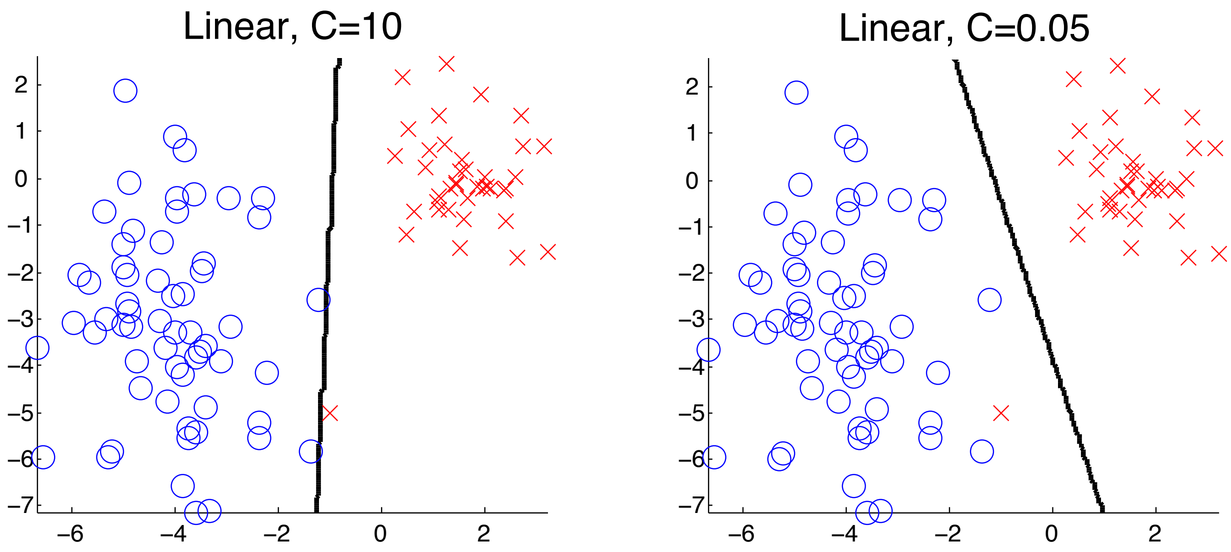
Model Evaluation

- We evaluate our model on the test set shown aside, and report only 1 mistake out of 10.
- We say that the classification accuracy of this model is 90% (or its test error is 10%)



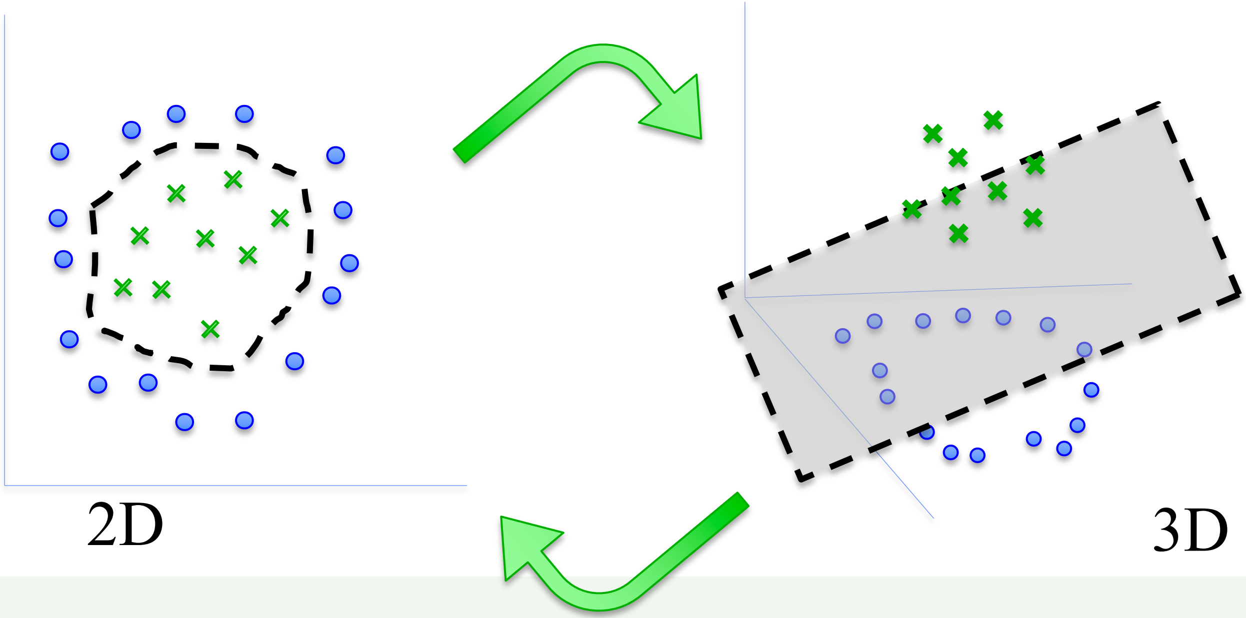
Open in Colab https://colab.research.google.com/github/maurapintor/ai4dev/blob/main/AI4Dev_02_data_visualization_learning_lab.ipynb

Effect of the Hyperparameter C



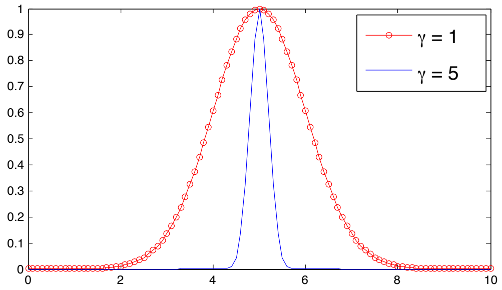
What If Data Is More Complex? Use Kernels!

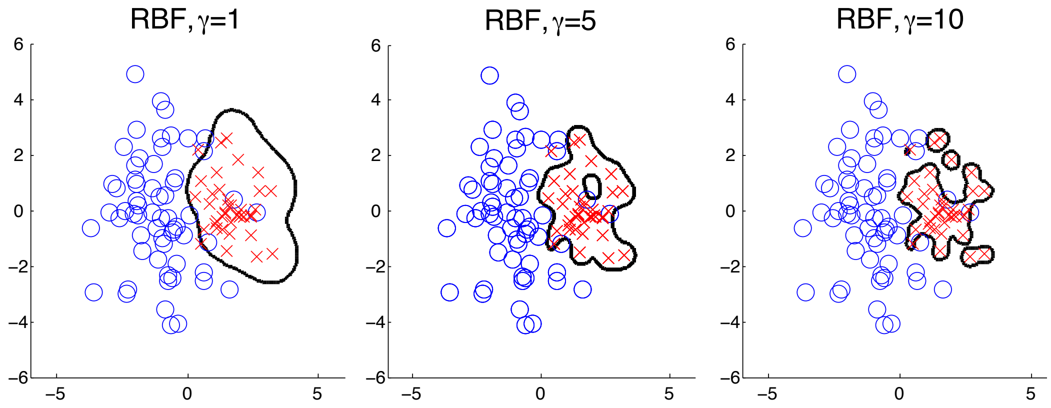
- Project it onto a higher-dimensional space, and optimize a linear model
  - This amounts to learning a non-linear model in the input space



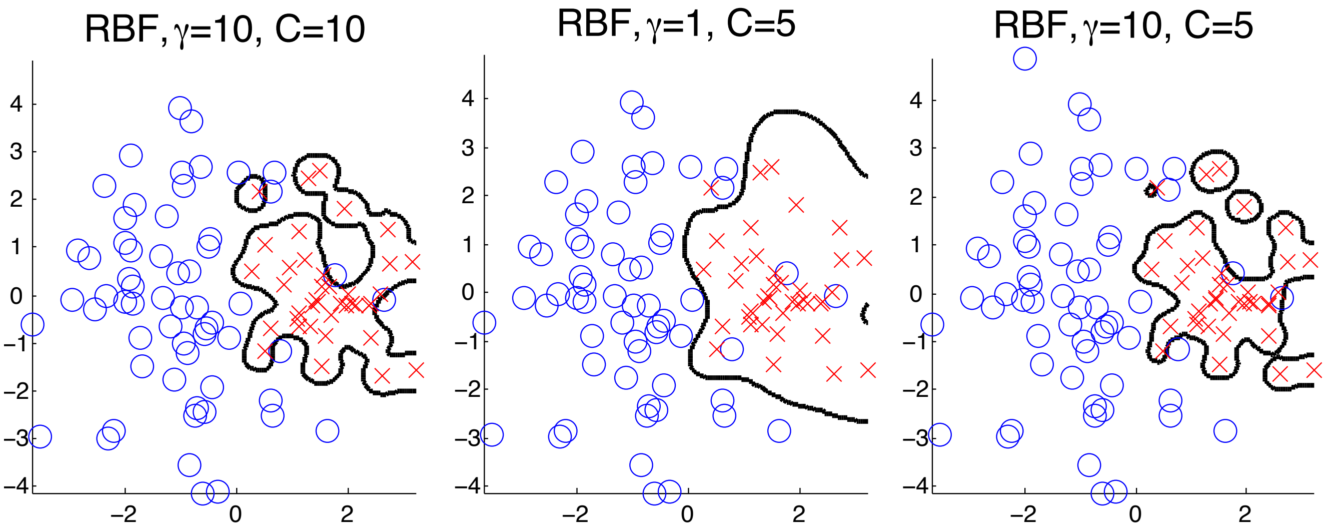
The Gaussian Kernel (Radial Basis Function, RBF)

$$K(\mathbf{x_i}, \mathbf{x'}) = e^{-\gamma(\mathbf{x}_i - \mathbf{x}')^2}$$



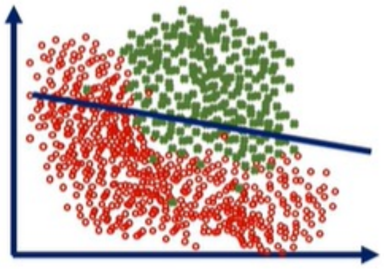


Playing with the Hyperparameters

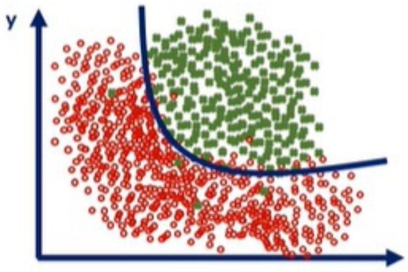


Model Training: Underfitting vs. Overfitting

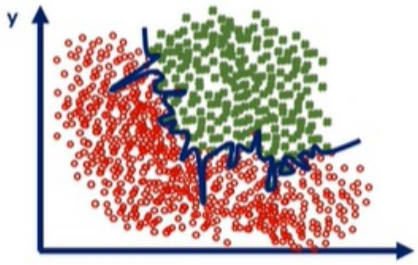
- Why do we need testing on a separate data split?



Underfitting
- high training error
- high test error

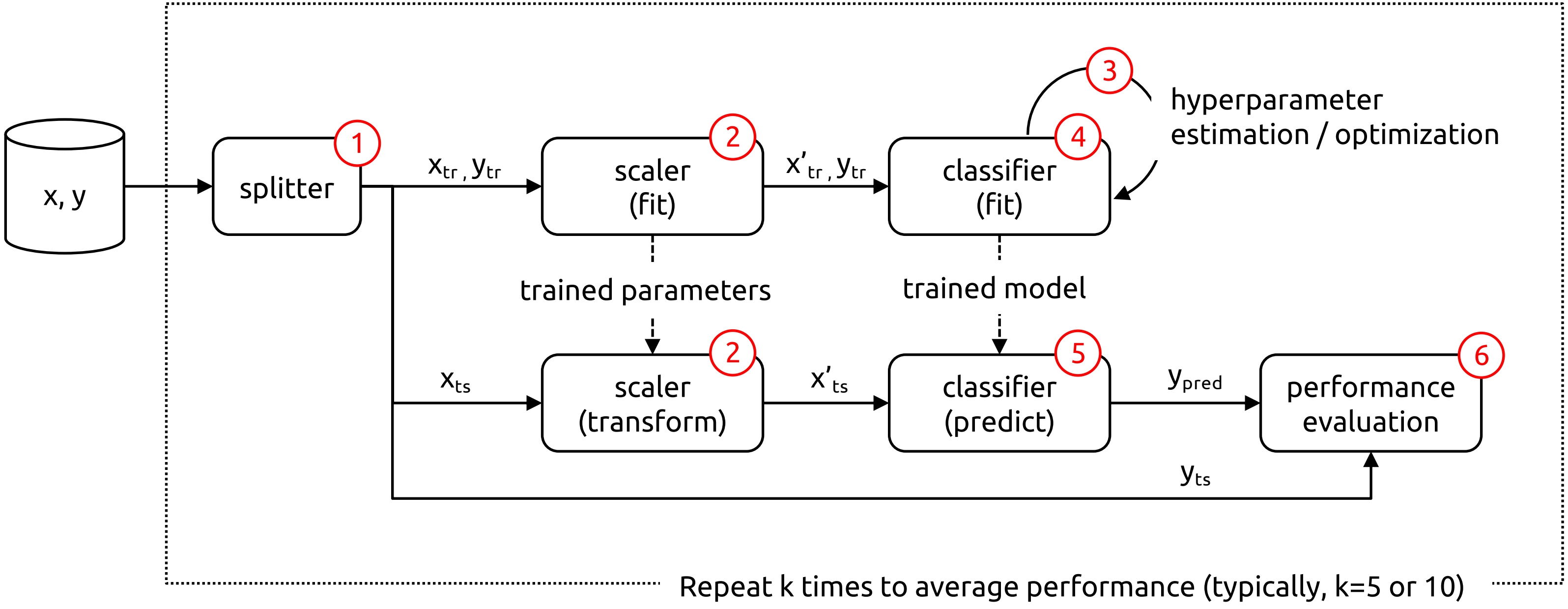


Good fit
- low training error
- low test error



Overfitting
- low training error
- high test error

ML Model Design



Open in Colab https://colab.research.google.com/github/maurapintor/ai4dev/blob/main/AI4Dev_03_ml_pipeline_lab.ipynb