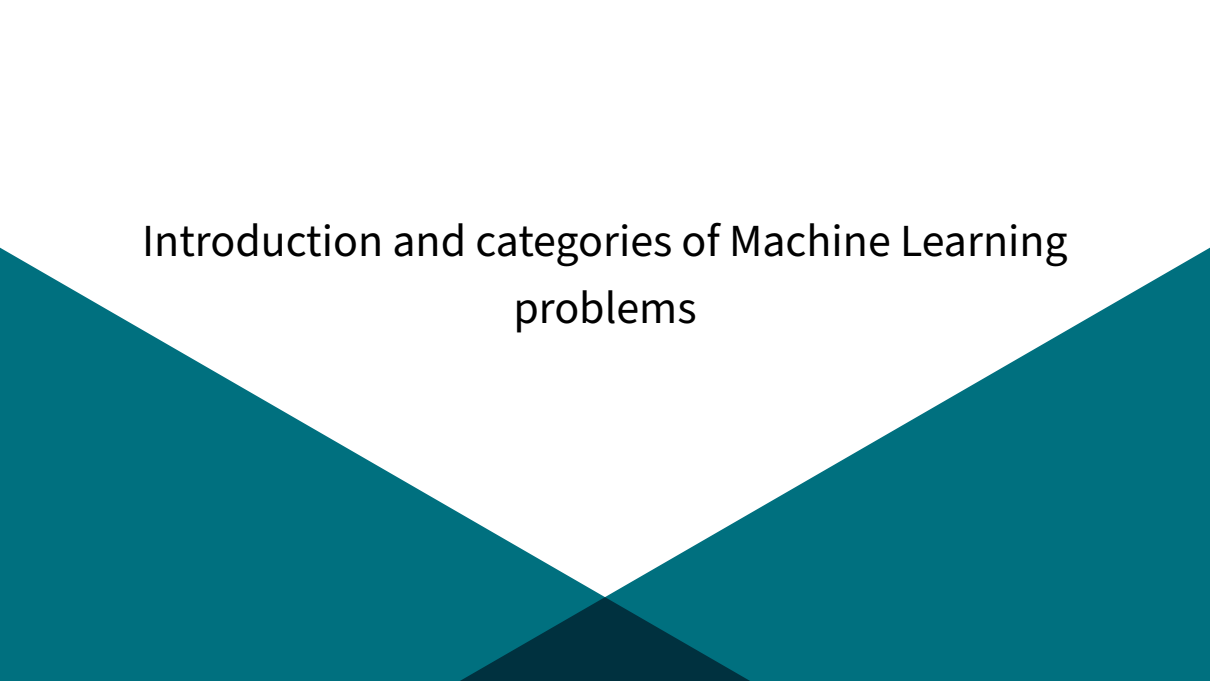## Overview

1. Introduction and categories of Machine Learning problems

2. Supervised Learning

3. Supervised Learning Methods

4. Steps to apply machine learning

5. Application

For more info

Short intro in my PhD thesis: `https://orbi.uliege.be/handle/2268/82167` An introductory course: `https://people.montefiore.uliege.be/lwh/AIA/`

Introduction and categories of Machine Learning problems

Machine Learning

Machine learning is the field of computer algorithms that improve automatically through experience and data. Common applications:

- ▶ Image recognition
- ▶ Email Spam & Malware filtering
- ▶ Speech recognition
- ▶ Social media customized advertising
- ▶ Netflix's recommendations
- ▶ Medical diagnosis

Machine Learning, why now?

- ▶ More Data
- ▶ Faster compute engines
- ▶ Algorithms (old and new)
- ▶ Machine learning software

Categories of Machine Learning problems

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Reinforcement Learning
- ▶ And many variants ...

Supervised Learning I

► Starting point: a dataset where inputs and outputs are clearly identified, e.g.:

| | Inputs | | | Output |
| | | | | |
X_1	X_2	X_3	X_4	Y
-0.61	-0.43	Y	0.51	Healthy
-2.3	-1.2	N	-0.21	Disease
0.33	-0.16	N	0.3	Healthy
0.23	-0.87	Y	0.09	Disease
-0.69	0.65	N	0.58	Healthy
0.61	0.92	Y	0.02	Disease

Supervised Learning II

▶ Goal: find a function f_Θ of the inputs that approximates at best the outputs, e.g.

$$\hat{y} = f(x_1, x_2, x_3, x_4)$$

▶ Training set: part of the available data used for identifying the best possible function $f_{\Theta^\star}$ (its best parameters), Test set: part of the dataset to evaluate f.

X_1	X_2	X_3	X_4	Y
-0.61	-0.43	Y	0.51	Healthy
-2.3	-1.2	N	-0.21	Disease
0.33	-0.16	N	0.3	Healthy
0.23	-0.87	Y	0.09	Disease
-0.69	0.65	N	0.58	Healthy
0.61	0.92	Y	0.02	Disease

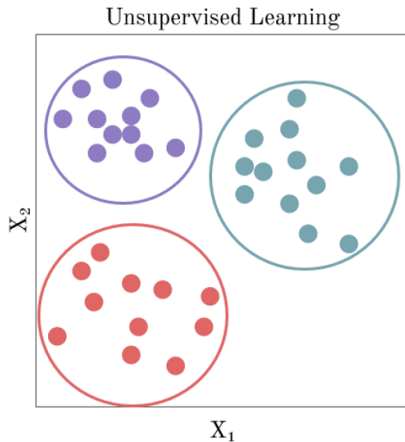# Supervised Learning III

▶ Inference: When *f* is determined, do a prediction using *f* for inputs when the output is not available

X_1	X_2	X_3	X_4	Y
0.75	0.49	Y	-0.88	$f(0.75, 0.49, Y, -0.88)$

Unsupervised Learning I

► Unsupervised learning does not use labeled data.

► Instead of making predictions, the goal of unsupervised learning is to find the underlying structure of dataset and group that data according to similarities.

► Example: Clustering. Assuming we have a dataset of points in a 2D space, group them in 3 clusters:
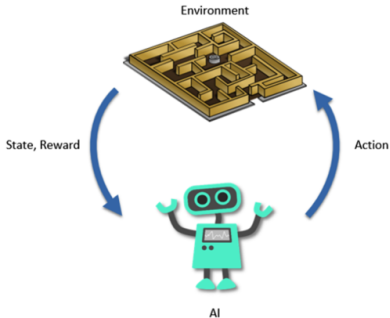

Unsupervised Learning

Unsupervised Learning II

Application: if you have 365 days of conmumption data with 15 minute resolution, identify e.g. a subset of 12 days that represent well the 365 days.

Reinforcement Learning

▶ Learning from interactions with an environment.
▶ Goal: By interacting with and environment, learn to take actions in any given state to maximize a reward function.
▶ Example:
  ▶ State: Robot position in a labyrinth.
  ▶ Actions: Go ahead, turn right, turn left.
  ▶ Reward: Get a reward each time the robot gets out.
  ▶ Goal: Learn how to get out as fast as possible.

Environment

State, Reward

Action

AI

Supervised Learning

Supervised Learning (SL)

► Data organized as a set of N objects described by input features and output labels.

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

► Features x_i are easily obtained by observation.

► Output y_i is provided by an expert or is the result of difficult/costly observation/computation.

Mapping, Loss Function, Expected Loss

▶ SL algorithms search for a mapping $f : \mathcal{X} \to \mathcal{Y}$ between feature and output spaces that generalizes well to elements for which the output value has not been observed.

▶ Define a loss function

$$L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$$

▶ Search for f which minimizes the expected loss:

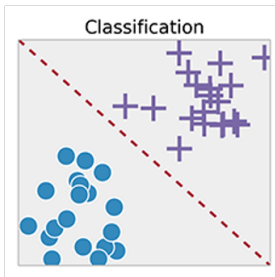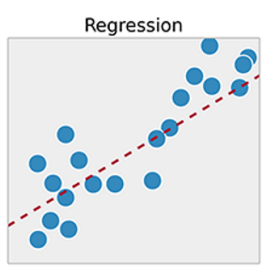$$\mathbb{E}_{\mathbb{P}_{X,Y}}[L(y, f(x))]$$

Classification and Regression

Classification:

► Approximate a mapping f from inputs X to **discrete** outputs y.

► Examples: Spam detection, room occupancy.

Regression:

► Approximate a mapping f from inputs X to **continuous** outputs y.

► Examples: House price prediction, time series forecasting.

Classification

Regression

Empirical Loss Minimization

- ▶ We usually don't know the joint distribution.
- ▶ Instead, minimize an estimate of the expected loss, the **empirical loss**:

$$\frac{1}{N} \sum_{i=1}^{N} L(y_i, h(x_i))$$

Model Selection

- By selecting a supervised learning method[1] (kNN, SVM, etc.), we restrict and parameterize the hypothesis space of functions $f_\Theta(x)$

- But how do we pick up a method / model, and how do we optimize its parameters? *The next example will give us some insights.*
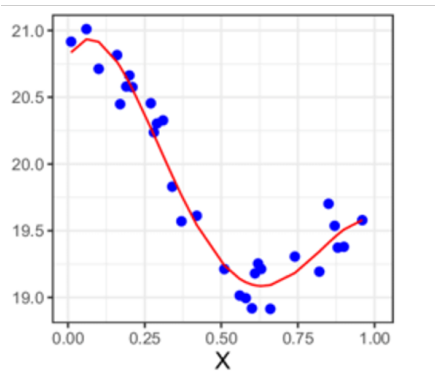
---

[1] See page 24.

Linear Model

- Polynomial of degree 1 regression model with 1 input and 1 output.
- Equation: $h(X) = \omega_0 + \omega_1 X$
- Learning phase: Find values for ω_0 and ω_1 to best fit the training set.
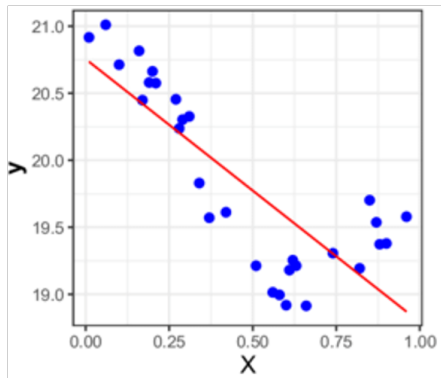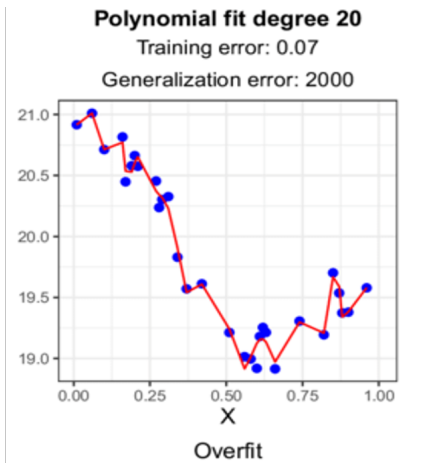
Polynomial of degree 4

▶ Polynomial of degree 4 regression model with 1 input and 1 output

▶ Equation:
$$h(X) = \omega_0 + \omega_1 X + \omega_2 X^2 + \omega_3 X^3 + \omega_4 X^4$$

▶ Learning phase: Find values for $\omega_0, \omega_1, \omega_2, \omega_3, \omega_4$ to best fit the training set.

Polynomial of degree 20

- Polynomial of degree 20 regression model with 1 input and 1 output.
- Equation: $h(X) = \sum_{i=0}^{20} \omega_i X^i$
- Learning phase: Find values for $\omega_0, \ldots, \omega_{20}$ to best fit the training set.

Polynomial fit degree 20
Training error: 0.07
Generalization error: 2000

Overfit

Model Selection

Error[a] on the **training set**

- ▶ Linear model: 0.4
- ▶ Degree 4 polynomial: 0.14
- ▶ Degree 20 polynomial model: 0.07

You need to evaluate your model on unseen
data → **Test set**

---

[a]E.g. mean absolute error, see page 39.

Model Selection

Error[a] on the **training set**

- ▶ Linear model: 0.4
- ▶ Degree 4 polynomial: 0.14
- ▶ Degree 20 polynomial model: 0.07

Error on the test set:

- ▶ Linear model: 0.42
- ▶ Degree 4 polynomial: 0.17
- ▶ Degree 20 polynomial model: 2

You need to evaluate your model on unseen data → **Test set**
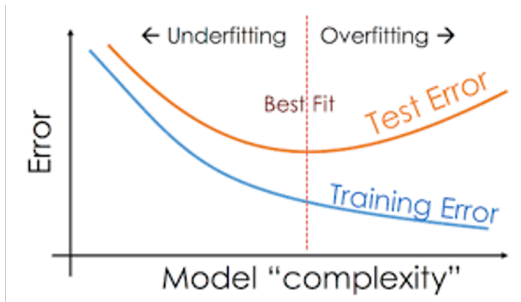
---
[a]E.g. mean absolute error, see page 39.

Linear model is **underfitting** while degree 20 polynomial is **overfitting**, the best fit for this problem is the degree 4 polynomial.

Overfitting and underfitting

We can consider that data is composed of:

► A main signal

► A noise signal


► Overfitting happens when the model captures the noise along with the main signal in data. (Degree 20 polynomial)

► Underfitting happens when a model is unable to capture the main signal in data. (Linear model)
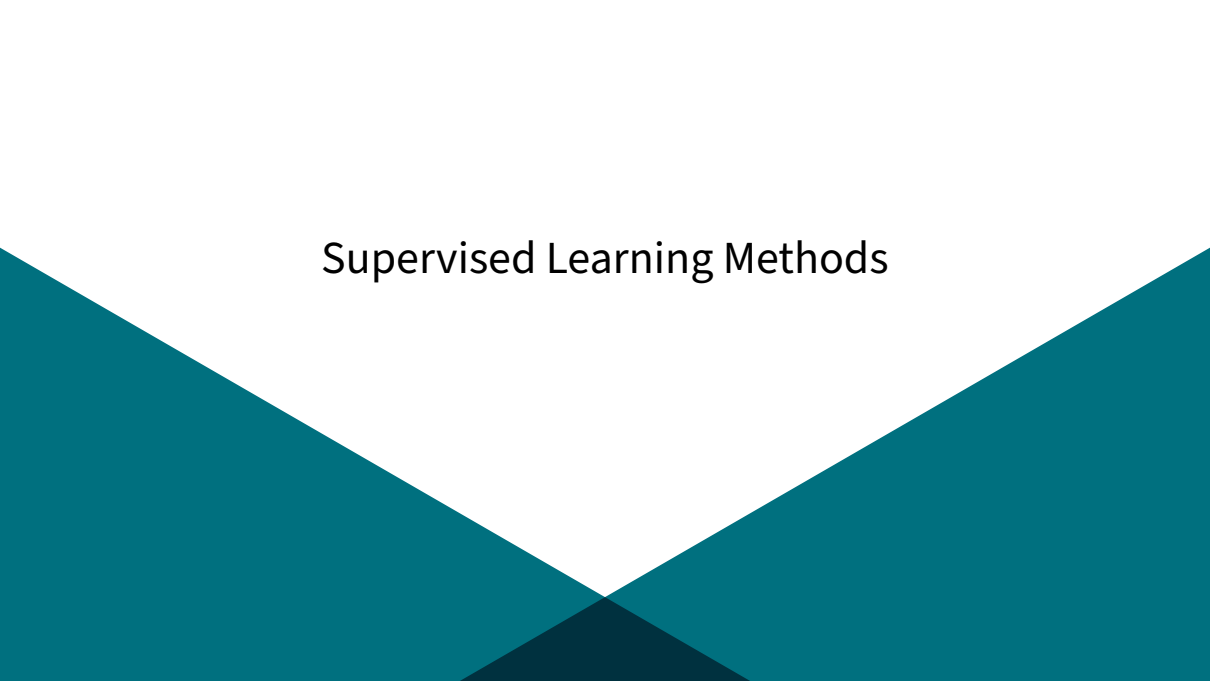
Cross-validation I

In general, adjusting your model based on your test set performance is bad practice. Cross-validation is a way to prevent overfitting without evaluating a model on the test set.

- ▶ Split your initial training data into k subsets.
- ▶ Iteratively train the algorithm on k-1 subsets while using the remaining fold as the "test" set (called validation set) to calculate the prediction error.
- ▶ Report the mean error over the k subsets.
- ▶ Cross-validation allows you to tune hyperparameters with only your original training set. This allows you to keep your test set as a truly unseen dataset for selecting your final model.

Cross-validation II
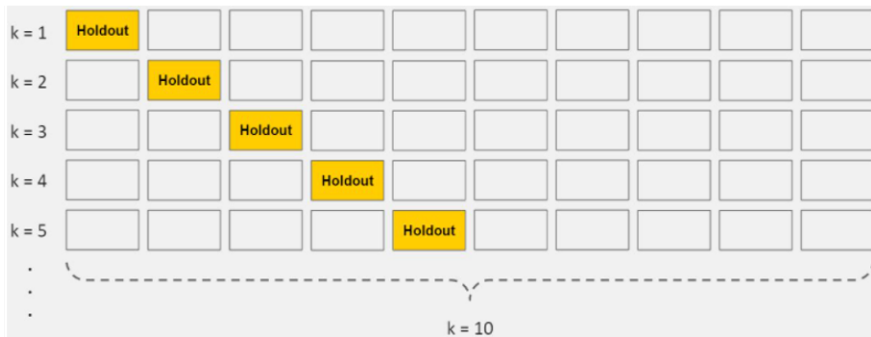
Supervised Learning Methods

There are many SL methods

- ▶ (Linear) regression
- ▶ Nearest neighbors
- ▶ Regression and decision trees
- ▶ Support vector machines / support vector regression
- ▶ Artificial neural networks $\implies$ deep learning
- ▶ Ensemble methods
- ▶ …

k-NN

▶ Find the *k* nearest neighbors with respect to Euclidean distance.

▶ Output the most frequent class (classification) or the average outputs (regression) among the *k* neighbors.
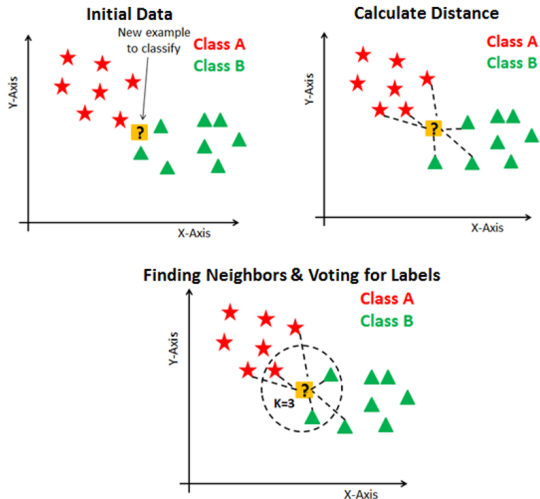
Decision Tree

▶ Choose the input that best separates the training set.

▶ Split the training set according to the chosen input.

▶ Proceed recursively until each object is correctly classified.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	High	Strong	Yes
D8	Sunny	Mild	Normal	Weak	No
D9	Sunny	Hot	Normal	Weak	Yes
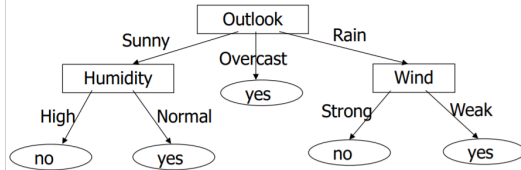
Regression tree with 2 real features and one output
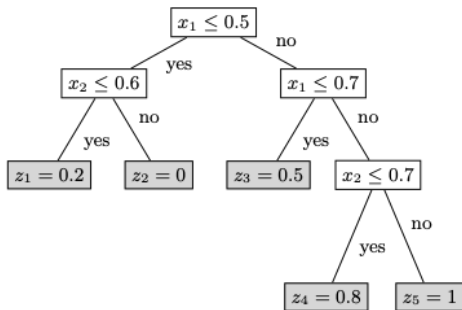
(a) A graphical model.
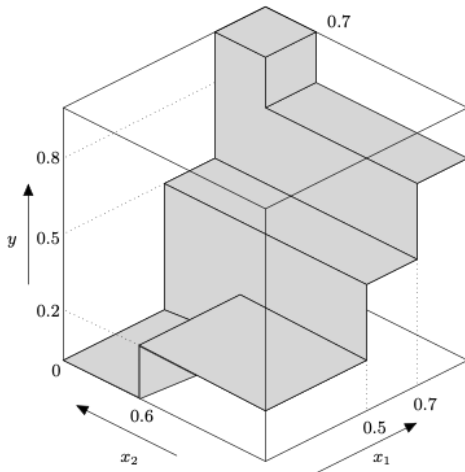
(b) The 3D corresponding representation.

Top-down induction of a regression tree (TDIRT).

1. *Initialization.*
   - Set $\mathcal{N} = \{(N_0, D_0)\}$, N_0 is the node number, D_0 the dataset.
   - Set $\mathcal{L} = \emptyset$, $\mathcal{I} = \emptyset$.

2. *Termination* if $\mathcal{N} = \emptyset$. Return the tree made of
   - the internal nodes and splits $\mathcal{I}$,
   - the leaves and associated values $\mathcal{L}$.

3. *Node Selection.*
   - Pop a node (N_i, D_i) out of $\mathcal{N}$.
   - Evaluate if the node must be split.

4. *Leaf assignment.*
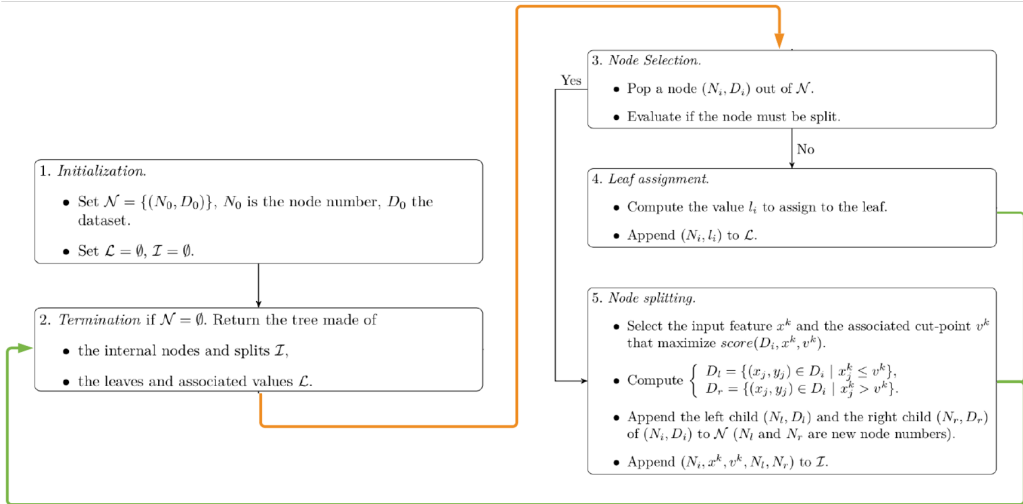   - Compute the value l_i to assign to the leaf.
   - Append (N_i, l_i) to $\mathcal{L}$.

5. *Node splitting.*
   - Select the input feature x^k and the associated cut-point v^k that maximize $score(D_i, x^k, v^k)$.
   - Compute $\begin{cases} D_l = \{(x_j, y_j) \in D_i \mid x_j^k \leq v^k\}, \\ D_r = \{(x_j, y_j) \in D_i \mid x_j^k > v^k\}. \end{cases}$
   - Append the left child (N_l, D_l) and the right child (N_r, D_r) of (N_i, D_i) to $\mathcal{N}$ (N_l and N_r are new node numbers).
   - Append $(N_i, x^k, v^k, N_l, N_r)$ to $\mathcal{I}$.

Yes

No

Decision Tree Comments

▶ If we use the mean squared error (MSE) criterion to estimate the accuracy of the tree predictor, defining l_i (step 4) as the mean value of the outputs of the objects constituting a leaf i is optimal with respect to empirical prediction error.

▶ To **split** a node i (step 5) , a test is defined by a feature x_k and a cut-off value v_k.

▶ All the objects satisfying the test $x_k > v_k$ are assigned to the right successor node and the remaining ones are assigned to the left successor node.

Splitting Score

▶ To find the best test, a score is computed for every input feature and for every possible cut-off value.

▶ For regression trees, a typical score measure is the relative decrease of output variance in the two successor nodes with respect to the output variance of the current node.

$$score(D_i, x^k, v^k) = \frac{var\{y|D_i\} - \frac{|D_l|}{|D_i|}var\{y|D_l\} - \frac{|D_r|}{|D_i|}var\{y|D_r\}}{var\{y|D_i\}}$$

▶ The test with the highest variance reduction is chosen.

Until When Do We Split?

▶ It is more complicated to assess if a node should be split, i.e., to mitigate the complexity of the model.

▶ A classical way is to stop splitting when the number of objects in a node is below a threshold value.

▶ Many other techniques have been developed to identify the tree of optimal complexity (pruning methods).

Steps to apply machine learning

Steps to solve a supervised learning problem

1. Collect and clean the data
2. Select a suitable model
3. Fit your model on the training set
4. Evaluate it on a validation set
5. Adjust and improve your model (Overfitting? Underfitting?)
6. *Repeat steps 3-5 until you are satisfied*
7. Evaluate your model on the test set

1. Data collection and cleaning

- ▶ Quantity: the more, the better
- ▶ Quality: Check if the data is well distributed
- ▶ Data rescaling (faster learning)
- ▶ Check missing values, wrong labeling, bad encoding
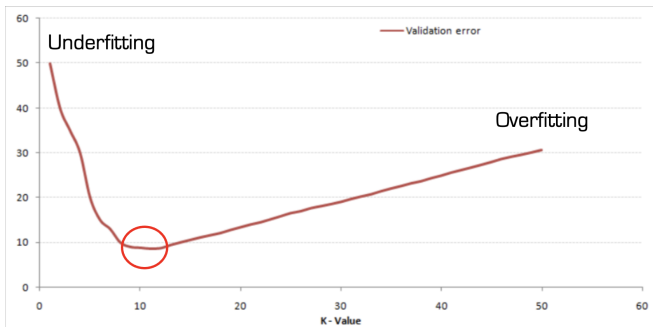- ▶ Feature engineering

Hyperparameter tuning I

Example: K-nn model Which number of neighbors k to choose ?
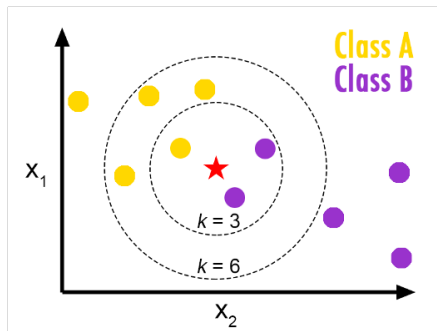
Hyperparameter tuning II

Use cross validation: try several values and pick the one that generalizes the best on the validation set

Classification metrics I

Imagine you have a classification problem with two classes *Sick* and *Not sick*.

To summarize your predictions and evaluate them, a *Confusion Matrix* displays in a tabular format the numbers of true positives (predicted Sick and is Sick), true negatives (predicted Not sick and is Not sick), false positives, and false negatives.

	Predicted	
Actual	A	B
A	85	15
B	10	90

Classification metrics II

Hence here Shown: True positives $TP = 85$, True negatives $TN = 90$
False positives $FP = 10$, false negatives $FN = 15$

From this information, we can compute a number of indicators such as

▶ the **Accuracy**, i.e. the number of correct predictions divided by the total number of predictions made:

$$A = \frac{TP + TN}{TP + TN + FP + FN} = \frac{85 + 90}{200} = 87.5\%$$

Classification metrics III

the **sensitivity**, i.e. the number of true positives detected divided by the total number of positive cases:

$$S = \frac{TP}{TP + FN} = \frac{85}{85 + 15} = 85\%$$

Notes:

- ▶ in this case the number of positive cases = the number of negative cases (balanced dataset); it is in general not the case and you must take care of it
- ▶ a confusion matrix can be generalized to a multiclass problem.

Regression metrics I

Some common error metrics are the

- **Mean absolute error**: $MAE = \frac{1}{N} \sum_{j=1}^{N} |y_j - \hat{y}_j|$
- **Mean squared error**: $MSE = \frac{1}{N} \sum_{j=1}^{N} \left(y_j - \hat{y}_j\right)^2$

They should be as small as possible. The MAE penalizes equally large and small errors, while the MSE penalizes large errors more than small errors.

The **Coefficient of determination**, R^2, should be as close as possible to 1, and weights errors :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Regression metrics II

Where:

- ► SS_{res} is the sum of squares of residuals (the sum of squared errors).
- ► SS_{tot} is the total sum of squares.
- ► y_i are the observed values.
- ► $\hat{y}_i$ are the predicted values.
- ► $\bar{y}$ is the mean of the observed values.

It indicates the proportion of the variance in the dependent variable (y) that is predictable from the independent variable(s) (x)

Wrap up example

► This code demonstrates a basic k-Nearest Neighbors (k-NN) classification using the 'scikit-learn' library in Python.

► It loads the Iris dataset, splits it into training and testing sets, scales the data, trains a k-NN classifier, and evaluates its accuracy.

```
https://colab.research.google.com/drive/
1v0BMKsY26pPStqNwEpoqOnbWTitRrB0C?usp=sharing
```

Application

Fitting the power curve of a wind turbine

See Google Colab. `https://colab.research.google.com/drive/`
`1UxhfjLj8UMz5EHqnYbizIXcROuo_9ItM?usp=sharing`