Inferential statistics

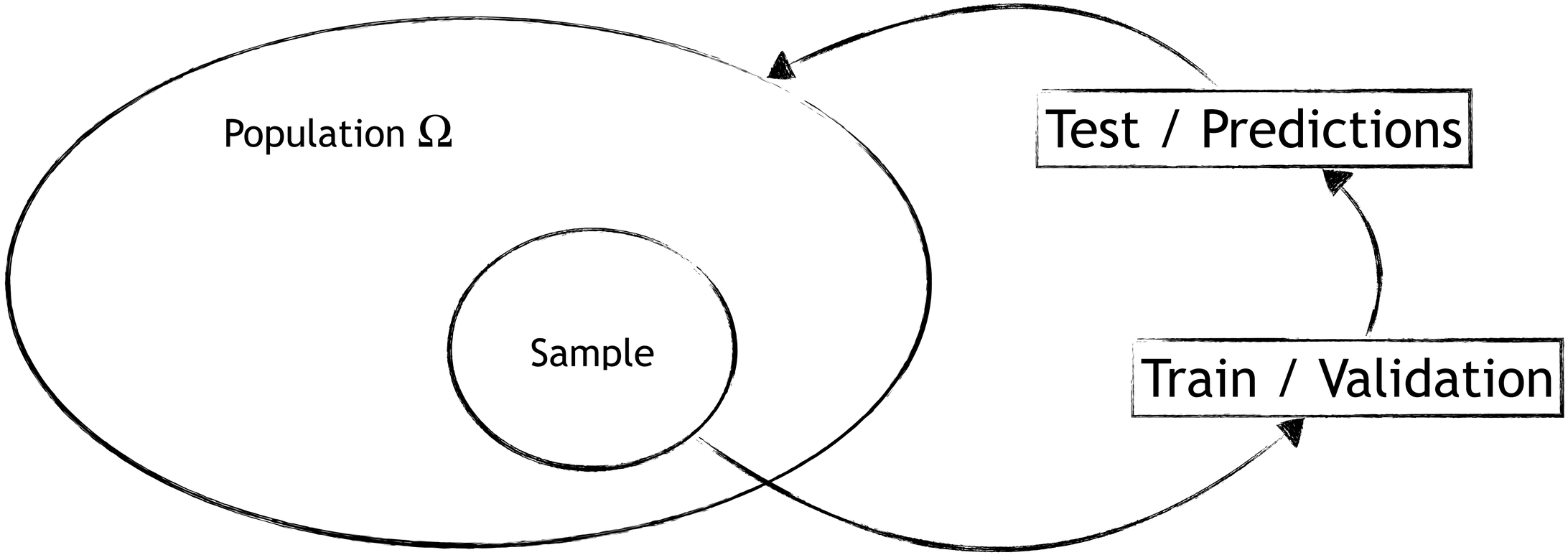
Applied Data Analysis (ADA) - May 2025

Nomades Advanced Technologies
Gaspard Villa

- Monday : Metrics and training process
  - Supervised & Unsupervised learning
  - Evaluation metrics
  - Training process
- Tuesday : Linear models
- Wednesday : Support Vector Machine models
- Thursday : Decision trees
- Friday : Time series analysis

Inferential statistics

Definition : Inferential statistics is the idea of drawing conclusions about a population base on data from a sample.



Supervised learning

Definition : Supervised learning is a type of machine learning where the model is trained on a labeled dataset, meaning each input data point is paired with the correct output (target).

=> The goal is for the model to learn the mapping from inputs to outputs and make predictions on new unseen data.

Supervised learning
- Example -

- Spam detection in mails => Input : email text / Label : spam or not spam
- Prediction of house prices => Input : number of rooms, area, location / Label : price.
- Medical diagnosis => Input : patient symptoms and test results / Label : disease present or not.
- Image classification => Input : image pixels / Label : “cat”, “dog”, etc…

Unsupervised learning

Definition : Unsupervised learningn is a type of machine learning where the model is given unlabeled data.

=> The goal is to find patterns, groupings, or structures within the data without knowing the “correct” answers.

Unsupervised learning
- Example -

- Customer segmentation => Task : grouping customers by purchasing behaviour.
- Anomaly detection => Task : Finding fraudulent transactions in banking data.
- Topic modelling => Task : discovering themes in a collection of new articles.
- Dimensionality reduction => Task : visualizing high-dimensional data (e.g., genetics, images) using PCA.

Supervised learning
- Regression methods -

Definition : It models the relationship between one or more independent variables (features) and a continuous dependent variable (target).

=> It’s a supervised learning method.

Regression methods
- Illustration of some models -

- *Simple linear regression :* single feature vs target.
- *Multiple linear regression :* multiples features to predict the target.
- *Polynomial regression :* Useful for modelling non-linear trends.
- *Regularized regression :* Penalizes weights to prevent overfitting and manage multicollinearity => Improves generalisation.

Supervised learning
- Classification methods -

Definition : It models the relationship between one or more independent variables (features) and a categorical dependent variable (target).

=> It’s also a supervised learning method.

Classification methods
- Example -

- *Logistic regression :* Probability model for binary tasks.
- *Decicision tree :* Tree based model, interpretable model.
- *Random forest :* multiple random trees.
- *Support Vector Machines (SVM) :* Find optimal separating hyperplane.
- *Neural Networks :* Flexible and powerful (especially for high-dimensional inputs).

Evaluation of models
- Metrics -

How do you know you model is working better than the others ?

➡ You define metrics to evaluate the model’s performance.

➡ But can you use the same metrics for regression and classification tasks ?
➡ How do you chose your metric ?

Evaluation of models
- Metrics for regression -

Goal : We need to have a metric that evaluate how close predictions of continuous variables are to the actual target variables.

1 - Error terms : Let y_i be the true value and $\hat{y}_i$ the predicted value:

$$\epsilon_i = y_i - \hat{y}_i$$

$$\Rightarrow \text{ Error} = \sum_{i=1}^{n} \epsilon_i$$

Evaluation of models
- Metrics for regression -

2 - Mean Absolute Error (MAE) : It measures average absolute difference.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Remark : Easy to interpret but doesn't penalise large errors.

Evaluation of models
- Metrics for regression -

3 - Mean Squared Error (MSE) : It measures the square of errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

Remark : Commonly used in optimisation!

Evaluation of models
- Metrics for regression -

4 - Root Mean Squared Error (RMSE) : Its measure has the same units as the target variable.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

Remark : More sensitive to outliers that MAE.

Evaluation of models
- Metrics for regression -

5 - R-Squared (R^2) : It measures a proportion of variance explained.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Remark : 0 means that the prediction is not better than the mean as a naïve prediction.

Evaluation of models
- Metrics for regression -

6 - Mean Average Percentage Error (MAPE) : It measures a prediction accuracy of the models.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Remark : Very intuitive error interpretation in terms of relative error.

Evaluation of models
- Metrics for classification -

Goal : We need to have a metric that evaluate the quality of the prediction given by a specific model.

0 - Basic concepts:

- *True Postive (TP) :* true value = 1, predicted value = 1
- *True Negative (TN) :* true value = 0, predicted value = 0
- *False Postive (FP) :* true value = 0, predicted value = 1
- *False Negative (FN) :* true value = 1, predicted value = 0

Evaluation of models
- Metrics for classification -

1 - Accuracy: It measures the proportion of correct predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Remark : Issues when facing imbalanced classes.

Evaluation of models
- Metrics for classification -

2 - Precision : It measures how many predicted positives were correct.

$$\text{Precision} = \frac{???}{???}$$

Remark : Useful for imbalanced classes.

Evaluation of models
- Metrics for classification -

2 - Precision : It measures how many predicted positives were correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Remark : Useful for imbalanced classes.

Evaluation of models
- Metrics for classification -

3 - Recall : It measures how many actual positives were captured.

$$\text{Recall} = \frac{???}{???}$$

Remark : Useful for imbalanced classes.

Evaluation of models
- Metrics for classification -

3 - Recall : It measures how many actual positives were captured.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Remark : Useful for imbalanced classes.

Evaluation of models
- Metrics for classification -

4 - F1-score : It measures harmonic mean of precision and recall.

$$\text{F1-score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Remark : Useful for imbalanced classes.

Evaluation of models
- Metrics for classification -

5 - Confusion matrix : It shows all 4 outcomes (TP, TN, FP, FN) in a 2x2 table.

		Prediction	
		0	**1**
Truth	**0**	TN	FP
	1	FN	TP

Evaluation of models
- Metrics for classification -

6 - Log-loss (cross-entropy loss) : It measures probabilistic confidence in predictions.

$$\text{Log-loss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right]$$

Remark : Commonly used in optimisation!

Underfitting

Definition : The model is too simple to capture underlying patterns. It can be visualised by a poor performance both on training and validation sets.

=> Make the model more complex by modifying its parameters or completely change the model.

Overfitting

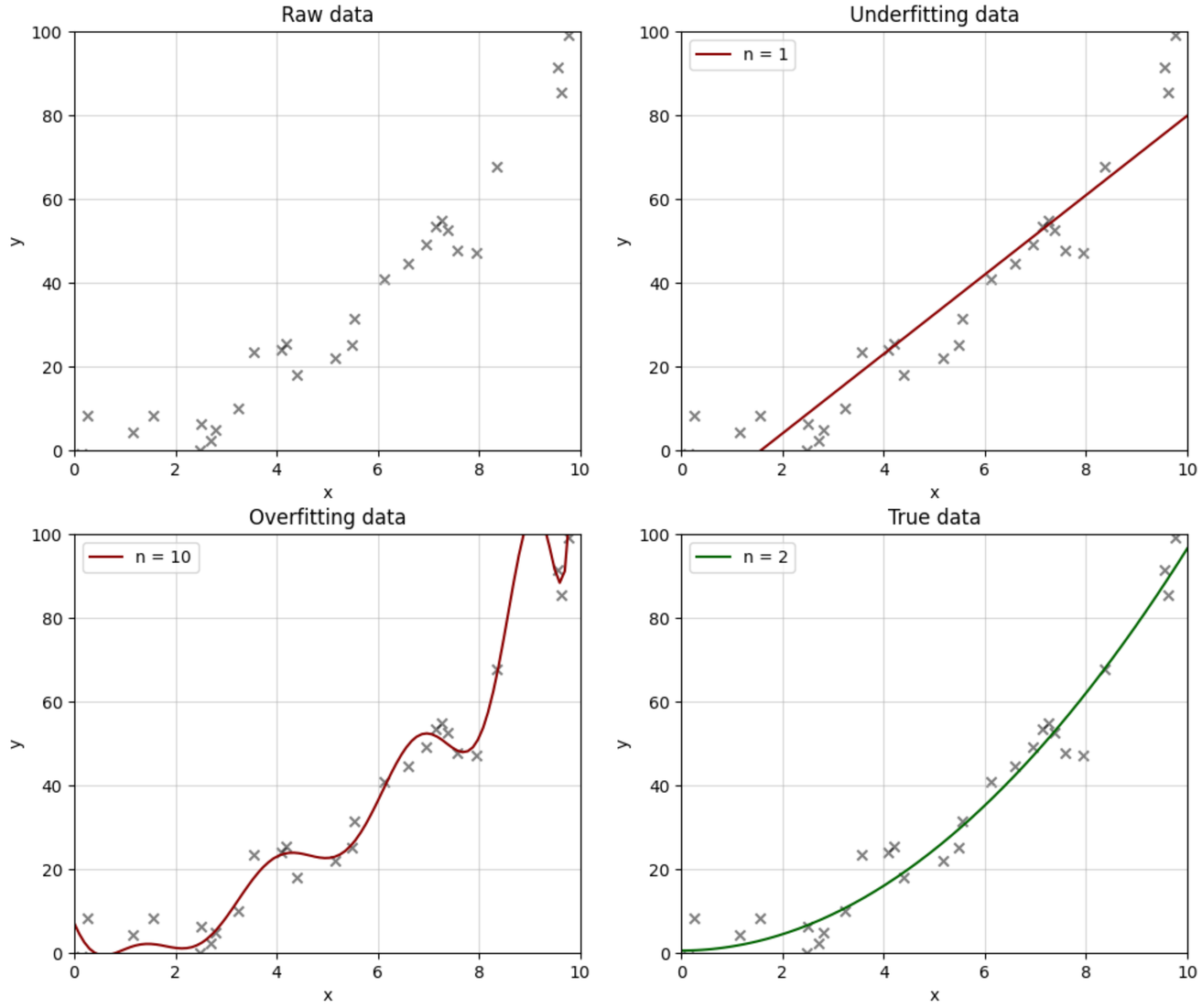
Definition : The model learns noise and details from the training data. It can be visualised by a good performance on training set but poorly on validation set.

=> Simplify the model, regularisation or training data set not well defined regarding the validation set (split to redo).

Underfitting / Overfitting

Practice : Use the data set `overfitting_data.csv`, train a Polynomial Regression model on it and plot the results. Do it for $n = 1, 2$ and 10.

Underfitting / Overfitting



Practice
- Cross-validation using sklearn -

Training process
- General overview -

- ***Data preprocessing :*** Cleaning, normalisation, encoding, split, etc… (EDA)
- ***Model selection :*** Chose one or multiples model types.
- ***Loss function definition :*** Quantify how wrong the model is (not performance!)
- ***Optimization :*** Algorithms to minimize loss such as gradient descent.
- ***Evaluation :*** Measure performance on training and validation sets.

Practice
- KNN -

- Monday : Metrics and training process
- Tuesday : Linear models
  - Linear regression
  - Polynomial & Interaction terms regression
  - Ridge & Lasso regression
- Wednesday : Support Vector Machine models
- Thursday : Decision trees
- Friday : Time series analysis

Linear Models

Linear regression

Defintion : It’s a classic model for regression task, the idea is to predict a continuous target variable with a linear combination of the features (categorical or continuous).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

Assumptions : Linearity, independence, normality of errors.

Linear regression
- How it trained technically ? -

To compute the « right » parameters of the model, we are aiming to minimise a loss. This loss can be MSE like:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

How to minimise it ?

Linear regression
- Gradient descent -

We start from a random parameter β and we iterate from it until convergence of the method. More formally it gives a iteration n the following:

$$\beta_{n+1} = \beta_n - \alpha \nabla_\beta \text{MSE}$$

$$= \beta_n - \alpha \frac{2}{n} X^T (X\beta - y)$$

Linear regression
- Practice -

Exercice : Load the data set `load_diabetes` from sklearn library, and train and test a linear regression model on this data set.

Polynomial and interaction terms regression

Defintion : We stick to a linear regression architecture but adding new feature to our data set. We add features combinations (e.g. x_i^2 or $x_i x_j$) to our set of features and do the same as before.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_2^2 + \epsilon$$

Remark : The main work here is done in EDA to determine which interaction should be included in your set of features

Ridge regression

Defintion : We face overfitting with linear regression and there is no inside parameters to play with to compensate this. Then, we use « regularizers ». Here we use L2-regularization, also called **Ridge regression :**

$$\text{loss} = \text{MSE} + \lambda \sum_i \beta_i^2$$

Specificities : Encourage small weights (but not zero) which helps keeping features when they are all important, keep the model smooth and reduce sensitivity to individual features.

Lasso regression

Defintion : Same situation as before but instead we use L1-regularization:

$$\text{loss} = \text{MSE} + \lambda \sum_i |\beta_i|$$

Specificities : Encourage weights to zero (sparsity), useful for feature selection, model robust when only few features matter.

Ridge and Lasso regressions
- Practice -

Exercice : Still on the data set `load_diabetes`, train and test Ridge and Lasso regression models and cross-validate the best parameter to use for λ.

- Monday : Metrics and training process
- Tuesday : Linear models
- Wednesday : Support Vector Machine models
  - Logistic regression
  - Support Vector Machine
  - Support Vector Regression
- Thursday : Decision trees
- Friday : Time series analysis

Correction of the exercises

Logistic regression

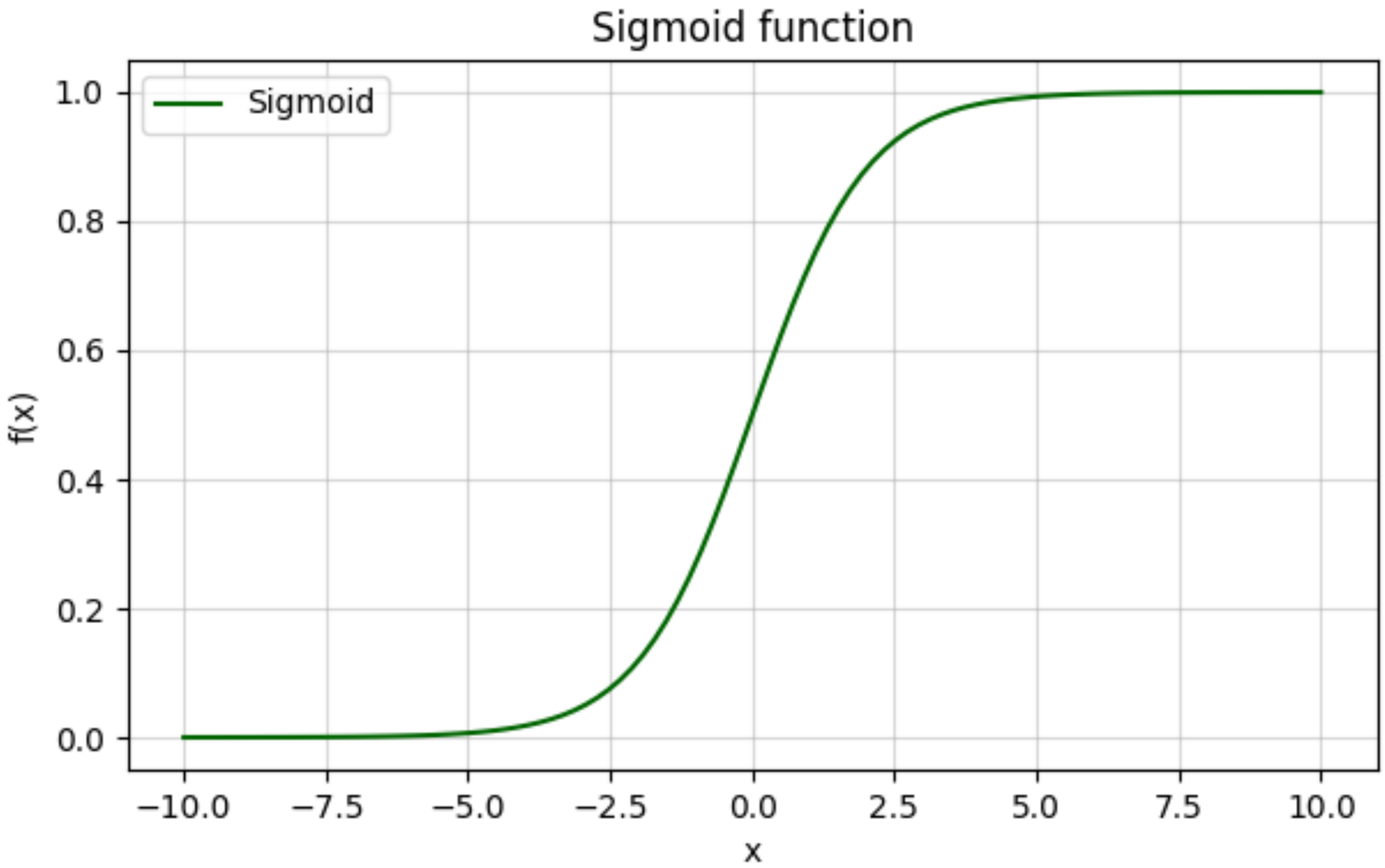
Definition : The logistic regression method is not for regression tasks but for classification tasks. The idea is to start from logistic function defined as:

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

And we define t as a linear combination of the exploratory variable such as $t = \beta_0 + \beta_1 x$ inducing (probability x belongs to class 1):

$$\mathbb{P}[y = 1 \,|\, x] = p_x = \sigma(\beta_0 + \beta_1 x) = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

Sigmoid function



Logistic regression

Then, we use the log-loss (or cross-entropy loss) to train the logistic regression model. It can be defined as follow :

$$\mathscr{L}(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right]$$

Remark : Commonly used when face to binary classification.

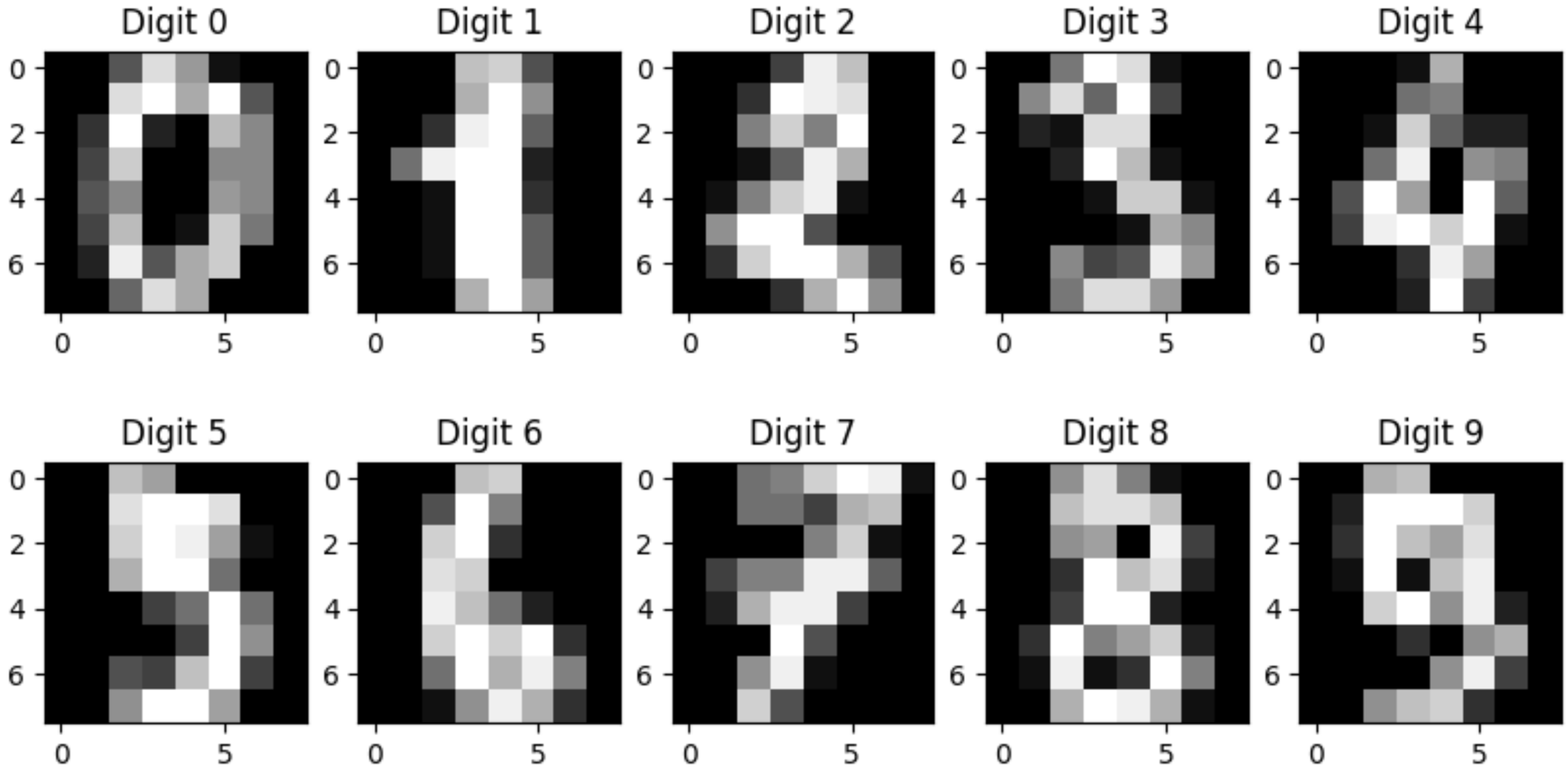
Multinomial logistic regression

We follow the same idea as logistic regression but we are facing multi-class target instead of binary, i.e. $y \in \{0,1,\ldots,K\}$ and we note:

$$p_{ij} = \mathbb{P}\left[y_i = j \,|\, x\right]$$

for each class $j \in \{0,1,\ldots,K\}$ and element $i = 1,\ldots,n$.

Some practice
- What model do you use ? -



Support Vector Machines
- SVM -

Definition : Intuitively, you have a set of points with different labels you want to classify. The Support Vector Machine (SVM) models are looking for an hyperplane that separates the classes.

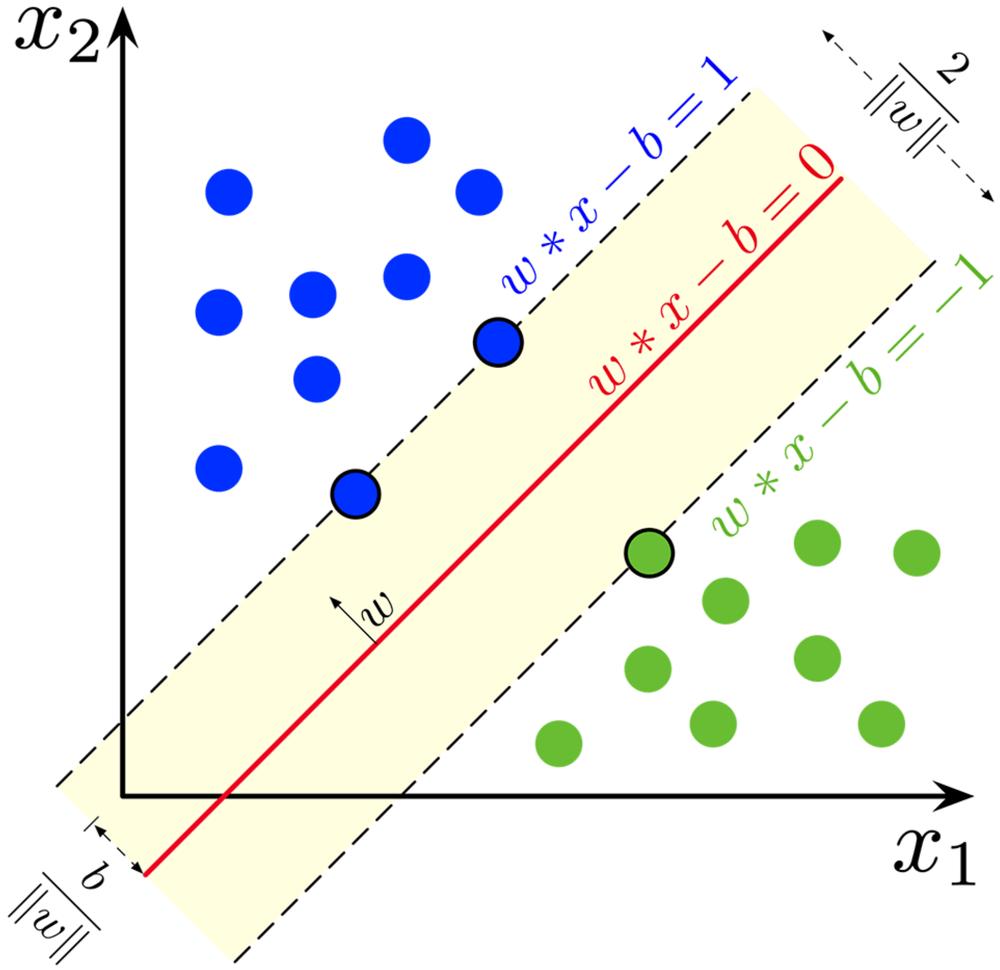
Remark : Here we consider binary classification with target variable $y \in \{-1,1\}$.

Support Vector Machines
- SVM -

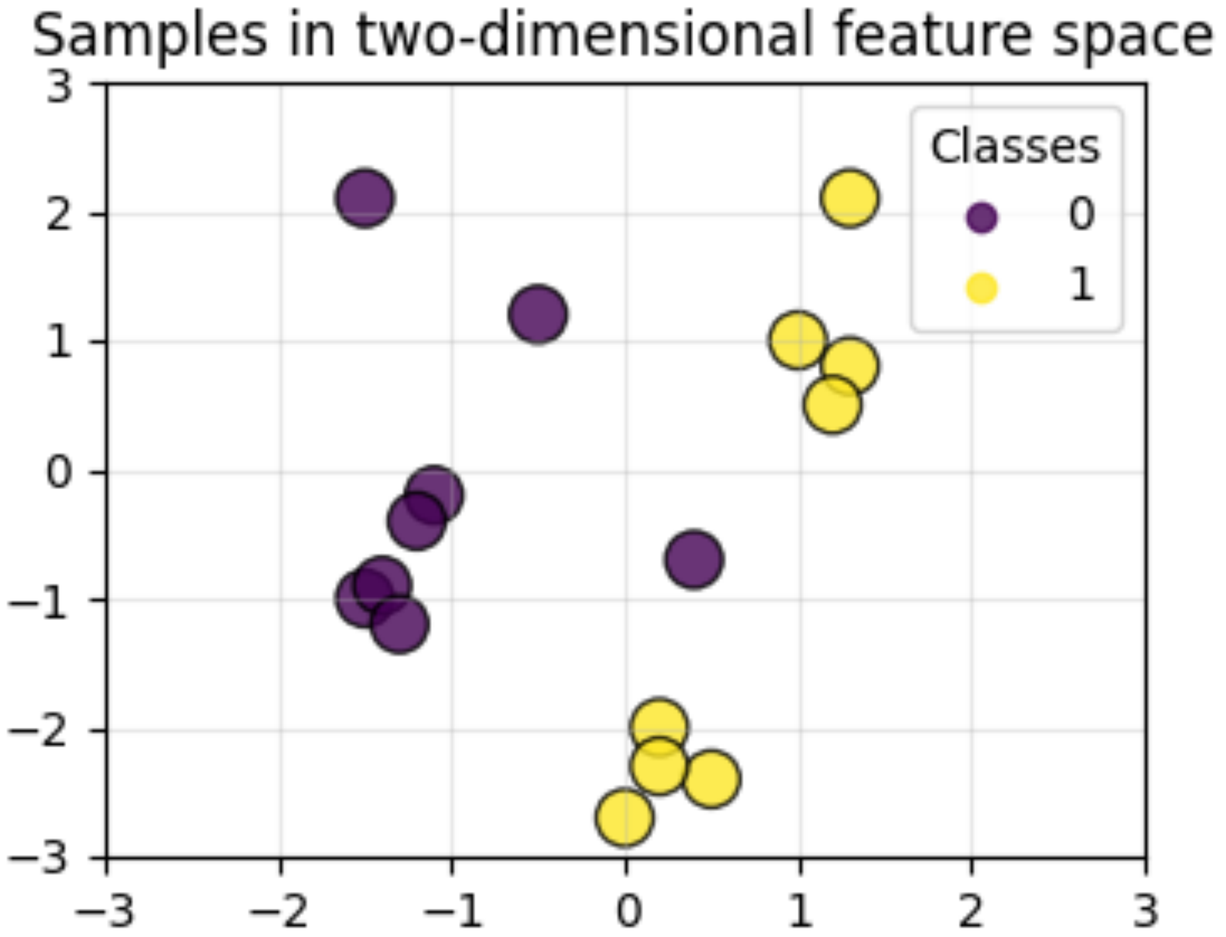
The formula to minimise for the SVM process is given by the following:

$$\min_{w,b} \frac{1}{2}\|w\|^2, \text{ such that } y_i(w^T x_i - b) \geq 1 \text{ for all } i.$$

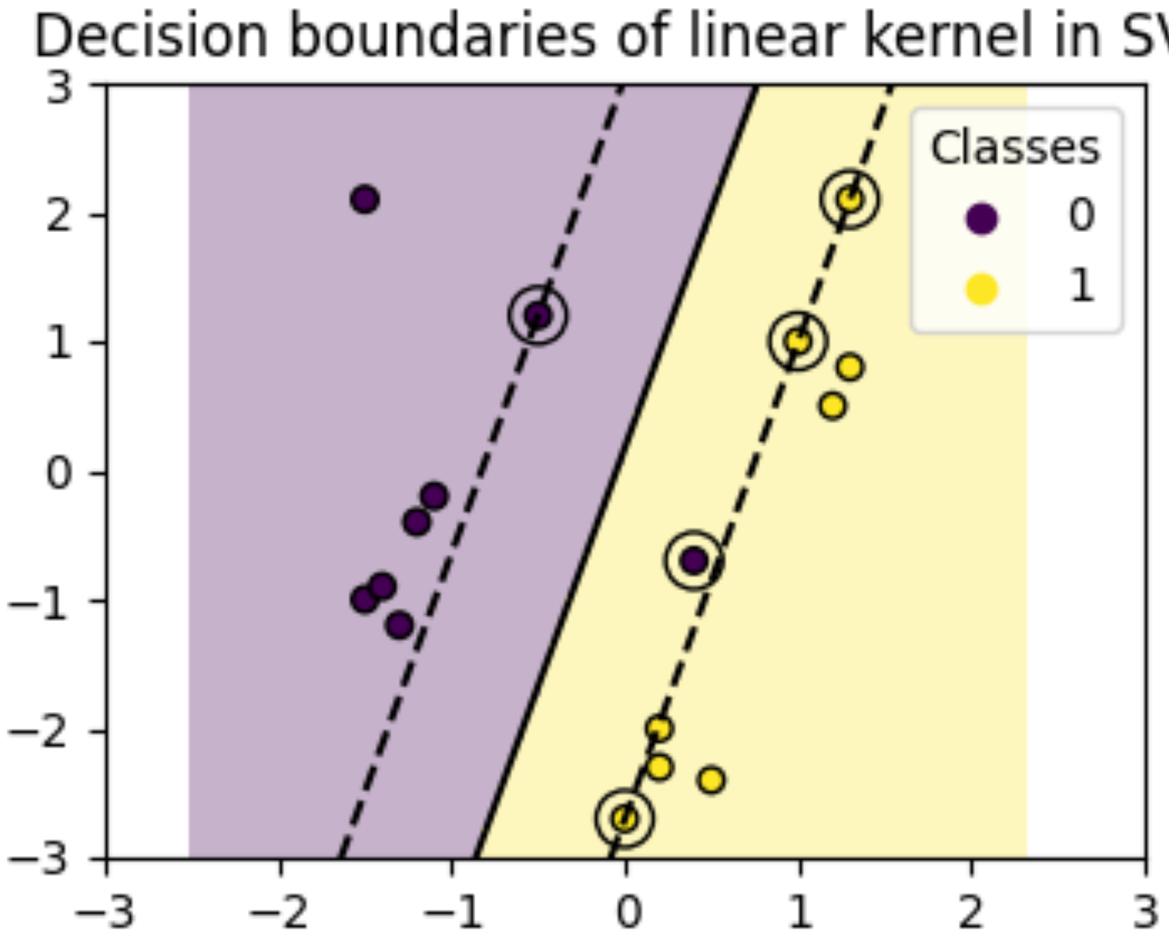
Support Vector Machines
- SVM -



SVM
- Example -


Samples in two-dimensional feature space
Classes
0
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

SVM
- Example -


Decision boundaries of linear kernel in SVC
Classes
0
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

SVM
- Slack variables -

In real world data set, there is not perfect plane separating the classes, then we have to accept the fact that some of the points violates the margin constraint :

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

such that $\xi_i = \max\left(0, 1 - y_i(w^T x_i - b)\right)$ for all i.

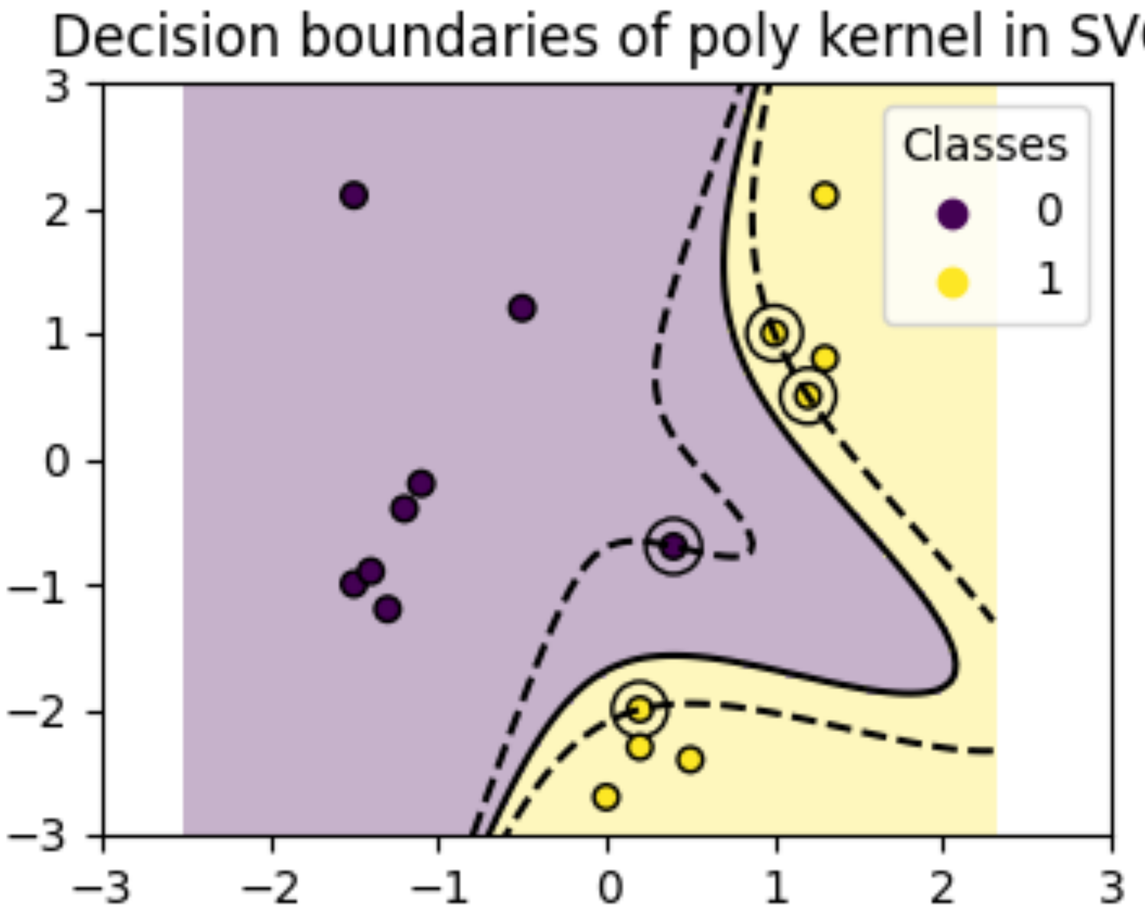
Remark : Large value of C => overfitting /vs/ low value of C => underfitting.

SVM
- Kernels -

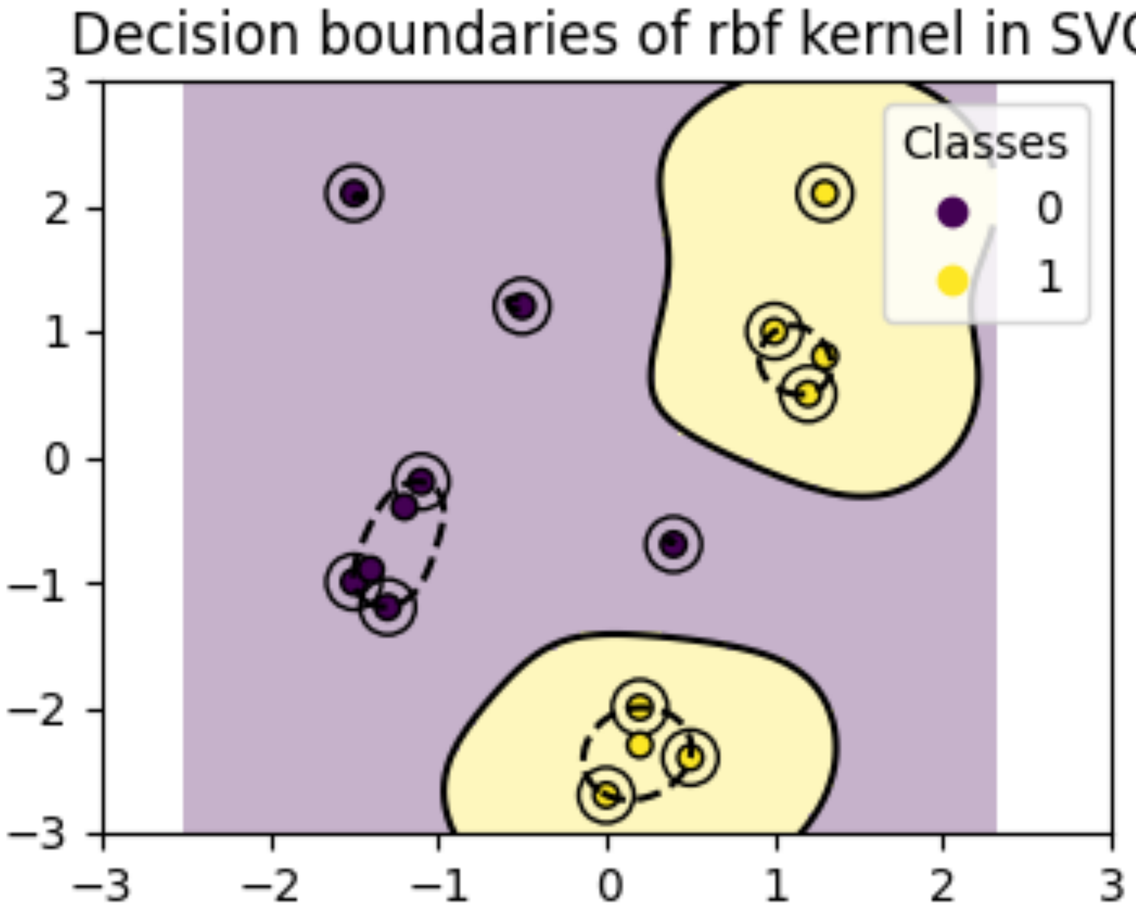
SVM can model non-linear decision boundaries using kernels. Some common kernels are used for SVM such as:

- Linear : $K(x, x') = x^T x'$
- Polynomial : $K(x, x') = (x^T x' + c)^d$
- RBF (Gaussian) kernel : $K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$
- Sigmoid kernel : $K(x, x') = \tanh(\kappa x^T x' + c)$

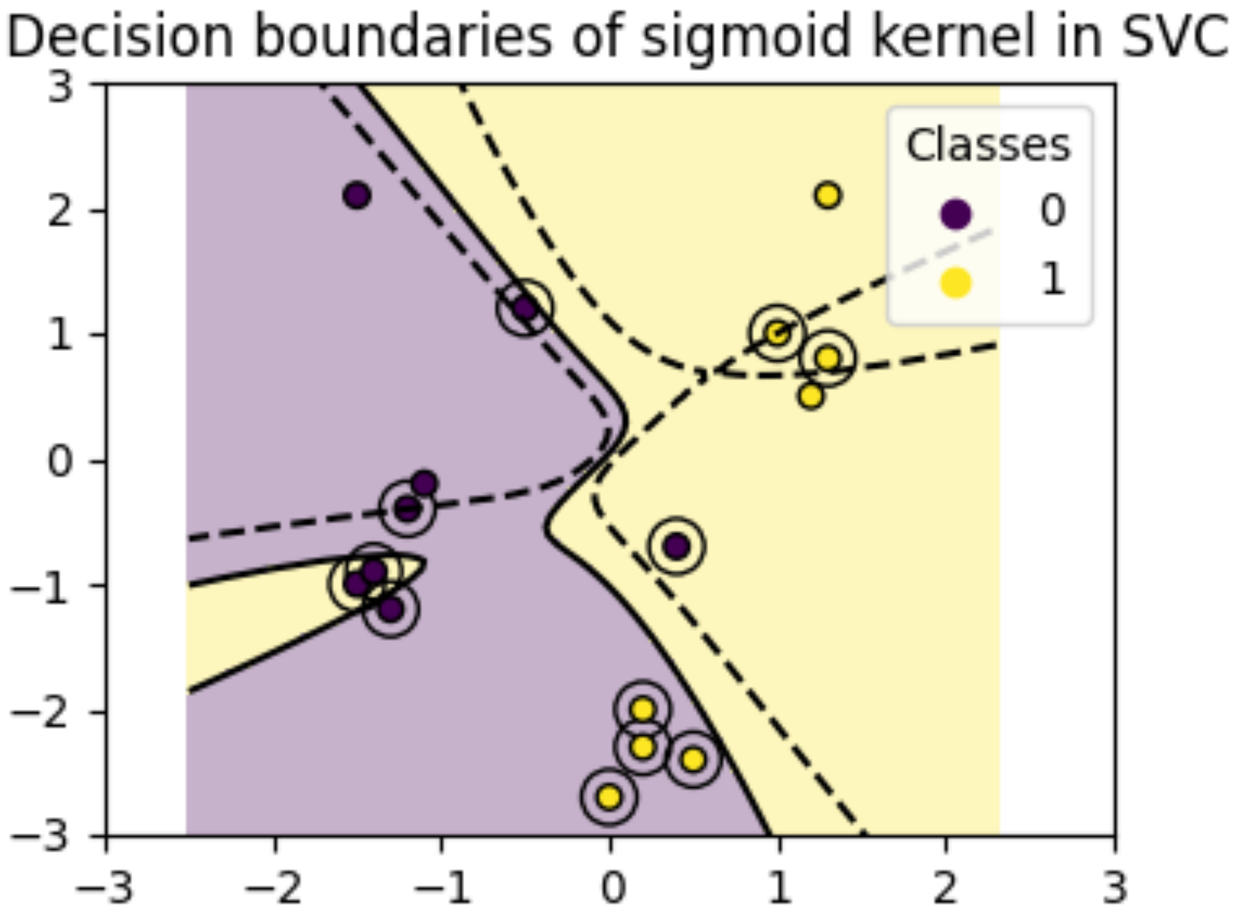
SVM
- Example -


Decision boundaries of poly kernel in SVC
Classes
0
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

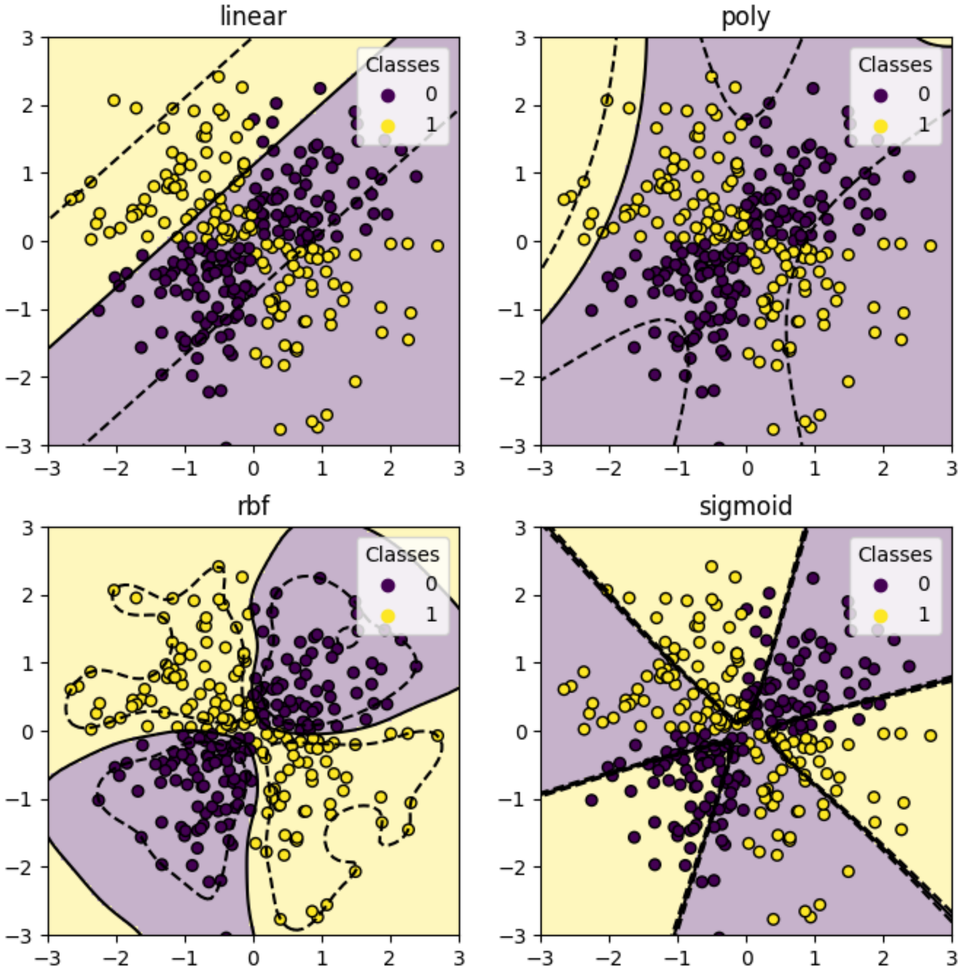
SVM
- Example -


Decision boundaries of rbf kernel in SVC
Classes
0
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

SVM
- Example -


Decision boundaries of sigmoid kernel in SVC
Classes
0
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

SVM
- Example -



SVM
- Practice -

Exercice : Train and validate the best kernel with the best parameters on the data set `load_digits` from sklearn. Compare your results with the ones found with logistic regression.

Support Vector Regression
- SVR -

Support Vector Regression is the regression counterpart of SVM. Instead of classifying data points, SVR predicts a continuous value.

Remark : It uses the same principles as SVM: maximizing margin, kernel trick, and regularisation.

- Monday : Metrics and training process
- Tuesday : Linear models
- Wednesday : Support Vector Machine models
- Thursday : Decision trees
  - Decision tree
  - Random forest
- Friday : Time series analysis

Correction of the exercises

Decision tree

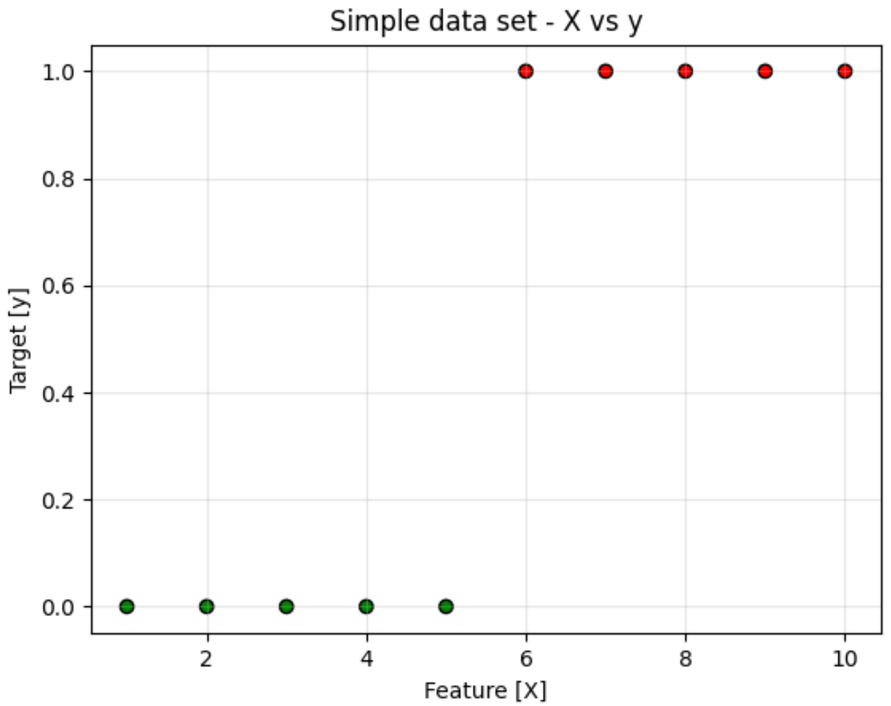
Definition : A decision tree is a flowchart-like structure used for classification and regression tasks.

Goal : The idea is to split the data set into subsets based on feature values to increase the homogeneity of the target variable (reduce the variance).

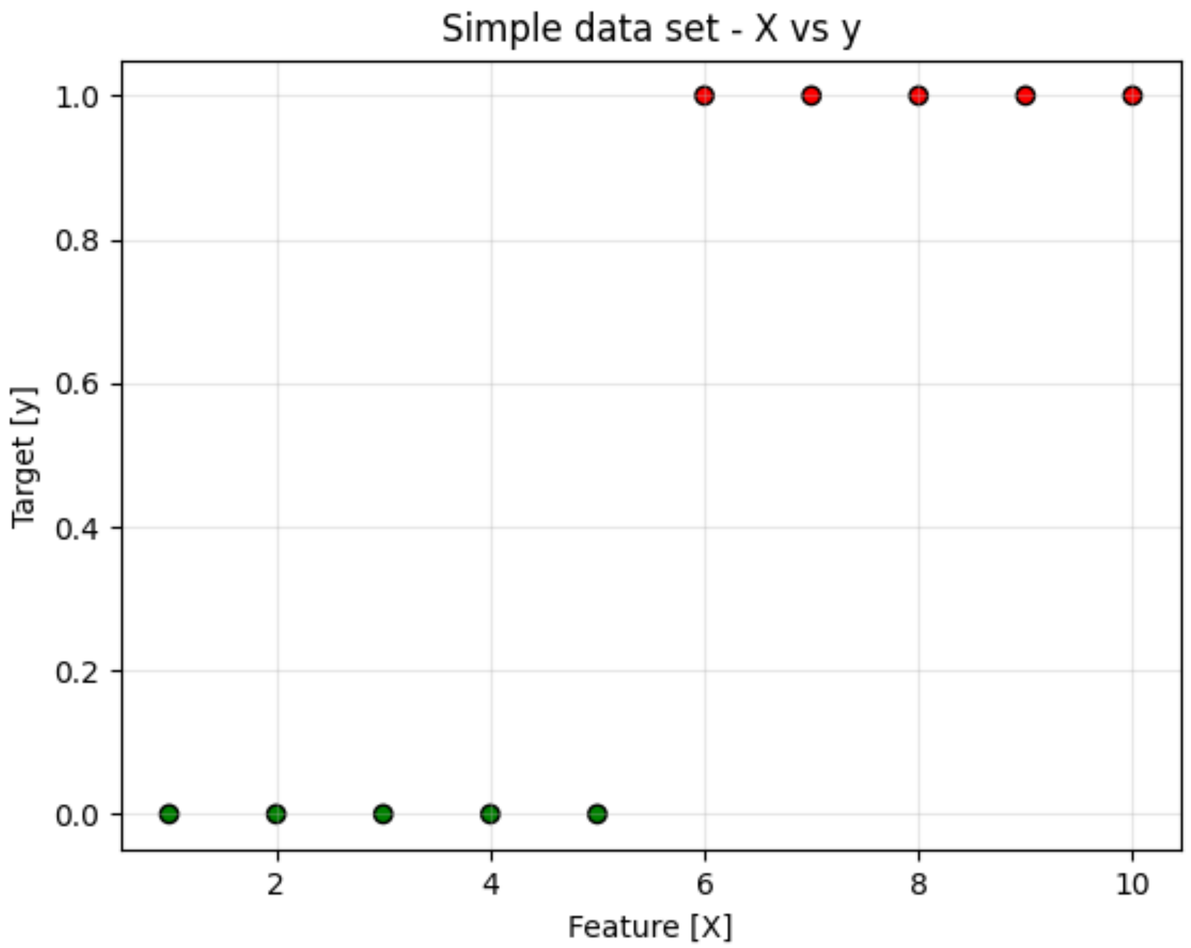
Decision tree
- How does it work ? -

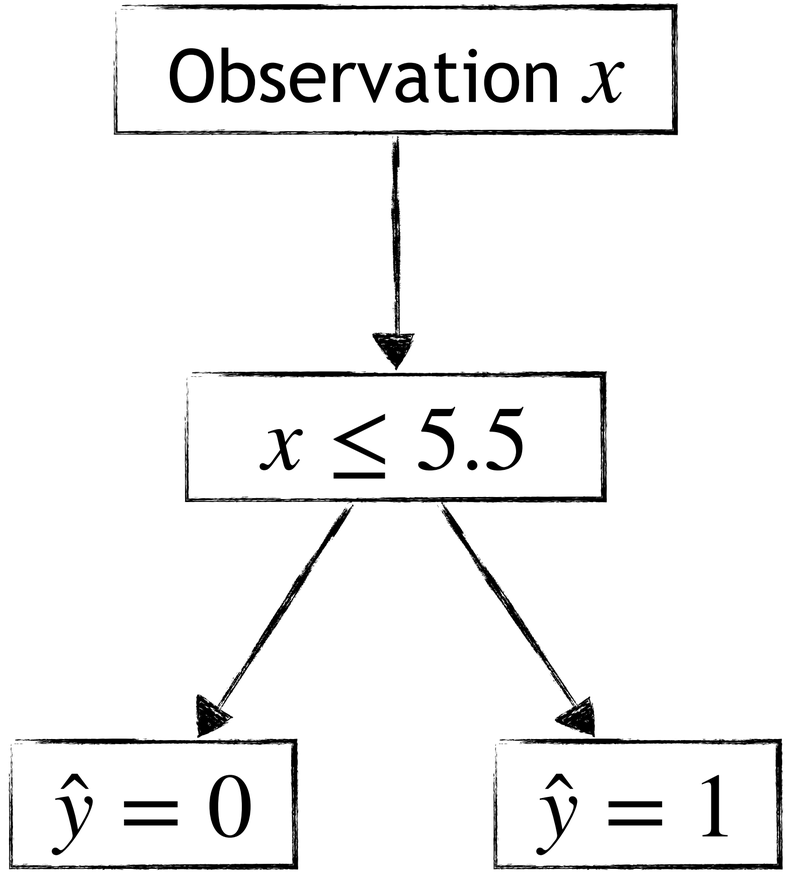
Idea : Suppose you have a binary classification task $\{0,1\}$ with a single continuous feature. Your data set has the following format :

ID	X	y
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0
5	6	1
6	7	1
7	8	1
8	9	1
9	10	1

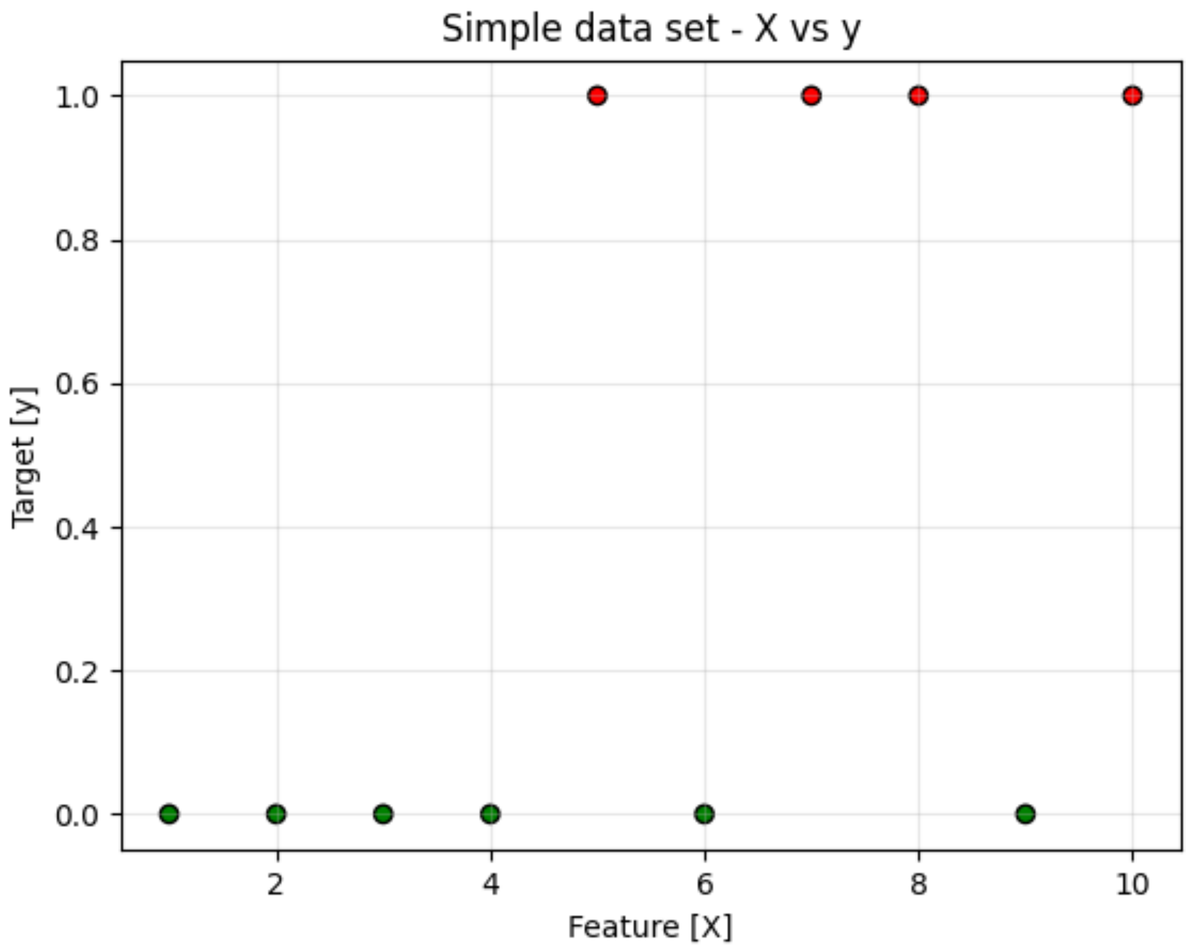


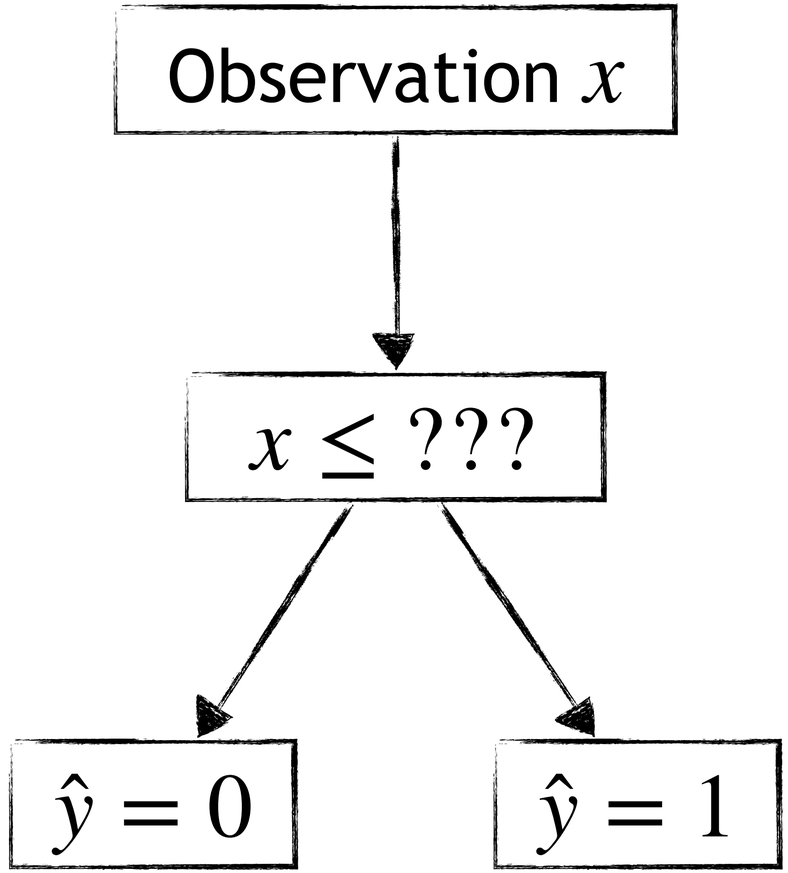
Decision tree
- How does it work ? -





Decision tree
- How does it work ? -





Decision tree
- How does it work ? -

1	2	3	4	5	6	7	8	9	10
0	0	0	0	1	0	1	1	0	1

Decision tree
- How does it work ? -

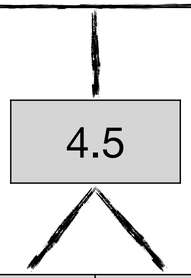
Mid-value

(4+5) / 2 = 4.5

1	2	3	4	5	6	7	8	9	10
0	0	0	0	1	0	1	1	0	1

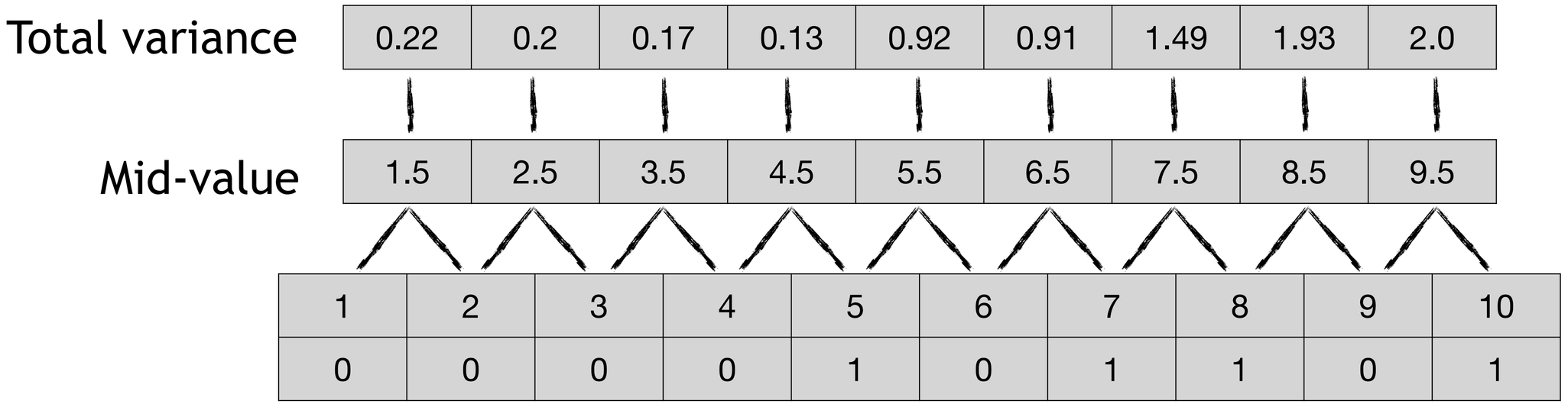
Decision tree
- How does it work ? -

$$\text{Total variance} = \frac{n_L}{n}\text{Var}(S_L) + \frac{n_R}{n}\text{Var}(S_R)$$

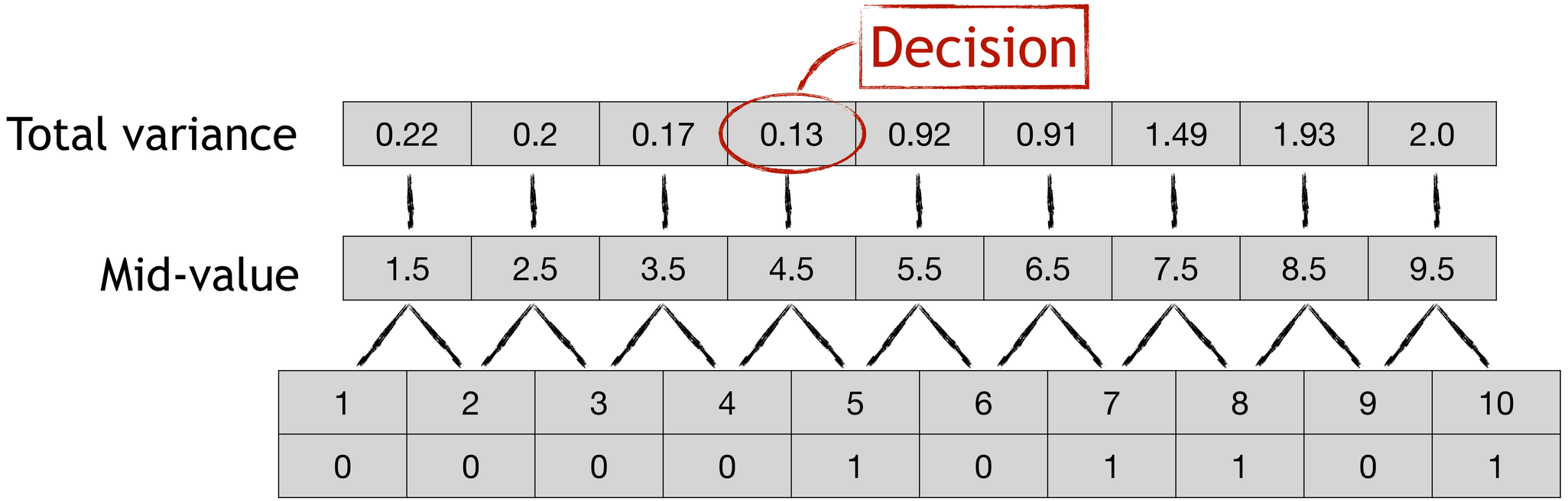


1	2	3	4	5	6	7	8	9	10
0	0	0	0	1	0	1	1	0	1

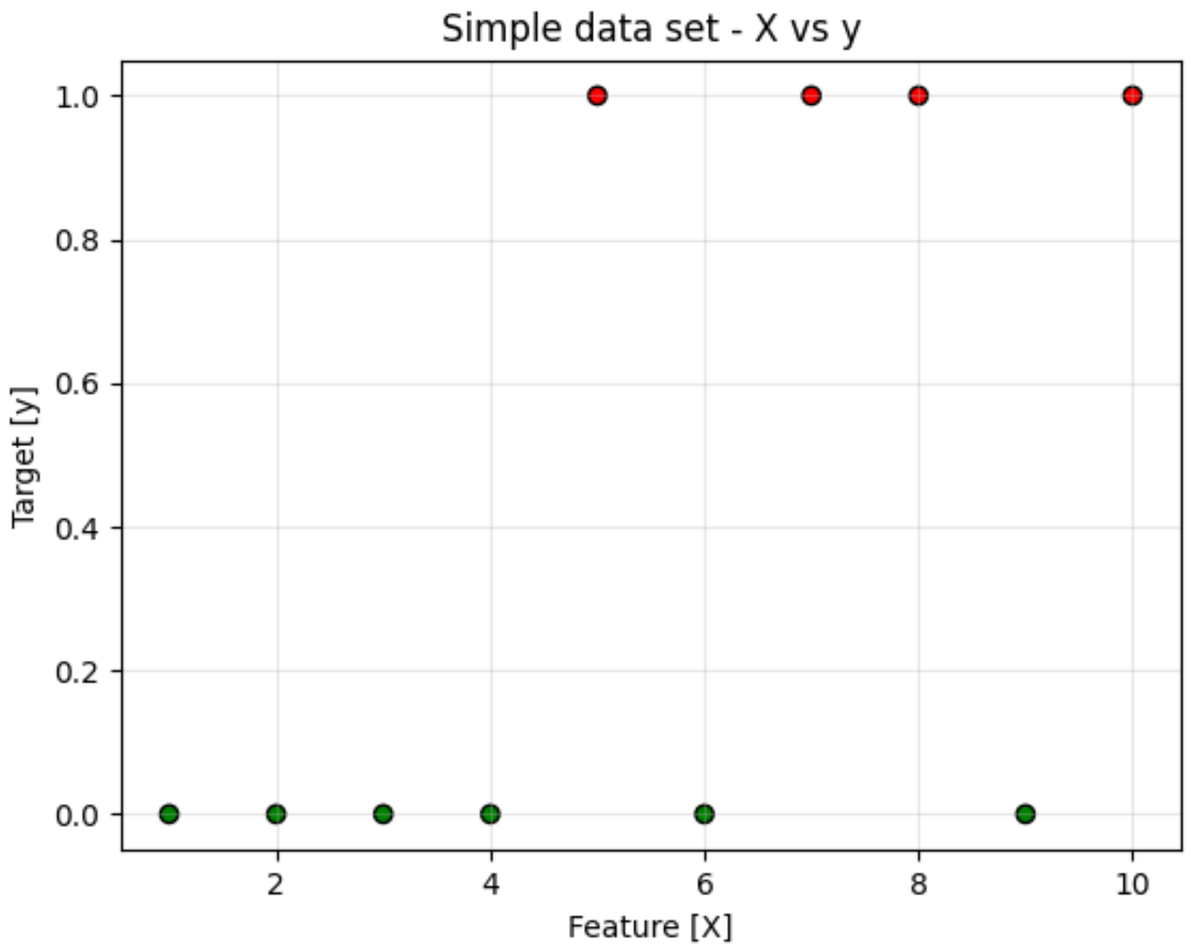
Decision tree
- How does it work ? -

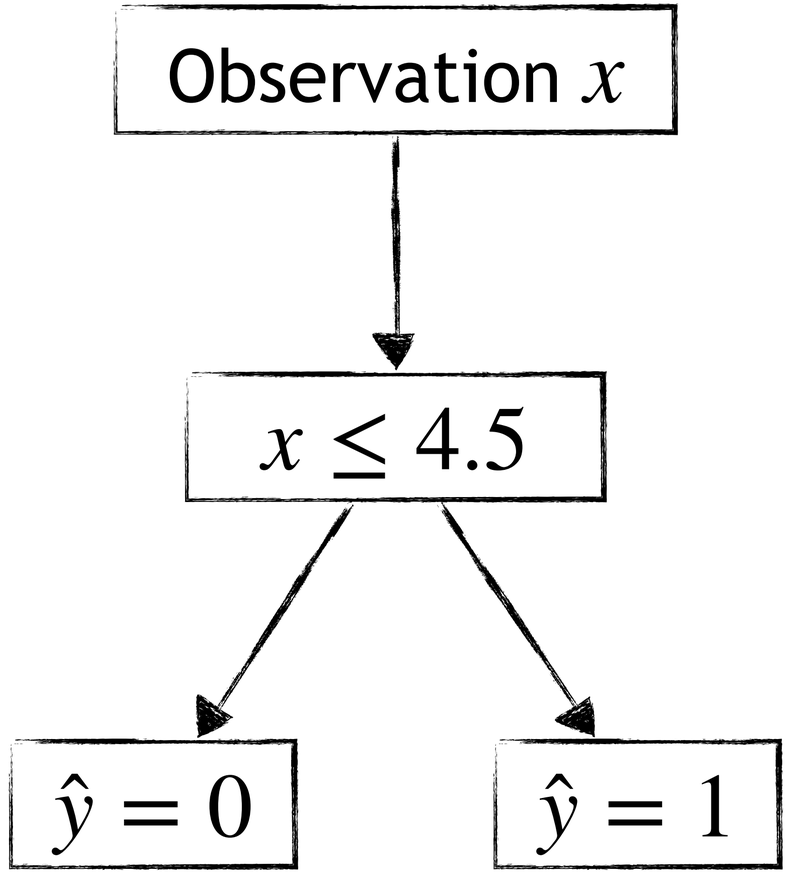


Decision tree
- How does it work ? -



Decision tree
- How does it work ? -





Decision tree
- And with more than one feature ? -

Issue : For now, we understand how a classification task is solved when we have one feature to describe one target variable. How does it works if we have more than one feature ?

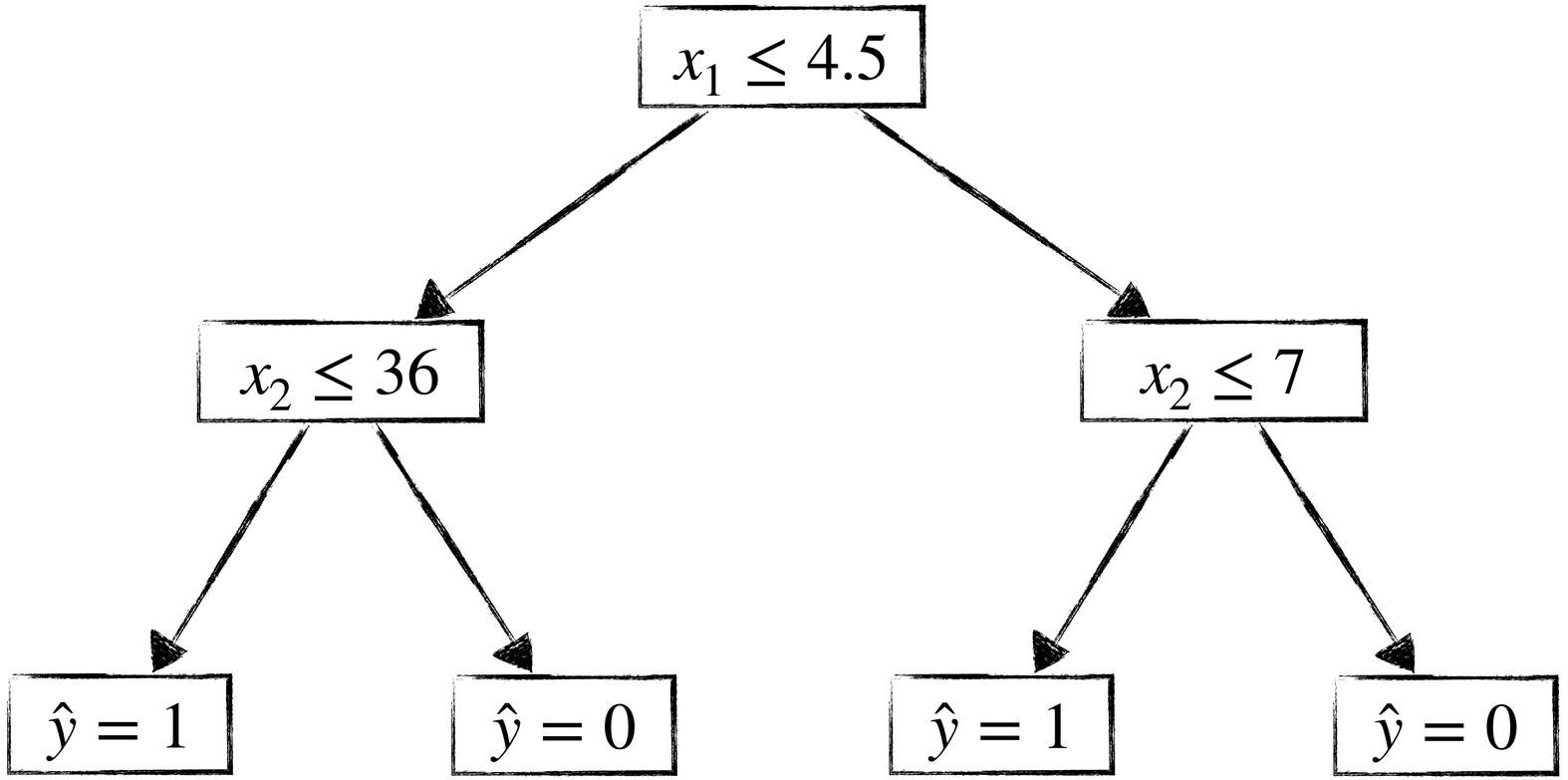
x1	1	2	3	4	5	6	7	8	9	10
x2	5	9	56	4	79	7	0	3	23	21
y	0	0	0	0	1	0	1	1	0	1

Decision tree
- And with more than one feature ? -

Solution : You do the same process as before on x_1 and x_2, and keep the split that have the least loss or variance or impurity.

x1	1	2	3	4	5	6	7	8	9	10
x2	5	9	56	4	79	7	0	3	23	21
y	0	0	0	0	1	0	1	1	0	1

Decision tree
- And with more than one feature ? -

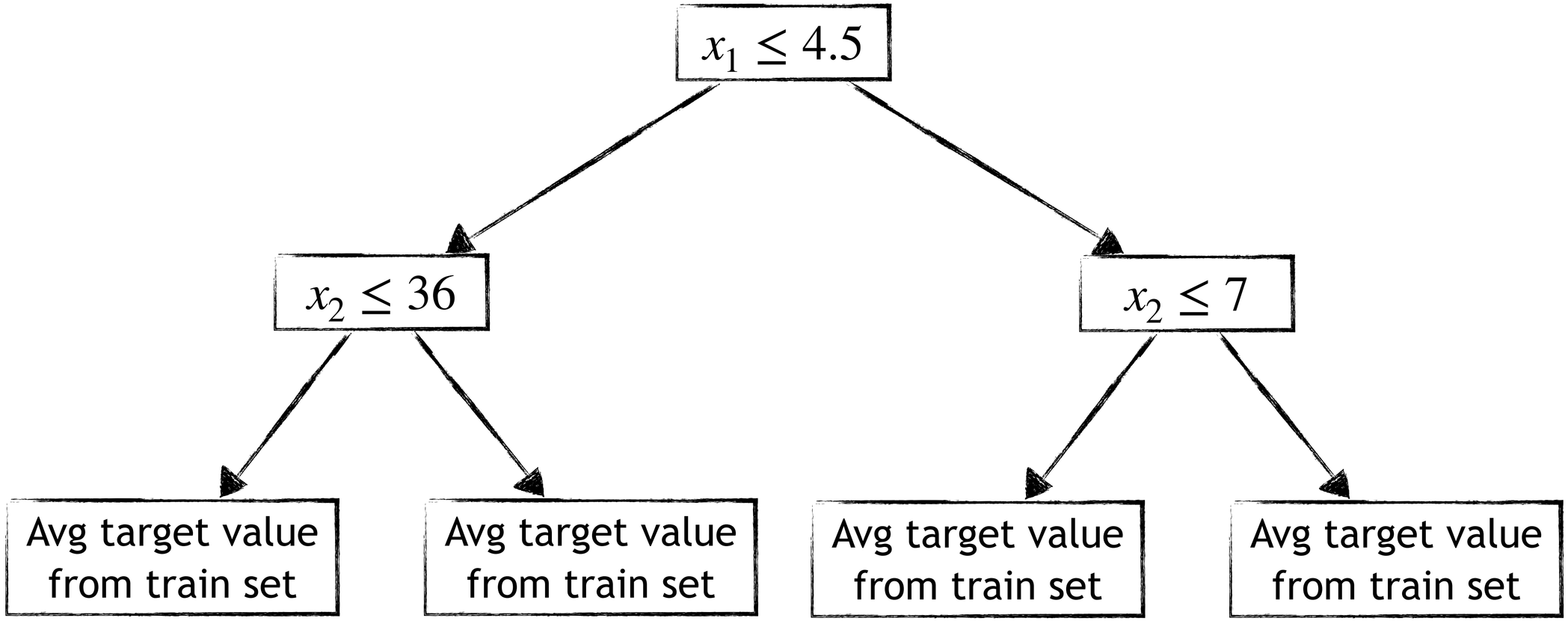


Decision tree
- And regression ? -

Issue : For classification task, we have the intuition behind. But what happens if we want to solve a regression task using random forests ?

=> You do exactly the same with the variance or the MSE and you take as a prediction the average target value given by the data observed in the train set.

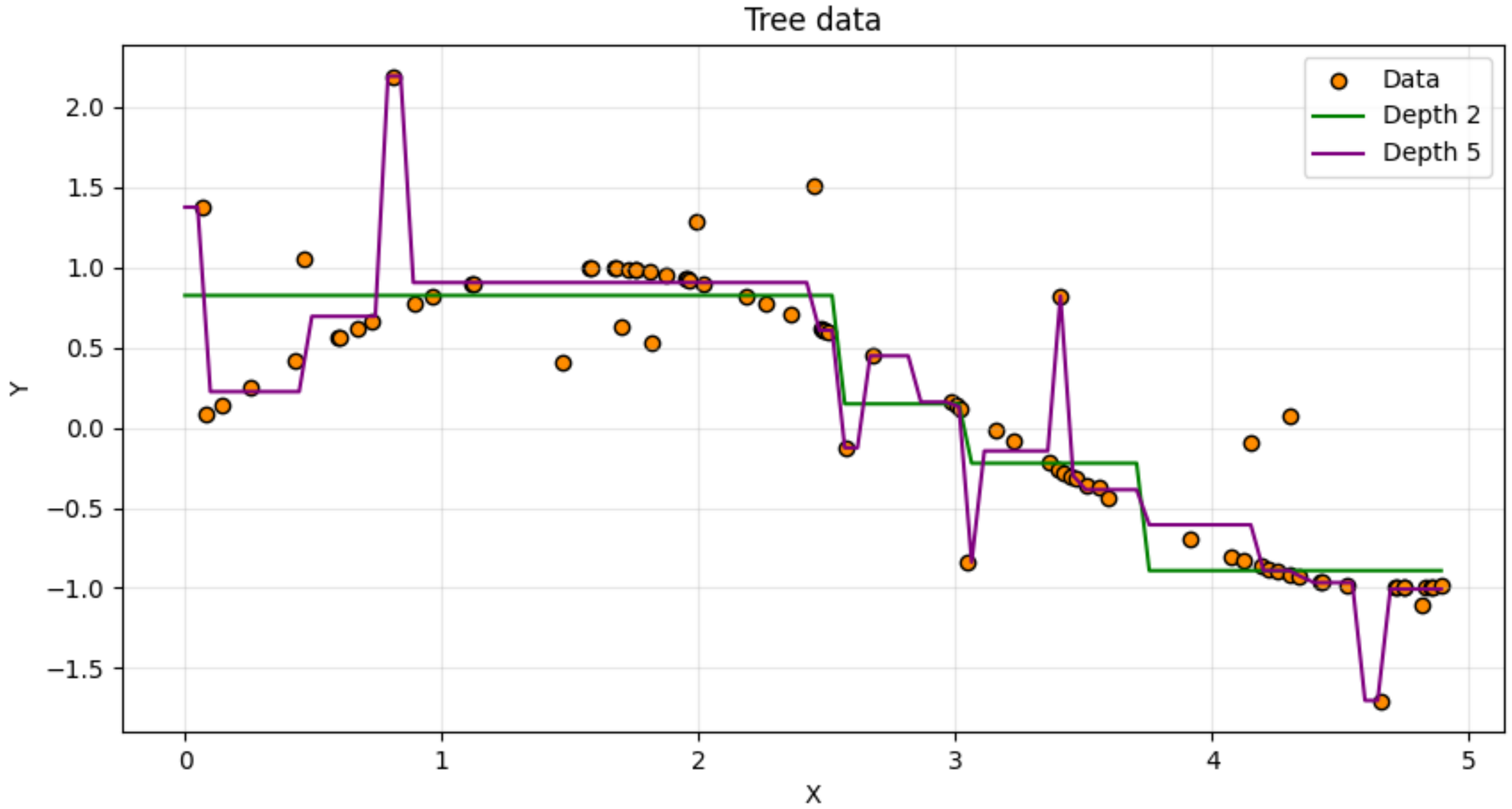
Decision tree
- And regression ? -



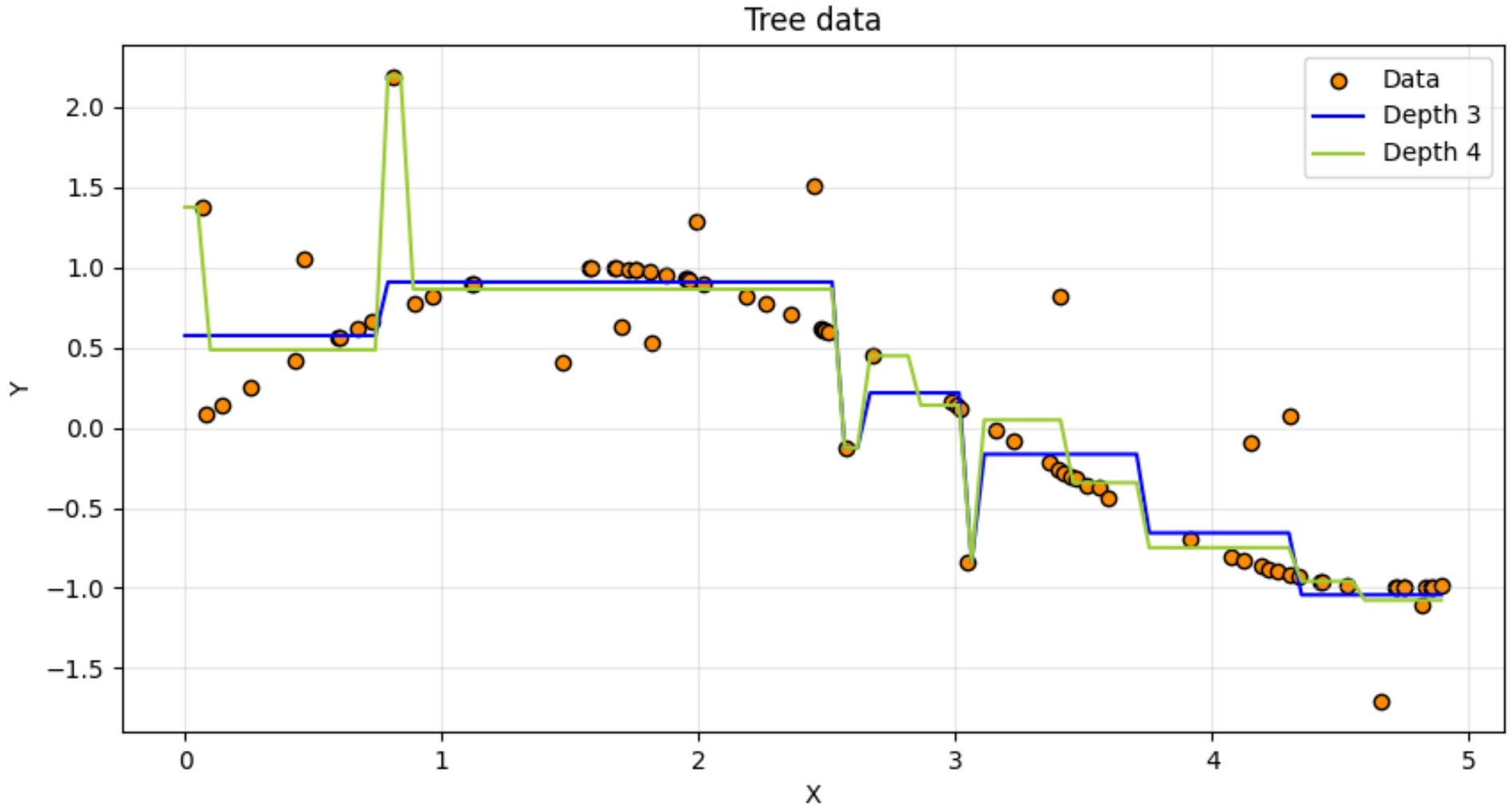
Decision tree
- Practice -

Exercice : Using the data generated by the script `generator.py`, train a DecisionTreeRegressor with different values of max depth going from 2 to 5.

Decision tree
- Practice -



Decision tree
- Practice -

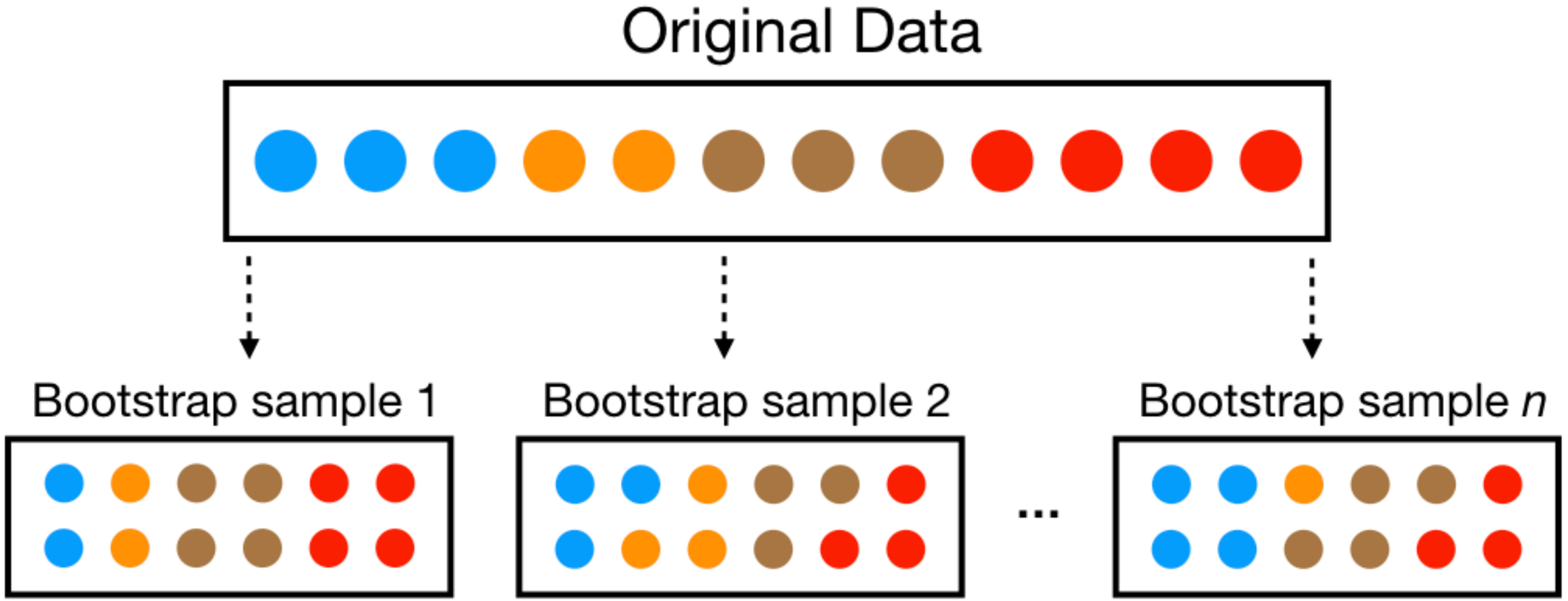


Random forest

Problem : We face a critical issue of overfitting with the decision tree regarding the depth of the model. If we face a lack of generalisation using decision trees, we have to use random forest.

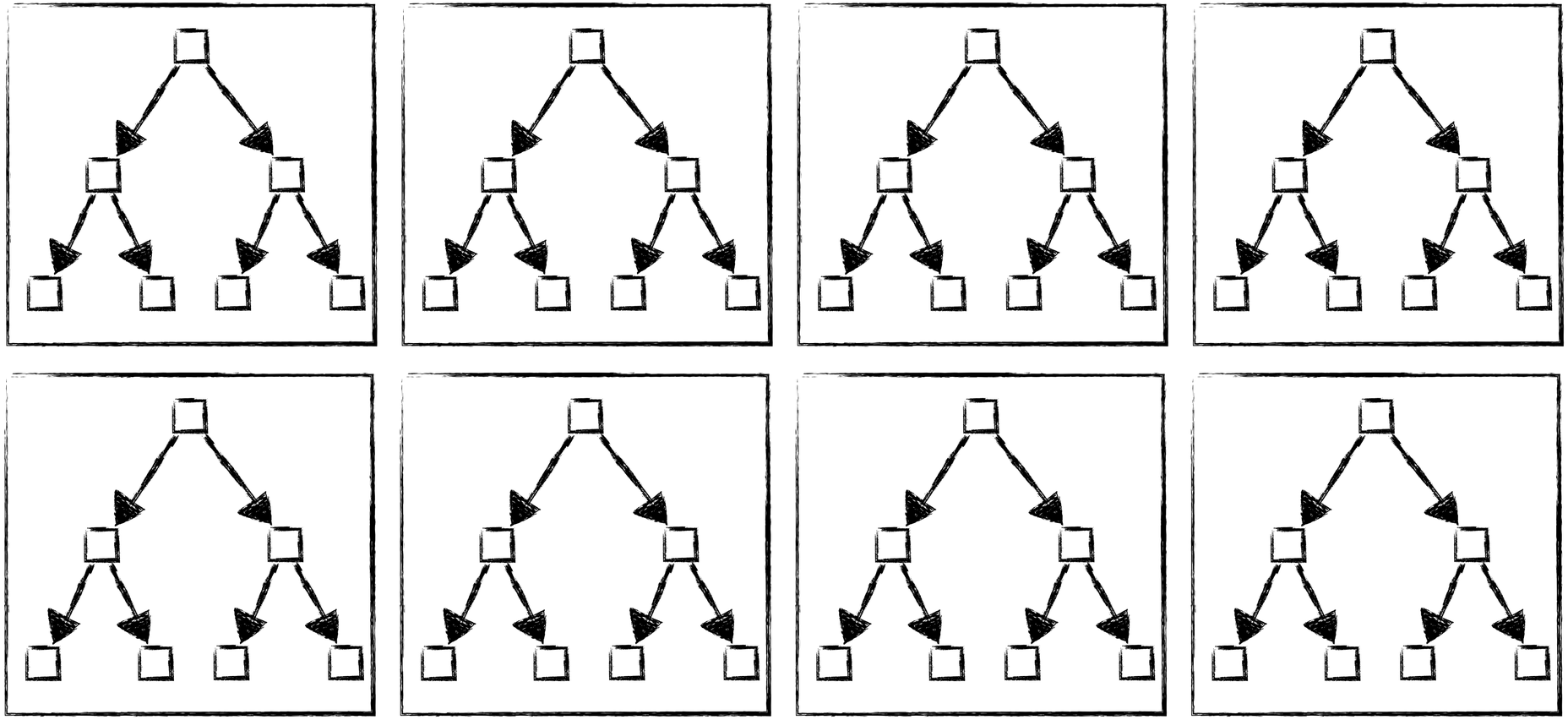
Defintion : A random forest is a set of decision trees, each trained on a random bootstrap sample from train set and a random subset of features.

Bootstrap sample
- Random sampling with replacement -

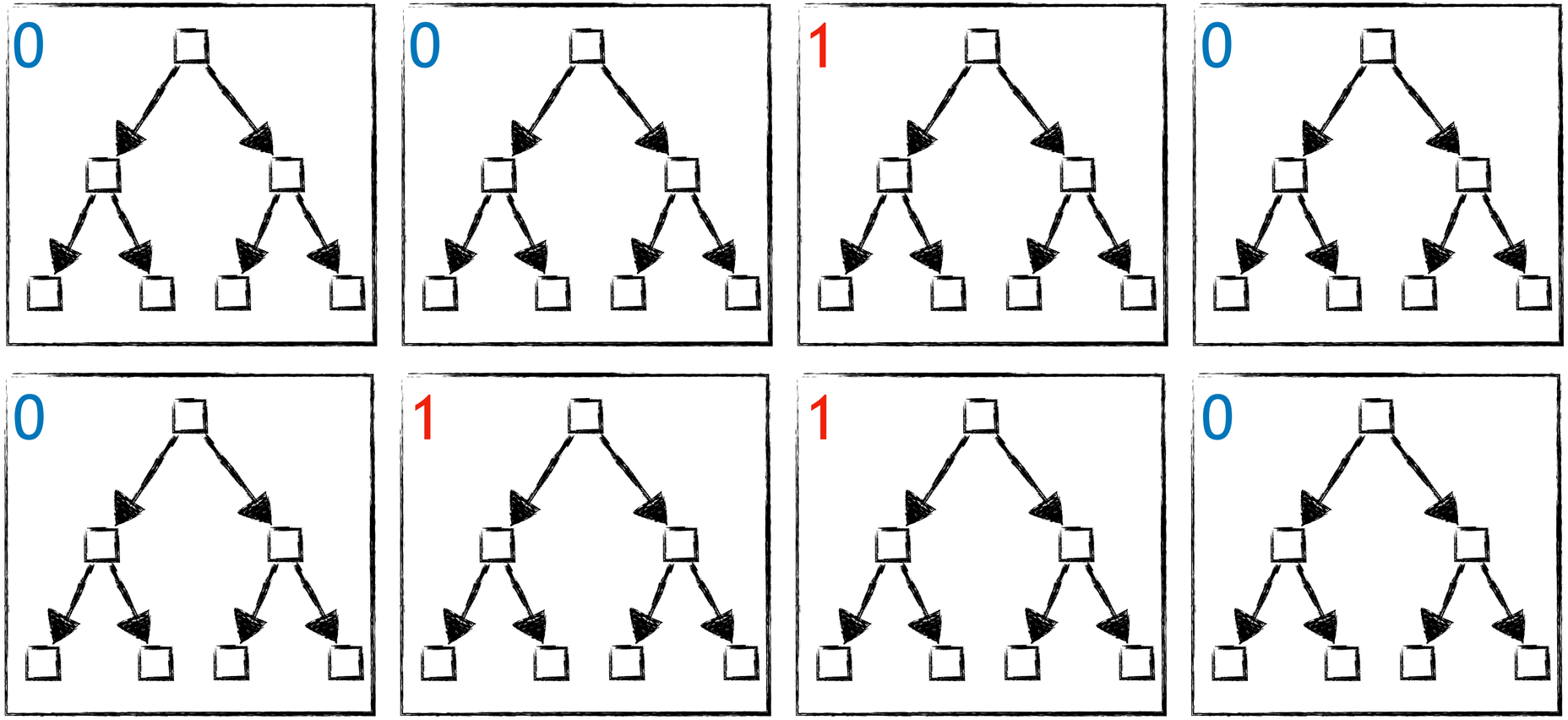


Random forest
- How does it work ? -

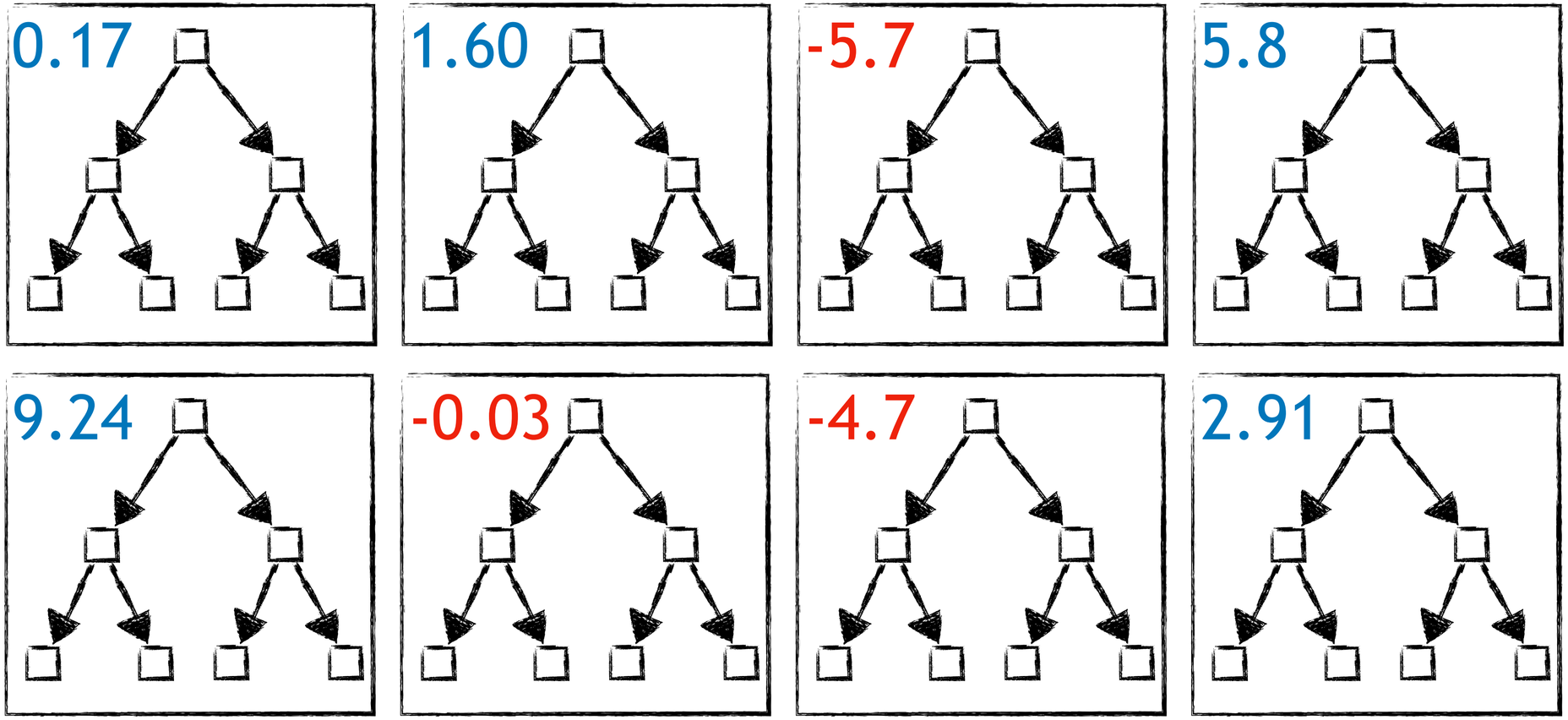
All trained in parallel



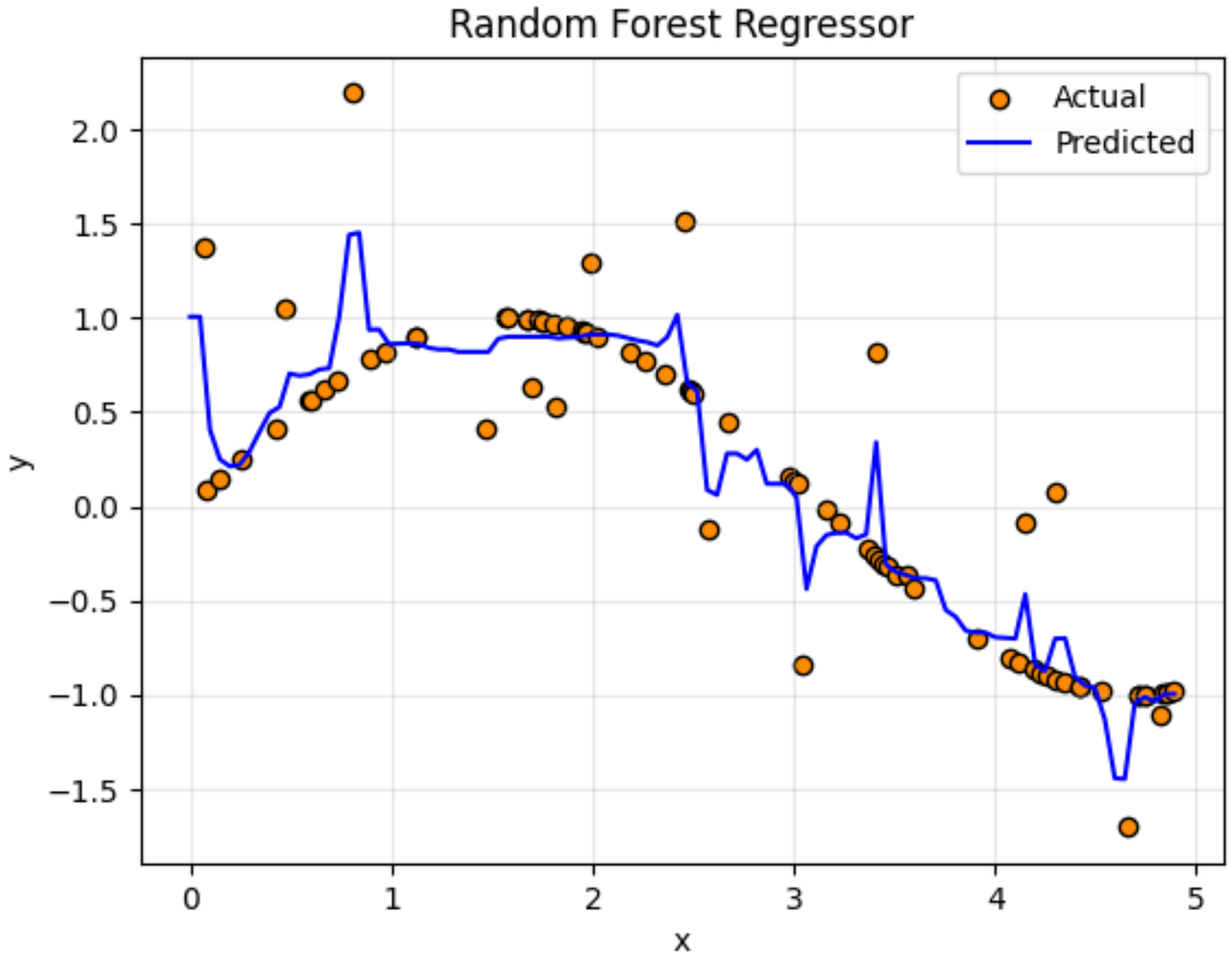
Random forest
- How does it work ? -



Random forest
- How does it work ? -



Random forest
- Example -



Random forest
- Practice -

Exercice : Using you favourite dataset load_diabetes from sklearn, train and test a random forest model (don’t forget to validate the hyper parameters - free parameters for your model).

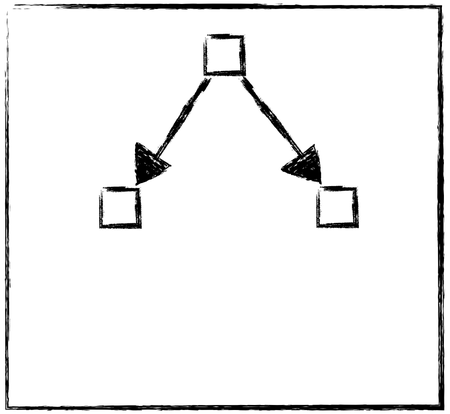
- Monday : Metrics and training process
- Tuesday : Linear models
- Wednesday : Support Vector Machine models
- Thursday : Decision trees
- Friday : Time series analysis & opening subject
  - XGBoost
  - Time series analysis
  - Semi-supervised & Reinforcement learning

XGBoost

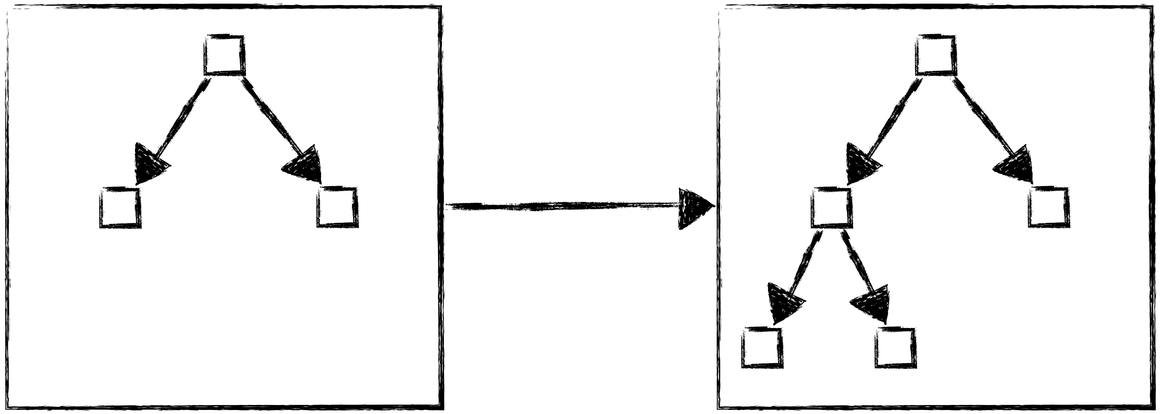
Problem : Still the same issue encountered with decision trees, but instead of training multiple independent trees, we « re-train » each time a new tree from the results of the previous ones.

Defintion : XGBoost is a set of decision trees, each trained sequentially correcting the errors of its predecessor trees.

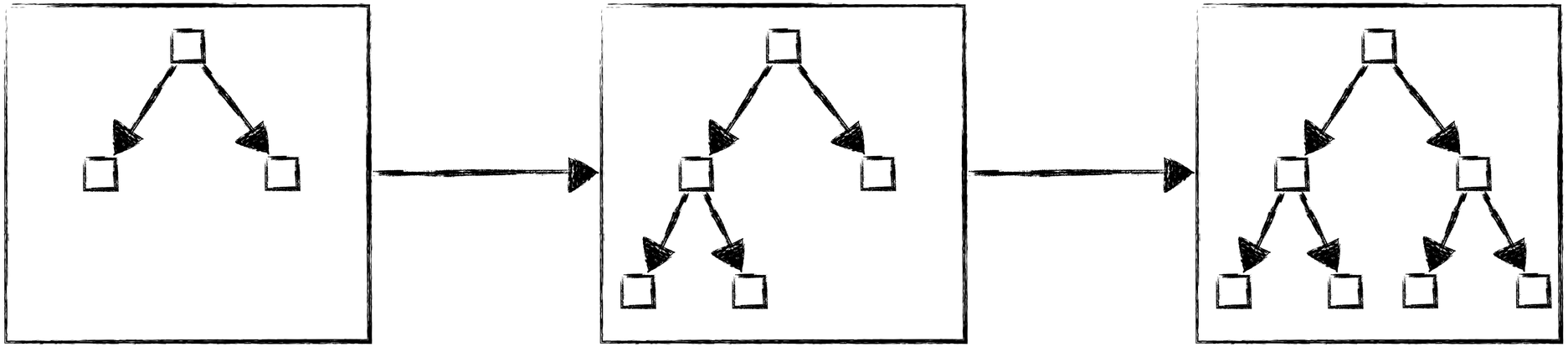
XGBoost
- How does it work ? -



XGBoost
- How does it work ? -

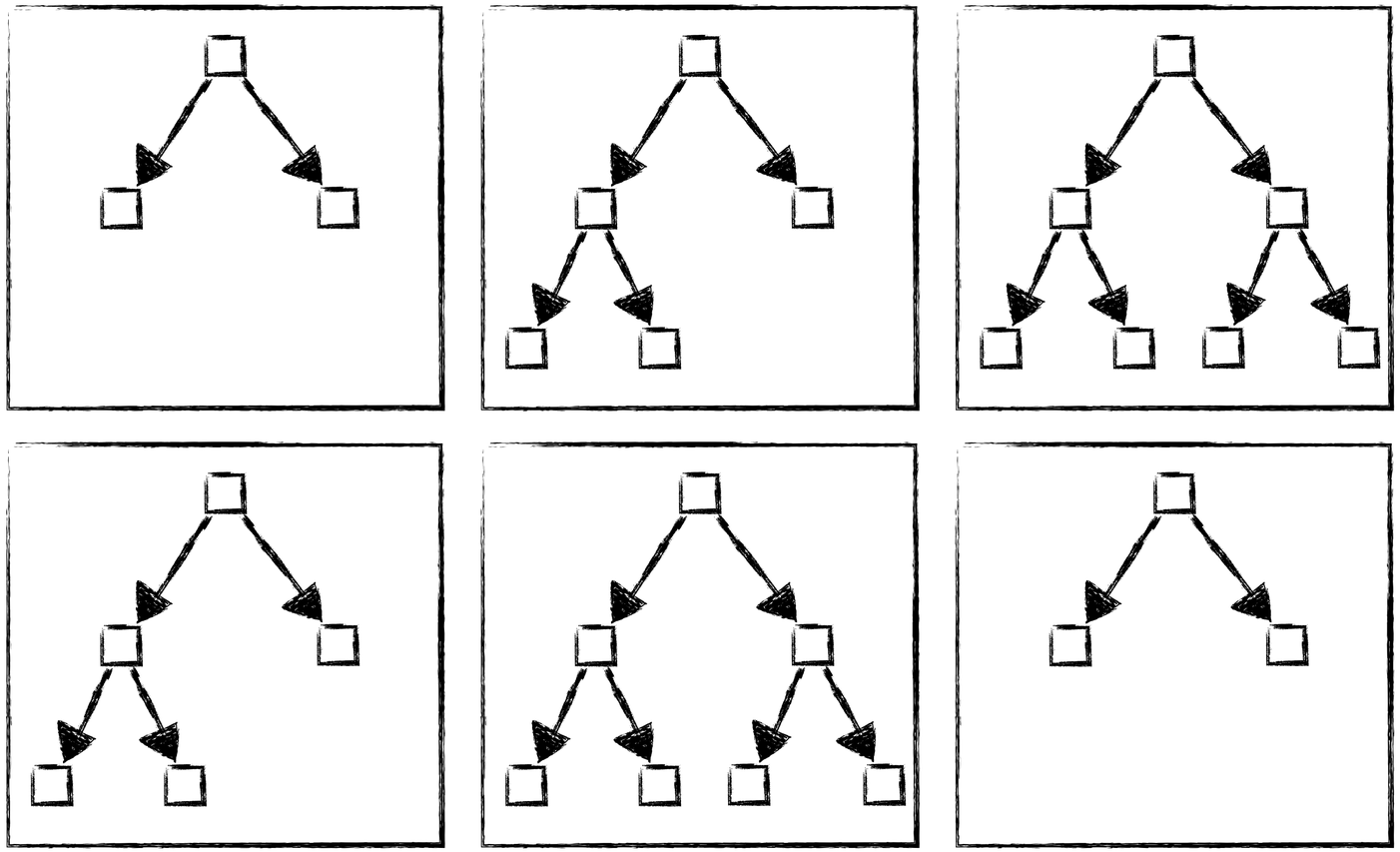


XGBoost
- How does it work ? -

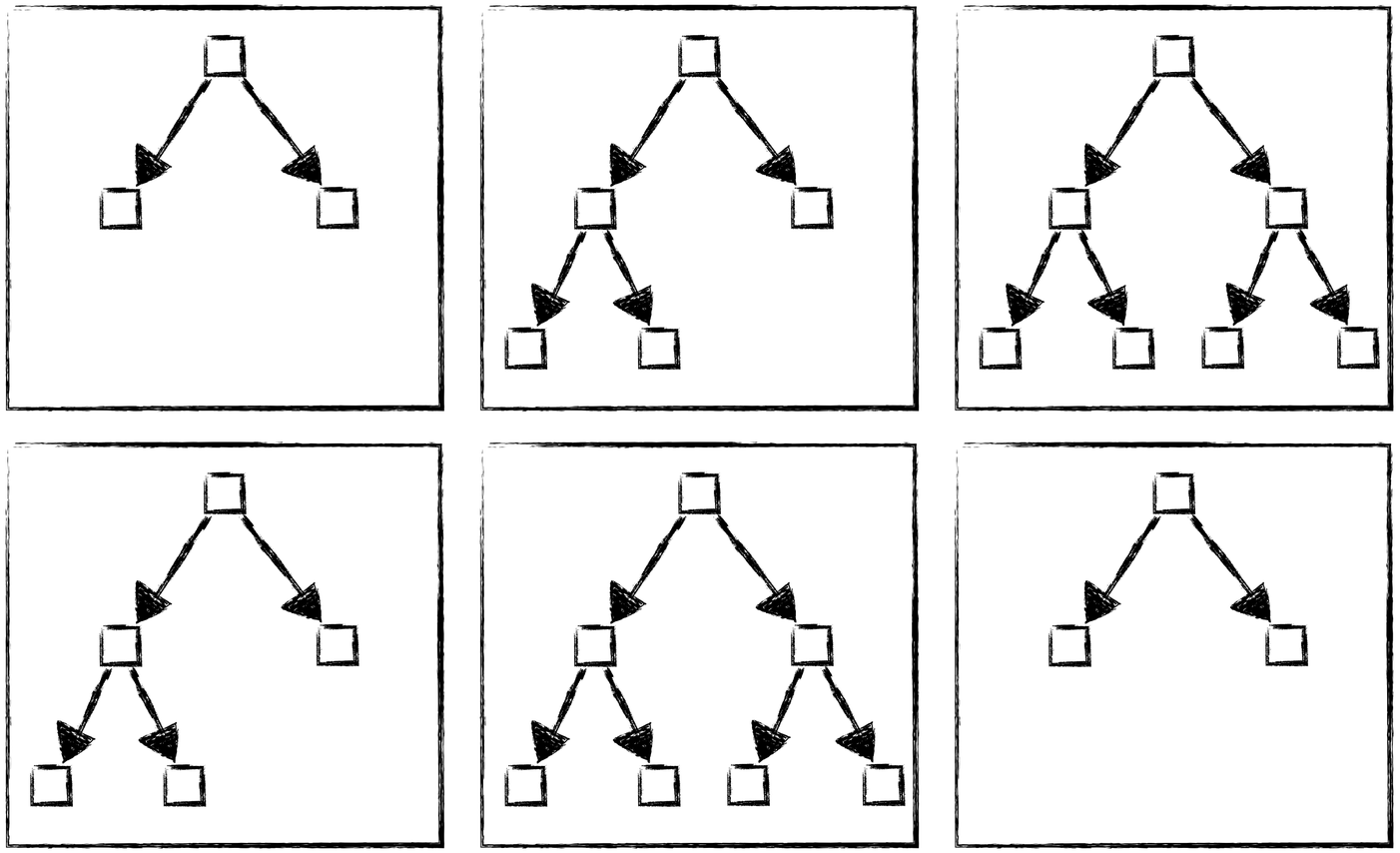


XGBoost
- How does it work ? -

NOT trained in parallel



How do you get a prediction with XGBoost ?



Regression : You still take the average of the target variables seen in train set.

Classification : You don’t take the votes (trees are not independent).
You take the sum of log-odds.

XGBoost - Hyperparametrization

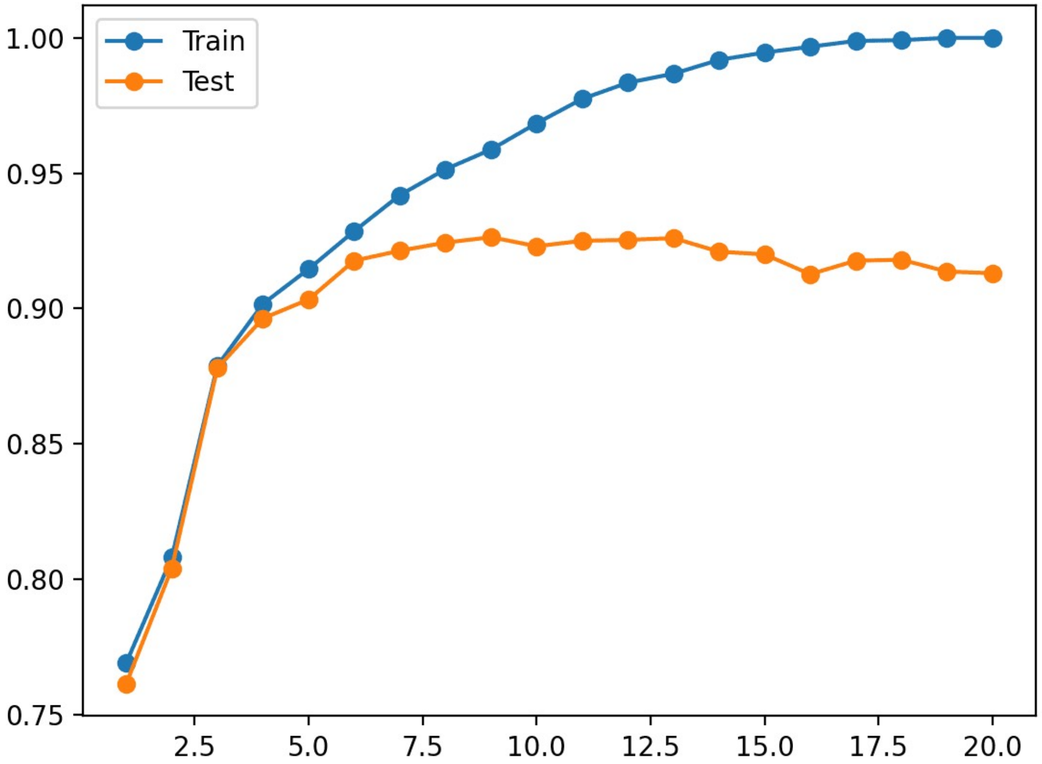
- *learning_rate :* it controls how fast the model learns
- *n_estimators :* number of trees
- *max_depth :* maximum depth for each tree
- *subsample :* fraction of training samples used per tree
- *colsample_bytree :* fraction of features used by tree
- *gamma :* minimum loss reduction to make a split
- *reg_alpha / reg_lambda :* regularisation terms (for L1 and L2).

XGBoost
- Practice -

Exercice : Using you favourite dataset load_diabetes from sklearn, train and test a XGBoost model (don’t forget to validate the hyper parameters - free parameters for your model).

One last thing about cross-validation

How do you observe overfitting ?



Conclusion on Tree-based models

- ***Decision trees :*** Fast training and inference, interpretable and prone overfitting.
- ***Random forests :*** Better generalisation and less overfitting.
- ***XGBoost :*** Strong performances, handle complex patterns and requires careful tuning.

Time series analysis

Definition : A time series is a 1-dimensional feature depending on time (can be year, month, day, hour, min, sec, etc…).

You may have multiple time series, then depending of the type of data and the relation they have each other, you may have to synchronise them, use downsampling / upsampling methods, etc…

Time series analysis

Idea : Either you have a regression or a classification task, you want to get a prediction from time series feature(s).

Mainly, the models stay the same as before. But the number of feature is not clear for the models.

=> Need to define new features.

Time series analysis
- Working example -

Example : You want to predict the height of waves for a specific surf spot regarding the meteorological time series features.

You have then access then to some features at the time you want to predict, but you know there is a relationship between the previous meteorological data from previous days and the current height of waves.

=> You want to predict for next morning what will be the height of waves.

Time series
- Features extractor -

1 - Lag-features : The idea is to use previous data from one or multiple features on specific time $(x_{t-1}, x_{t-2}, x_{t-3}, \ldots)$.

2 - Rolling statistics : Use of moving average, standard deviation or median values over a time window.

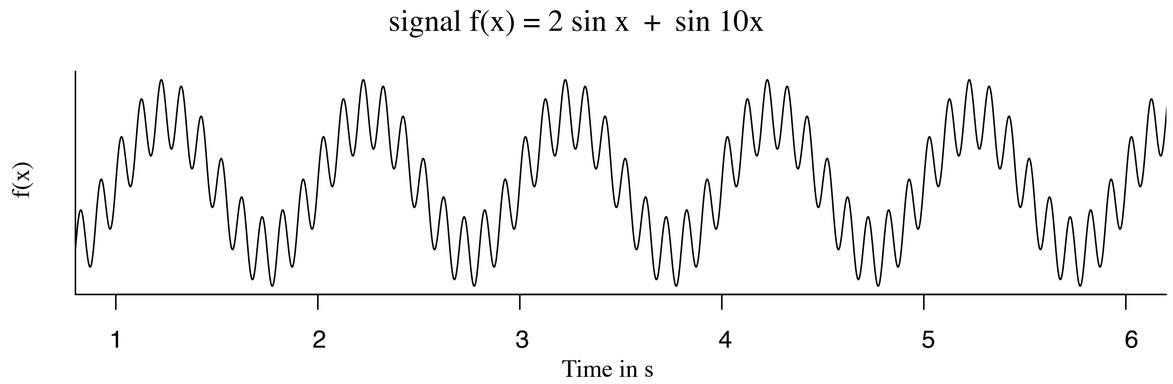
Time series
- Features extractor -

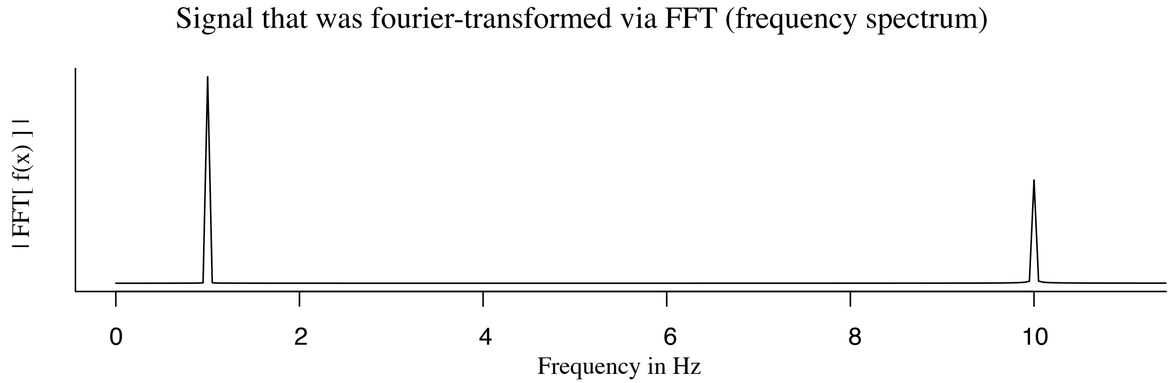
3 - Time based feature : Generally, you don’t use time in your features but for time series classification or regression tasks, you may need to use them as features.

4 - Trend / Cycle decomposition : Smooth measure of the underlying direction (trend) or measure of repeating patterns (e.g. daily, weekly or yearly).

Time series
- Features extractor -

5 - Fourrier transform (FFT) : Apply the Fast Fourrier Transform to the time series on a time window. New features may be the extracted dominant frequencies (periods) and their magnitudes.





Time series
- Features extractor -

6 - Autocorrelation / Partial autocorrelation : Given a time series $(x_1, x_2, \ldots, x_t)$, the autocorrelation at lag k is:

$$\rho_k = \frac{\text{Cov}(x_t, x_{t-1})}{\sigma^2},$$

- $\text{Cov}(x_t, x_{t-1})$: covariance between current and lagged value
- σ^2 : variance of the series

Time series
- Features extractor -

7 - Peak / Valley counts, zero crossing, etc… : It captures the shape and oscillation of the signal / time series.

- **Peaks/Valleys** : Local maxima / minima in the time series.
- **Zero crossings** : Number of times the series changes sign (crosses 0).
- **Slope sign changes** : Number of times the derivative of the signal changes sign.

Downsampling method

Idea : Reducing the time resolution of the time series. You may use different methods of aggregation as follow :

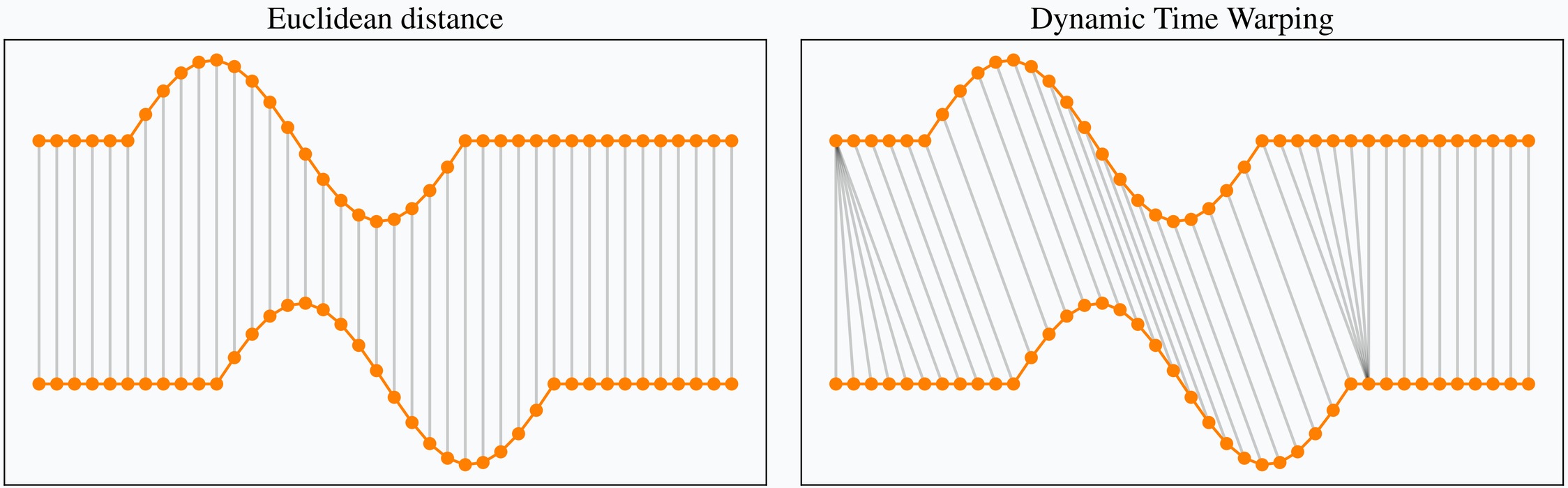
- Mean
- Sum
- Maximum
- Minimum
- Last value
- Count of values

Upsampling method

Idea : Increasing the time resolution of the time series. You create missing time stamps that must be filled in:

- Interpolation
- Backward fill
- Forward fill

Dynamic Time Wrapping



Supervised vs Unsupervised
- What else ? -

Semi-supervised learning : A learning approach that uses a small amount of labeled data and a large amount of unlabeled data.

Semi-supervised learning : A learning approach where an agent / model learns by interacting with an environment to maximise a « reward ». This is the reward function that is hard to define.