

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

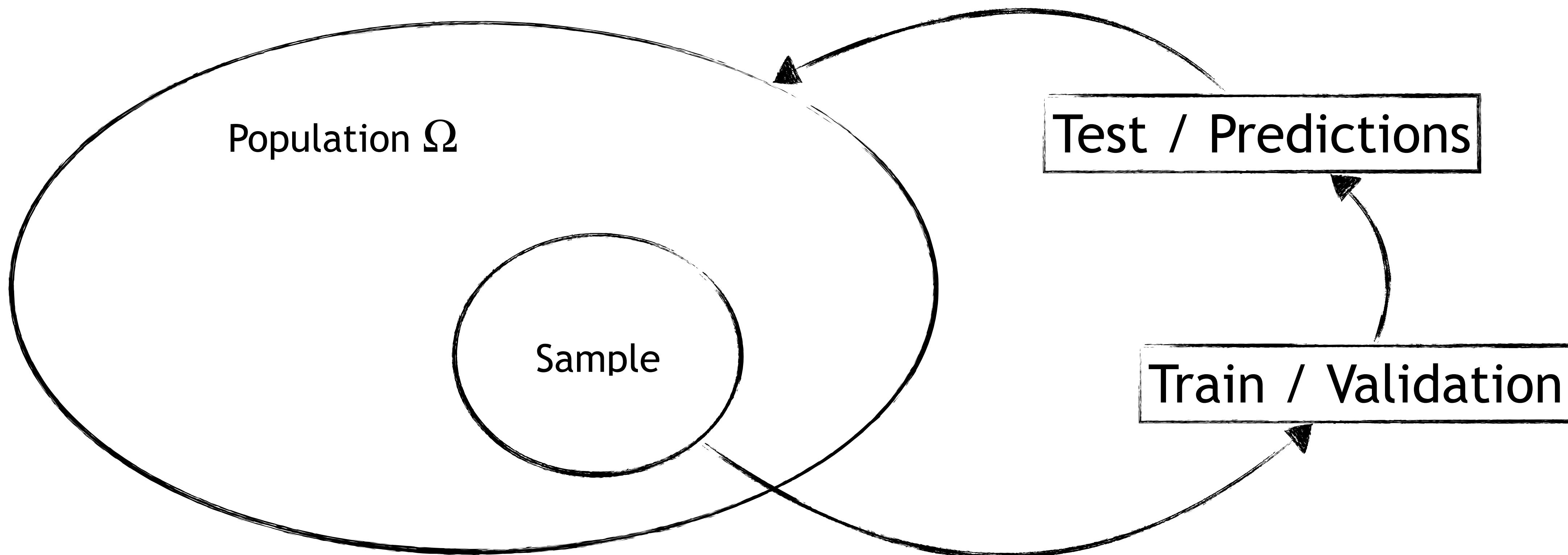
❖ Wednesday : Support Vector Machine models

❖ 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 learning 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.
- ▶ *Decision 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 your 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 choose 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.

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

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 than 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 Positive (TP)** : true value = 1, predicted value = 1
- **True Negative (TN)** : true value = 0, predicted value = 0
- **False Positive (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.

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

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.

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

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.

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

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

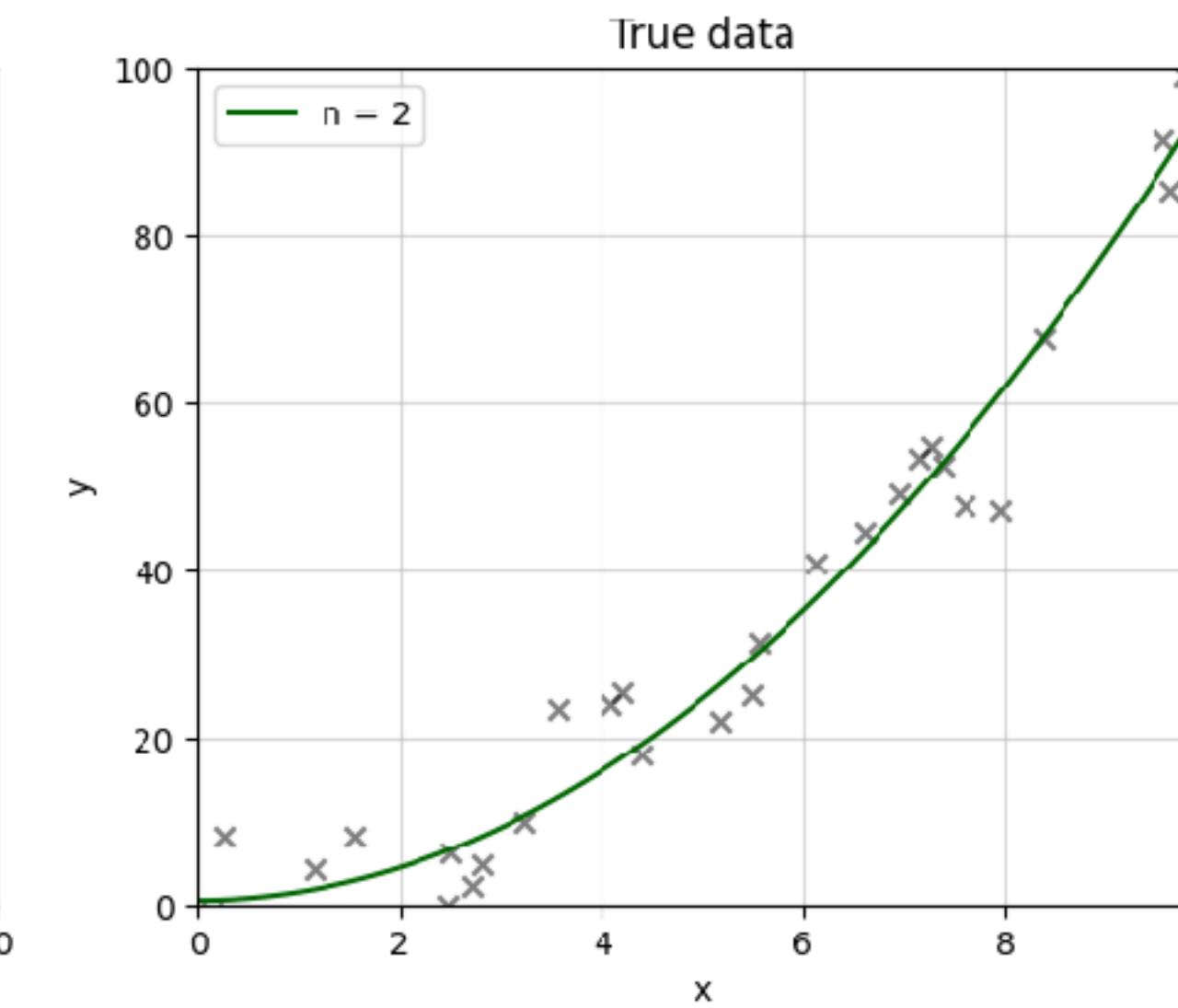
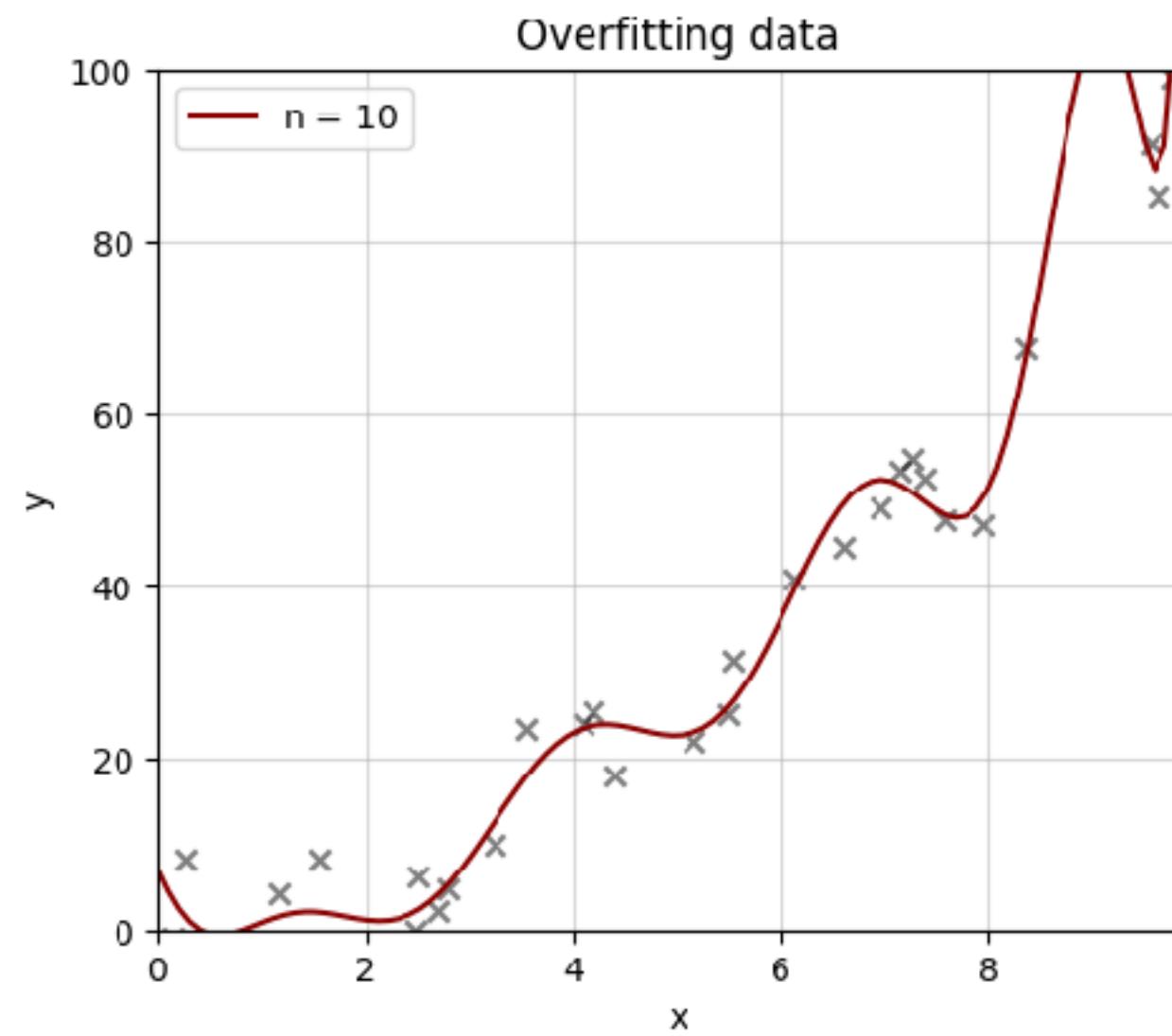
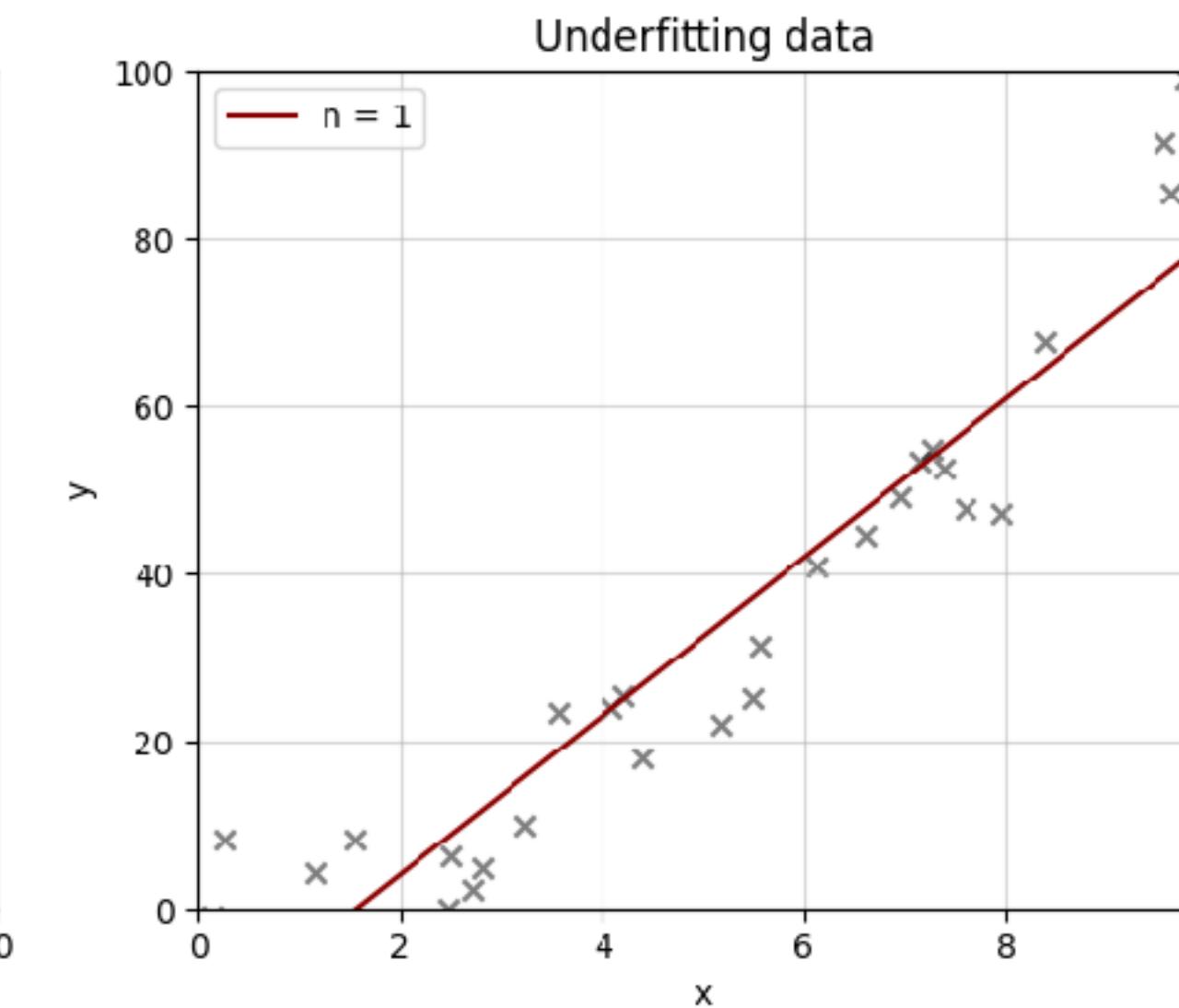
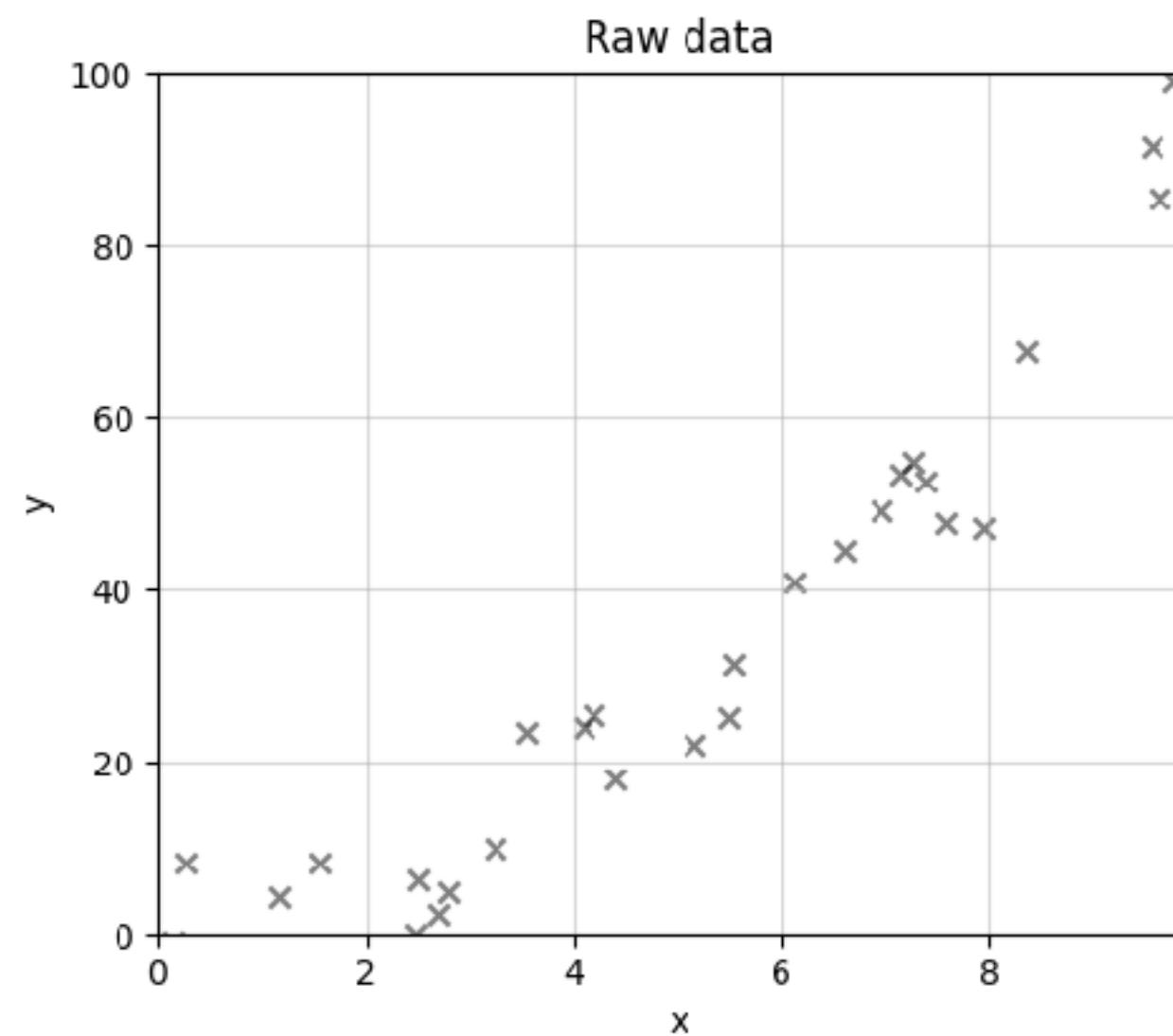
Definition : The model learns noise and details from the training data. It can be visualised by a good performance on training set but poorly on validation set.

=> 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** : Choose 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).

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

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:

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

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 at iteration n the following:

$$\begin{aligned}\beta_{n+1} &= \beta_n - \alpha \nabla_{\beta} \text{MSE} \\ &= \beta_n - \alpha \frac{2}{n} X^T (X\beta - y)\end{aligned}$$

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

Definition : 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

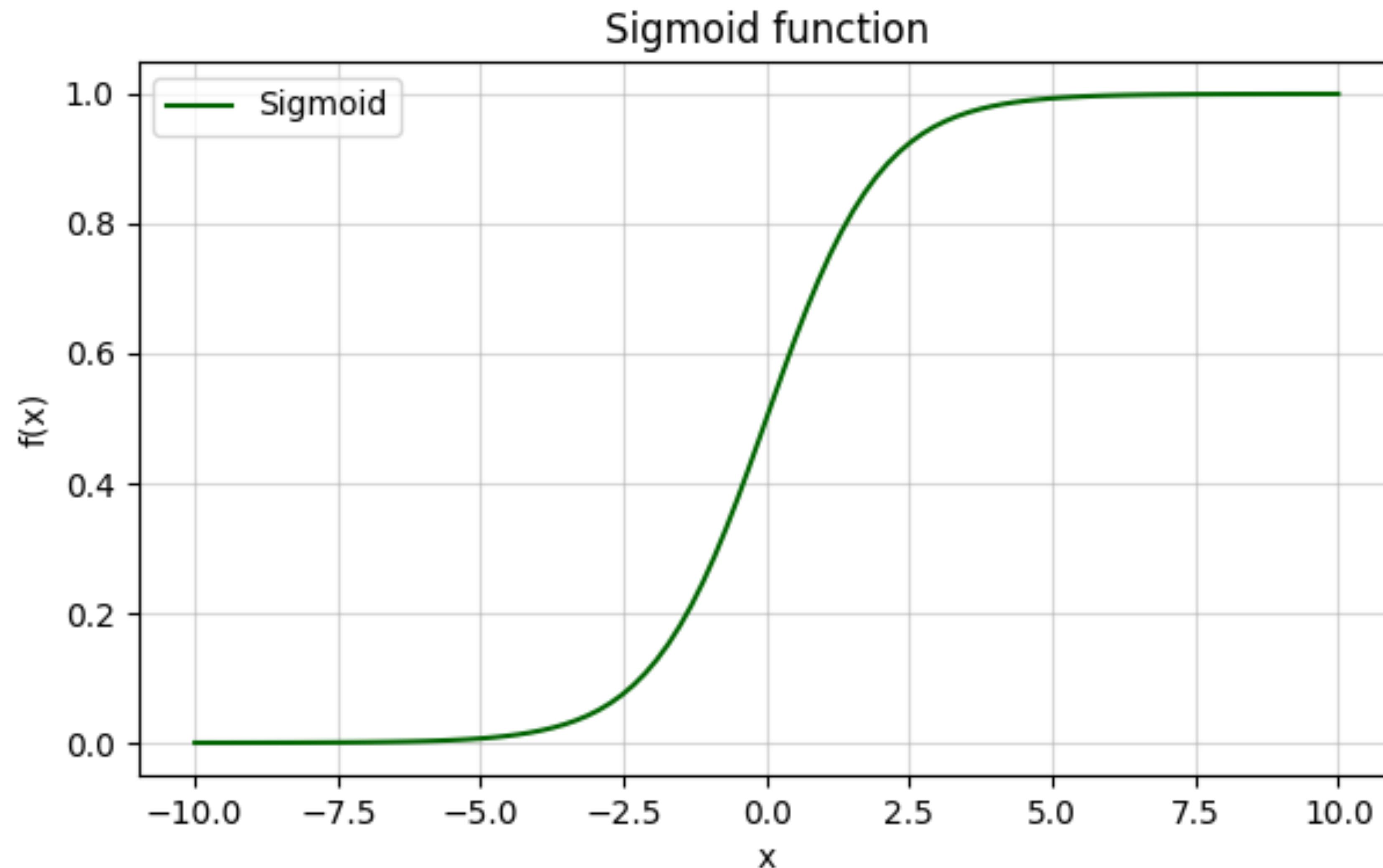
Definition : The logistic regression method is not for regression tasks but for classification tasks. The idea is to start from logistic function defined as:

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

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 :

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

Remark : Commonly used when face to binary classification.

Multinomial logistic regression

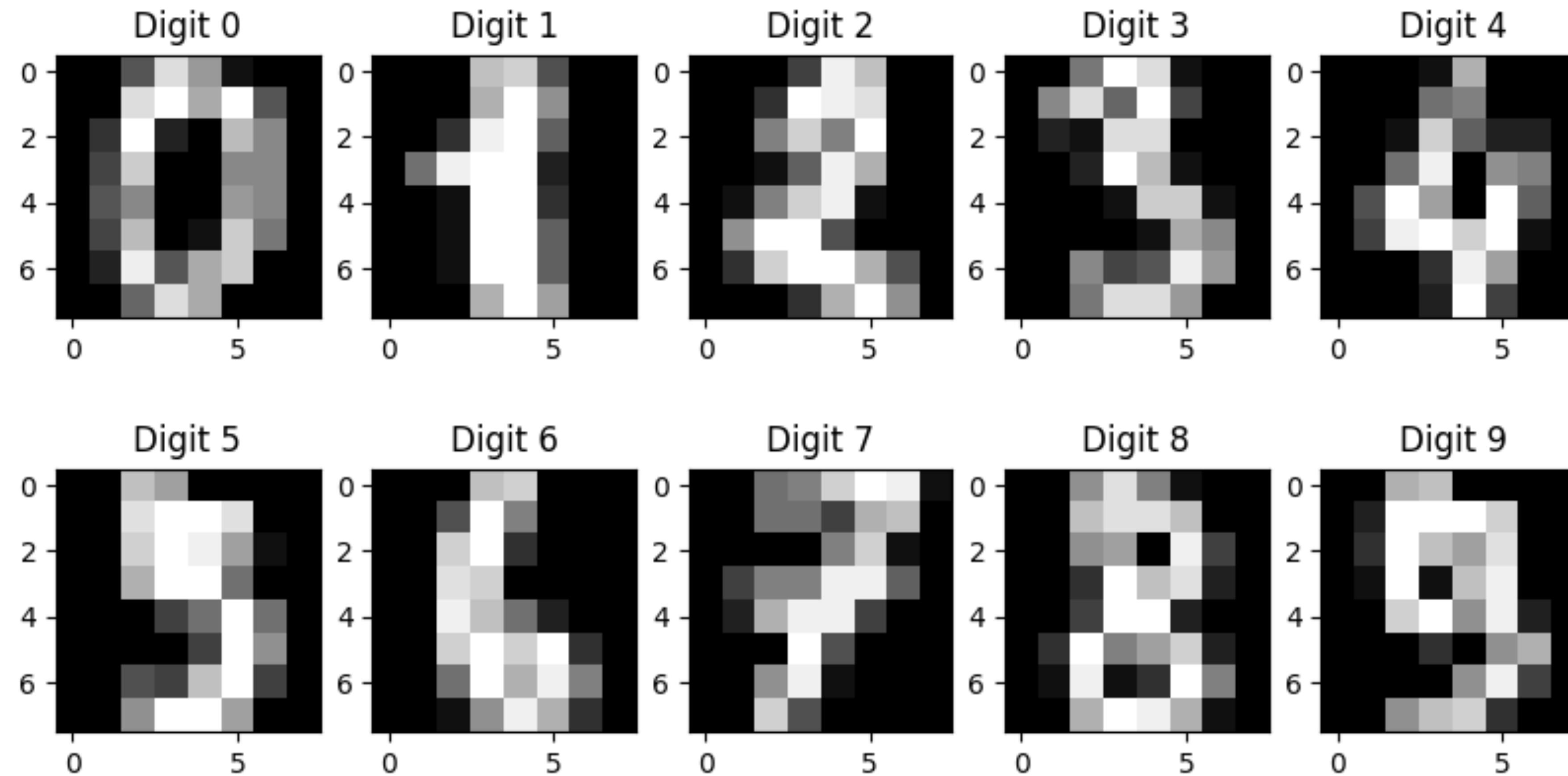
We follow the same idea as logistic regression but we are facing multi-class target instead of binary, i.e. $y \in \{0, 1, \dots, K\}$ and we note:

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

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

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

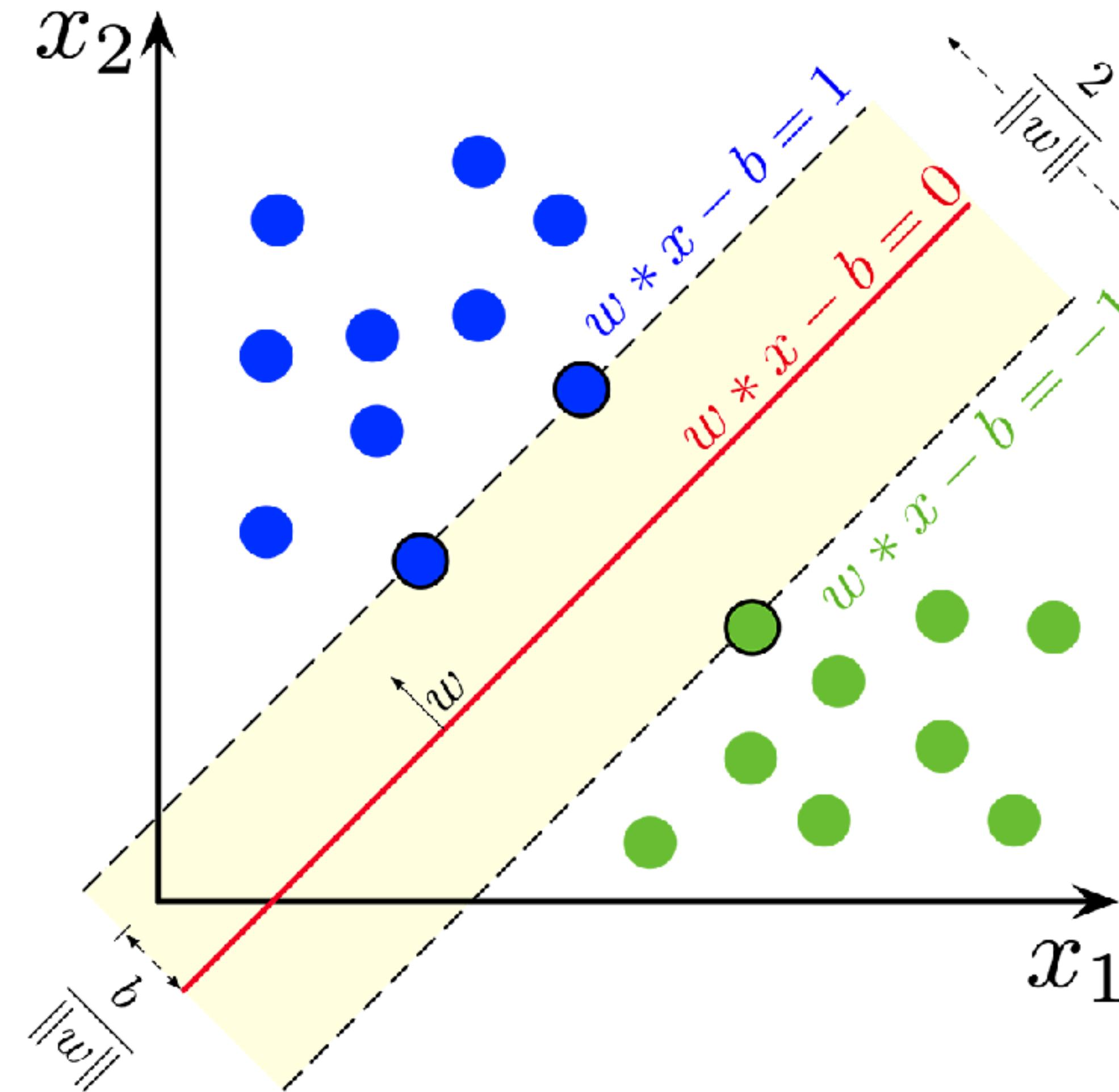
- SVM -

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

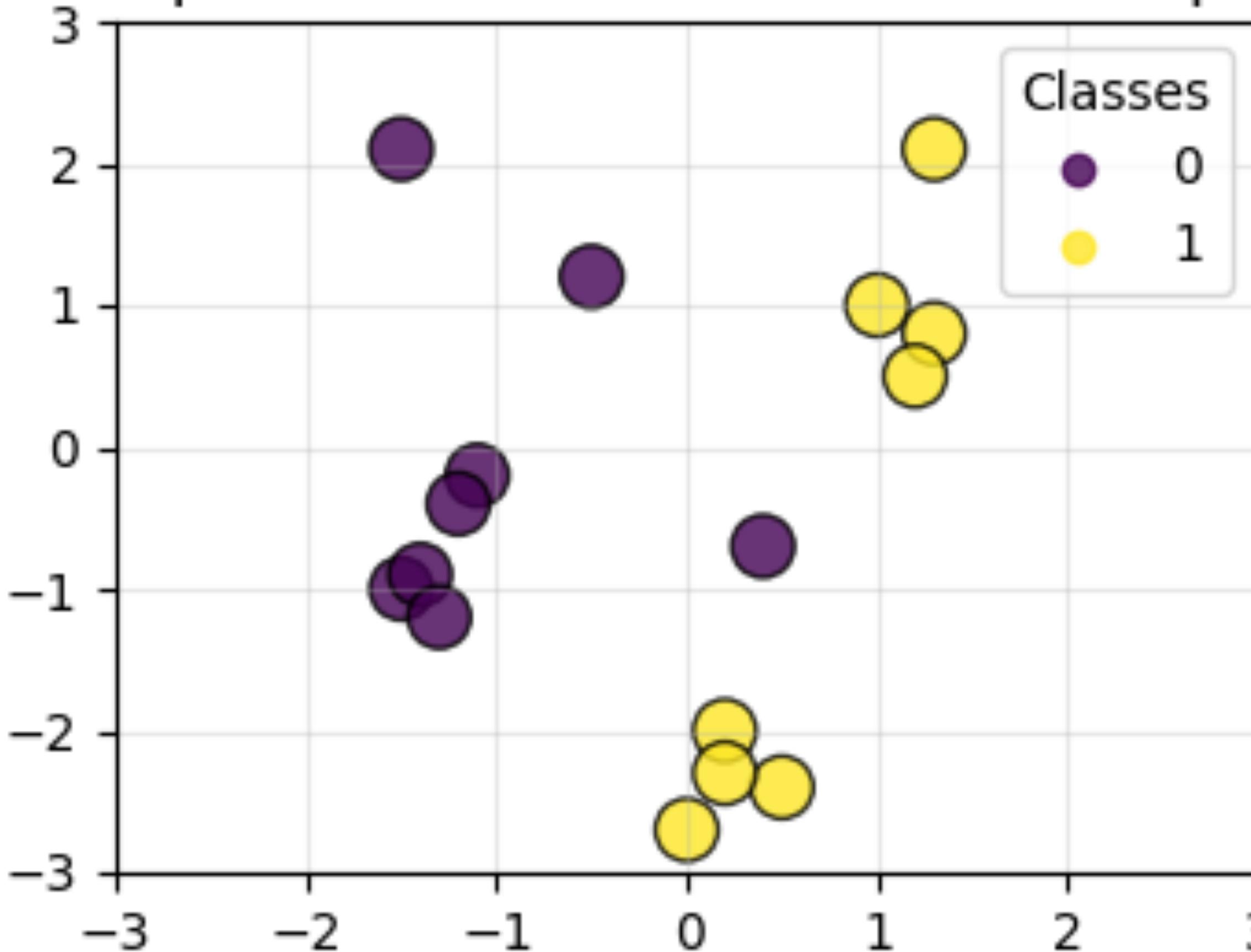
- SVM -



SVM

- Example -

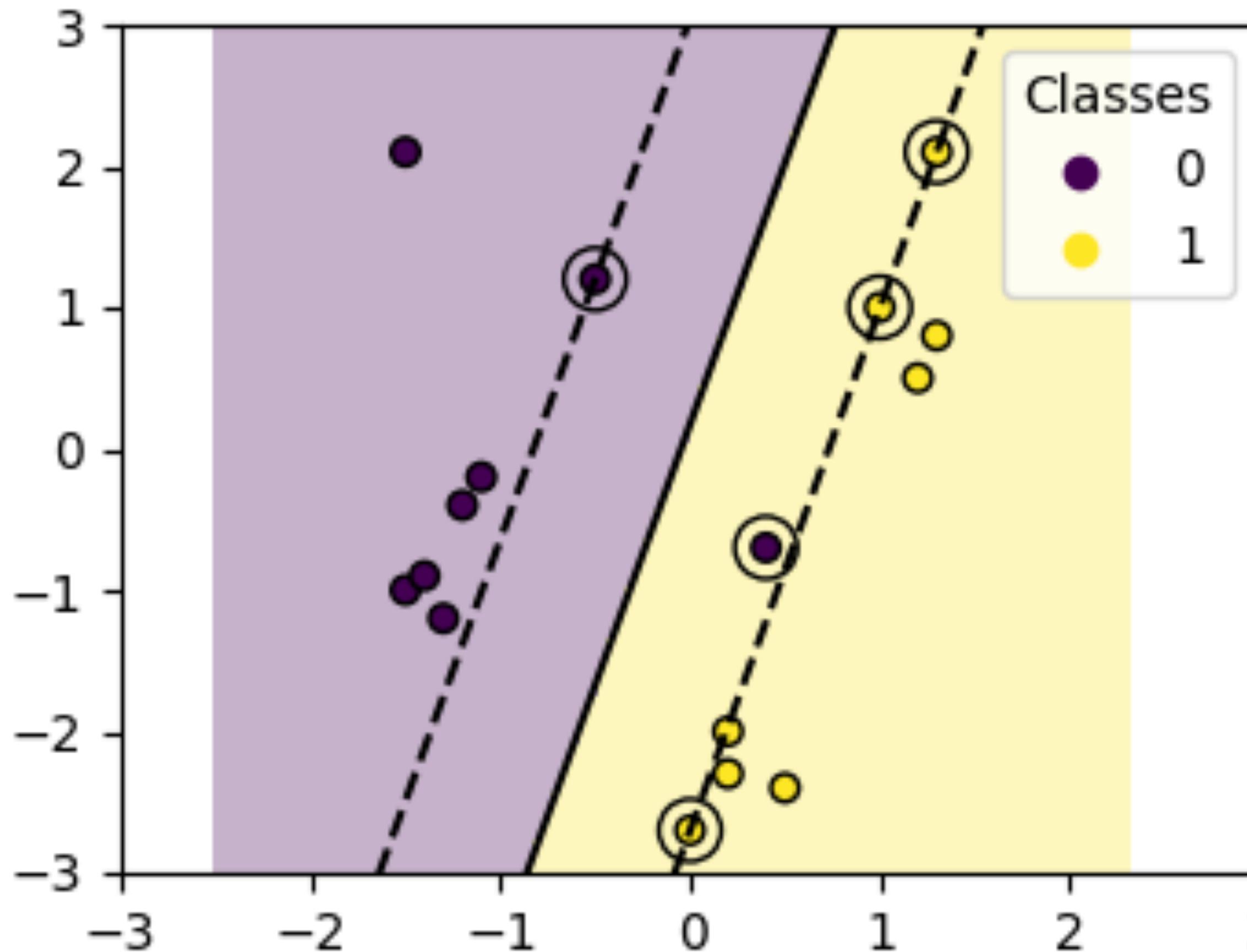
Samples in two-dimensional feature space



SVM

- Example -

Decision boundaries of linear kernel in SVC



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

Remark : Large value of $C \Rightarrow$ overfitting /vs/ low value of $C \Rightarrow$ underfitting.

SVM

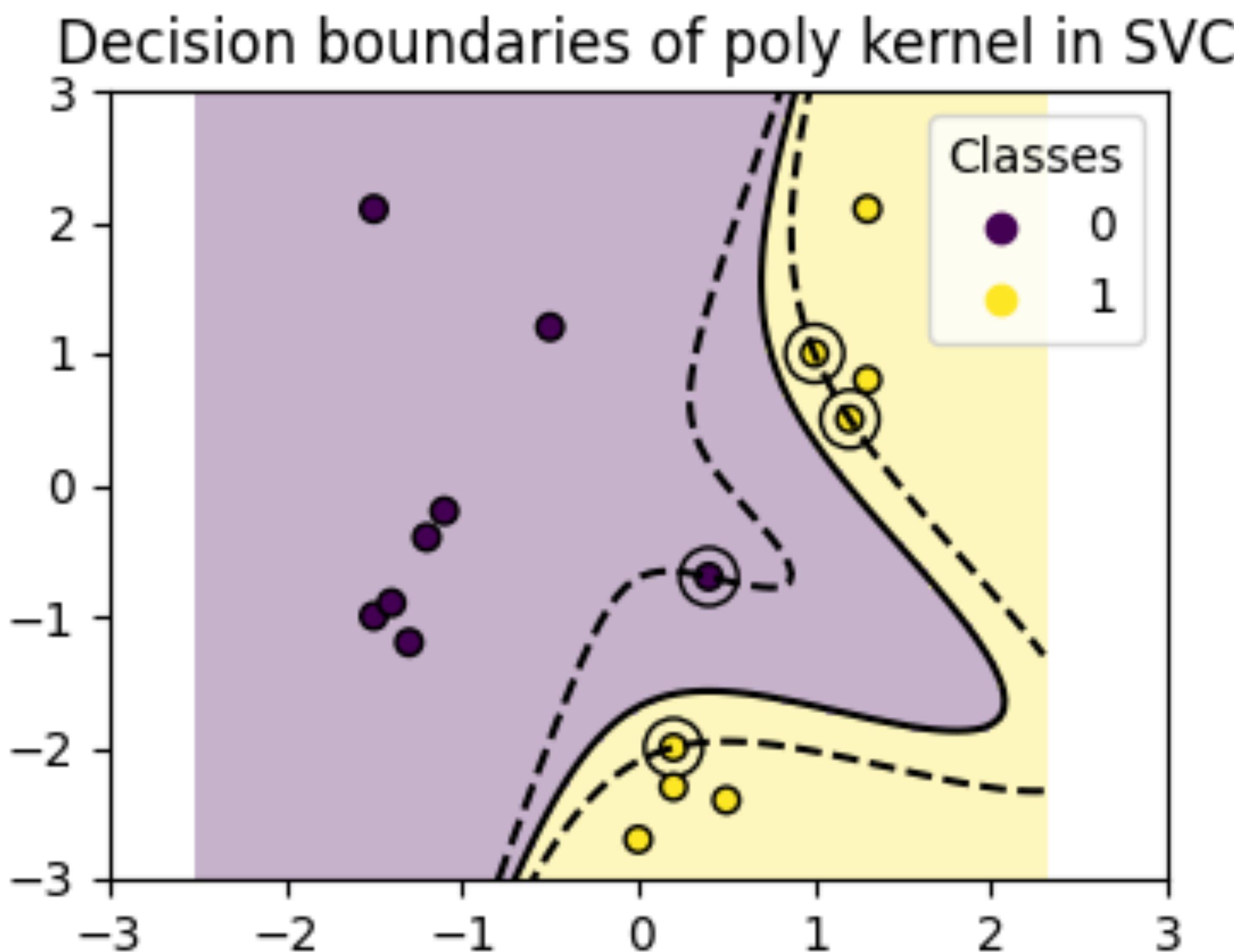
- Kernels -

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 : $K(x, x') = \tanh(\kappa x^T x' + c)$

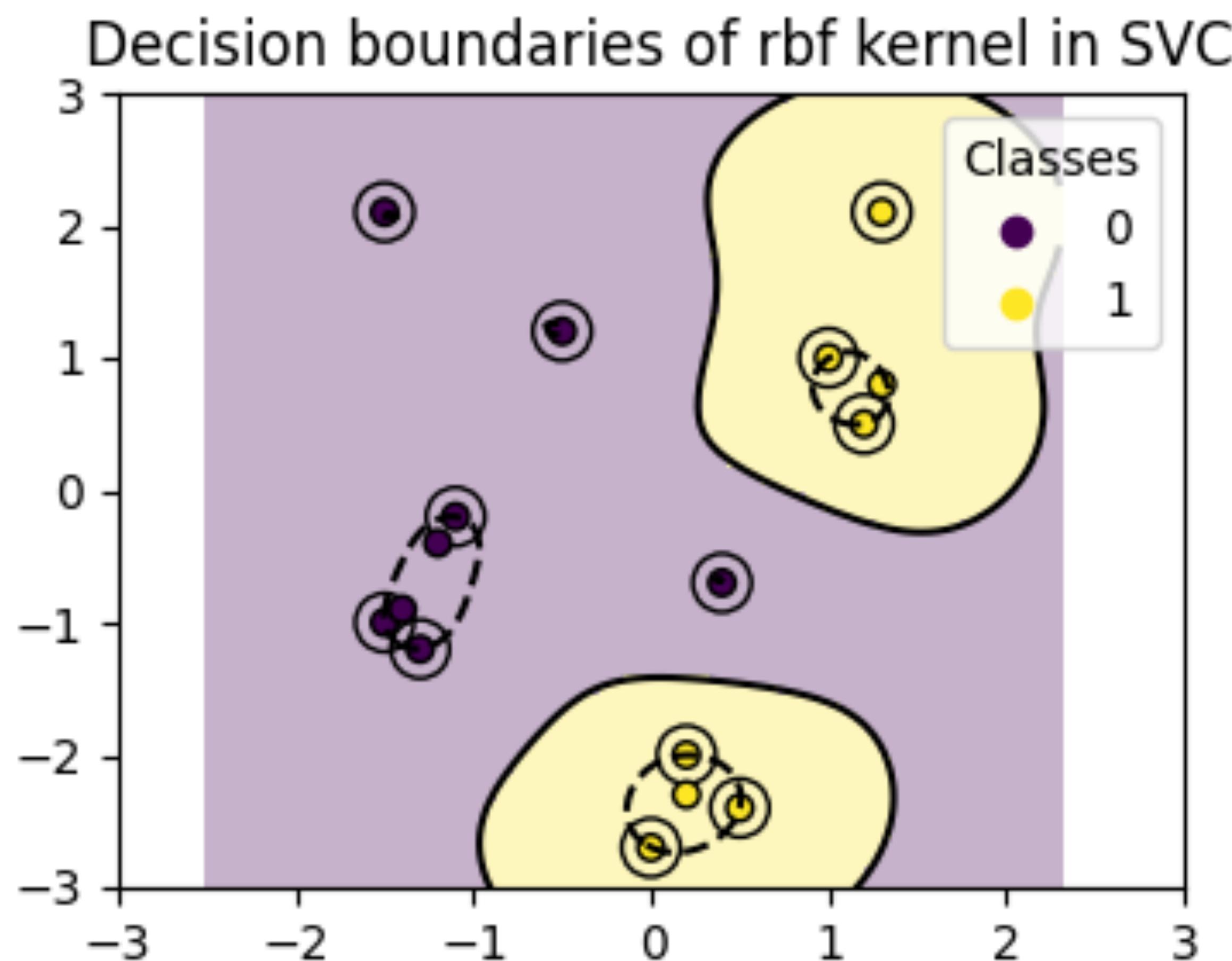
SVM

- Example -



SVM

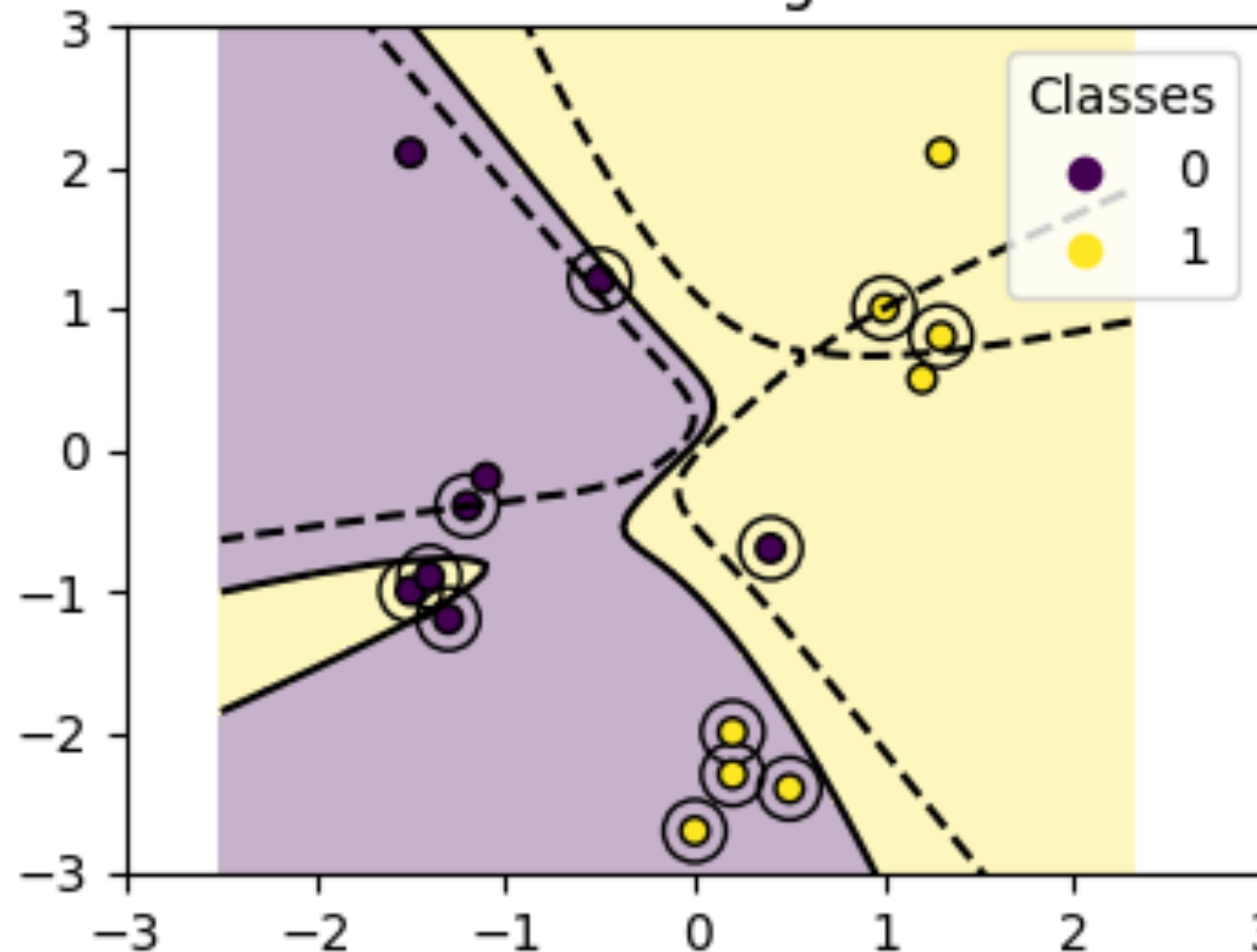
- Example -



SVM

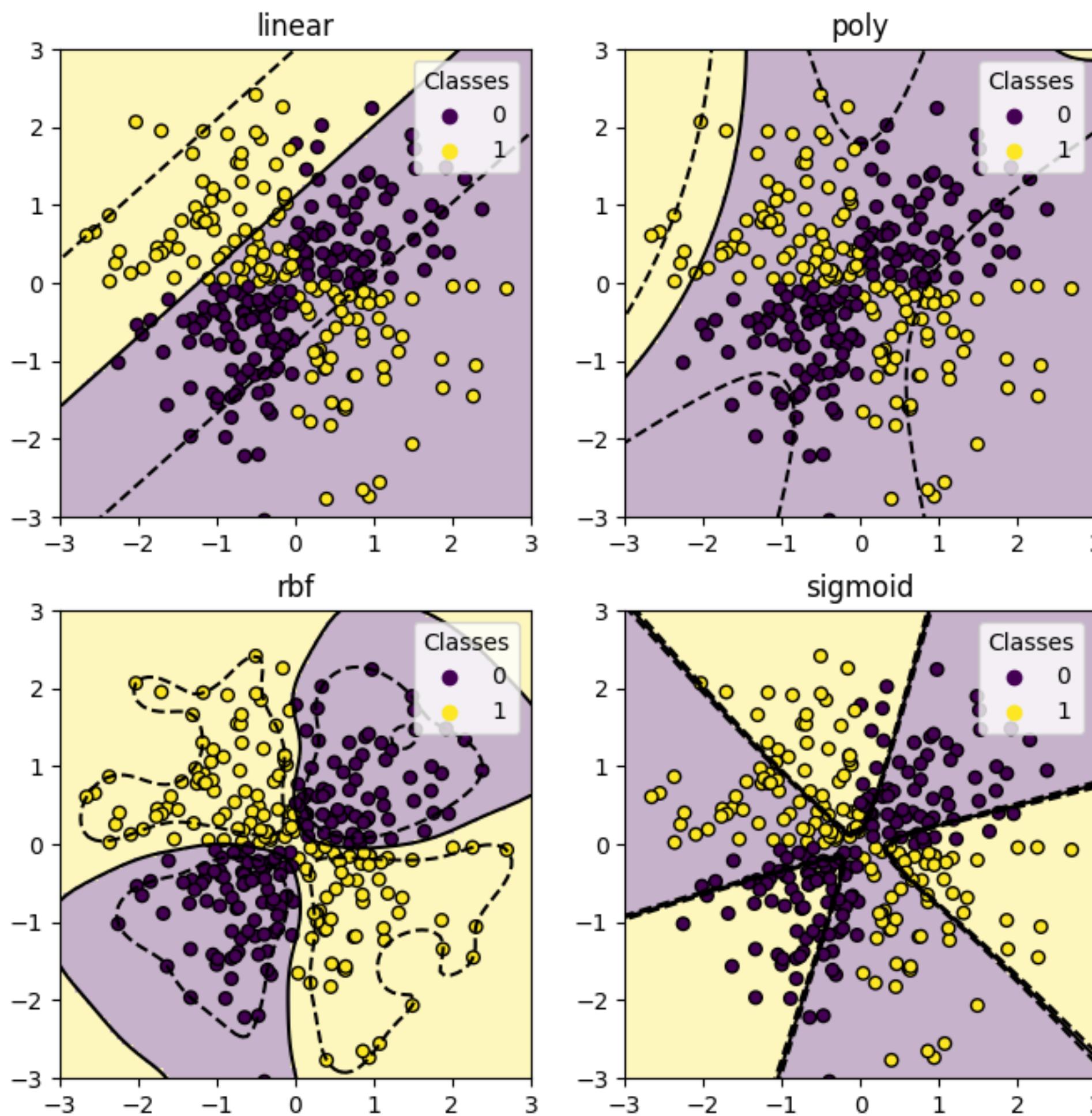
- Example -

Decision boundaries of sigmoid kernel in SVC



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
- ▶ XGBoost

❖ 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.

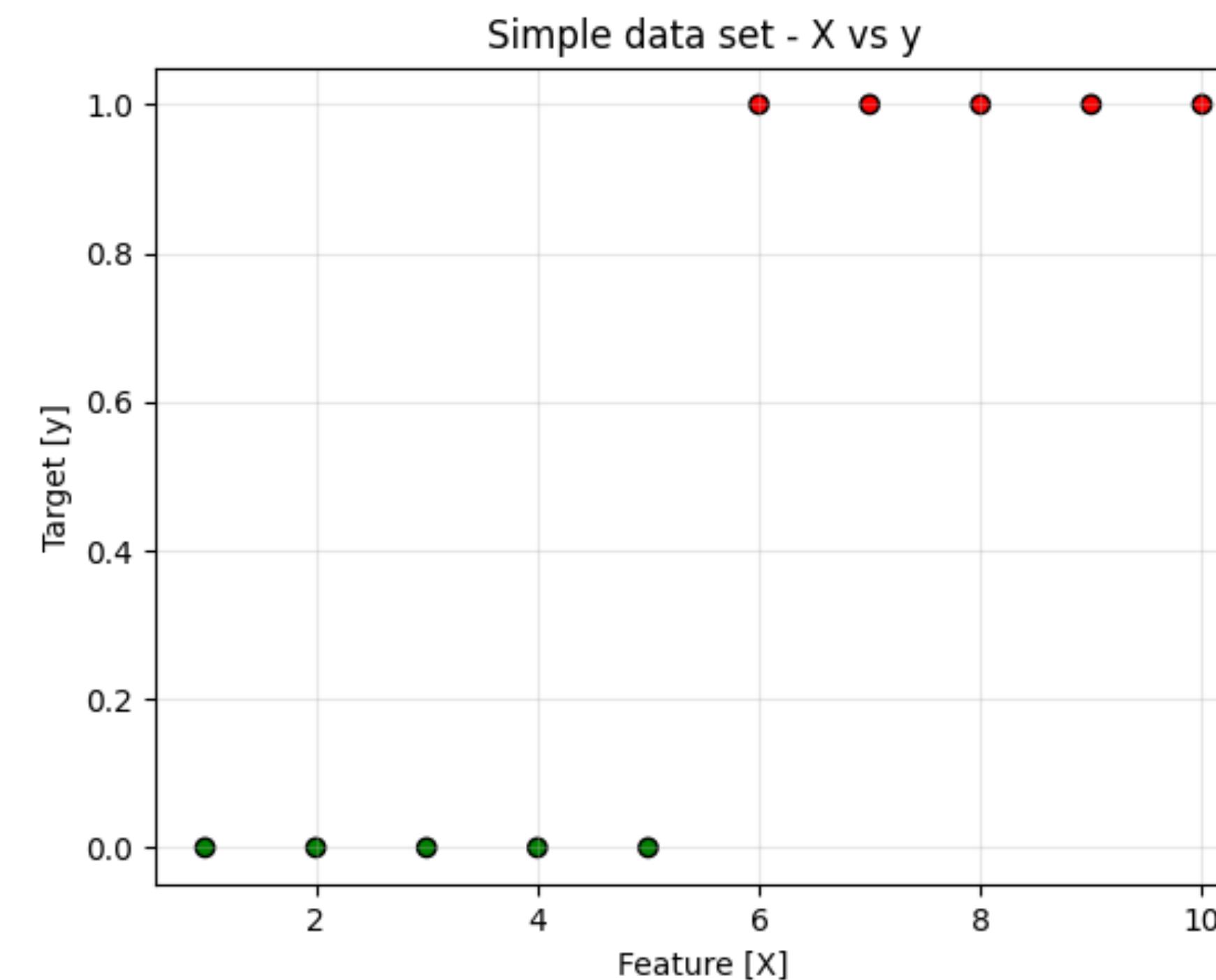
Goal : The idea is to split the data set into subsets based on feature values to increase the homogeneity of the target variable (reduce the variance).

Decision tree

- How does it work ? -

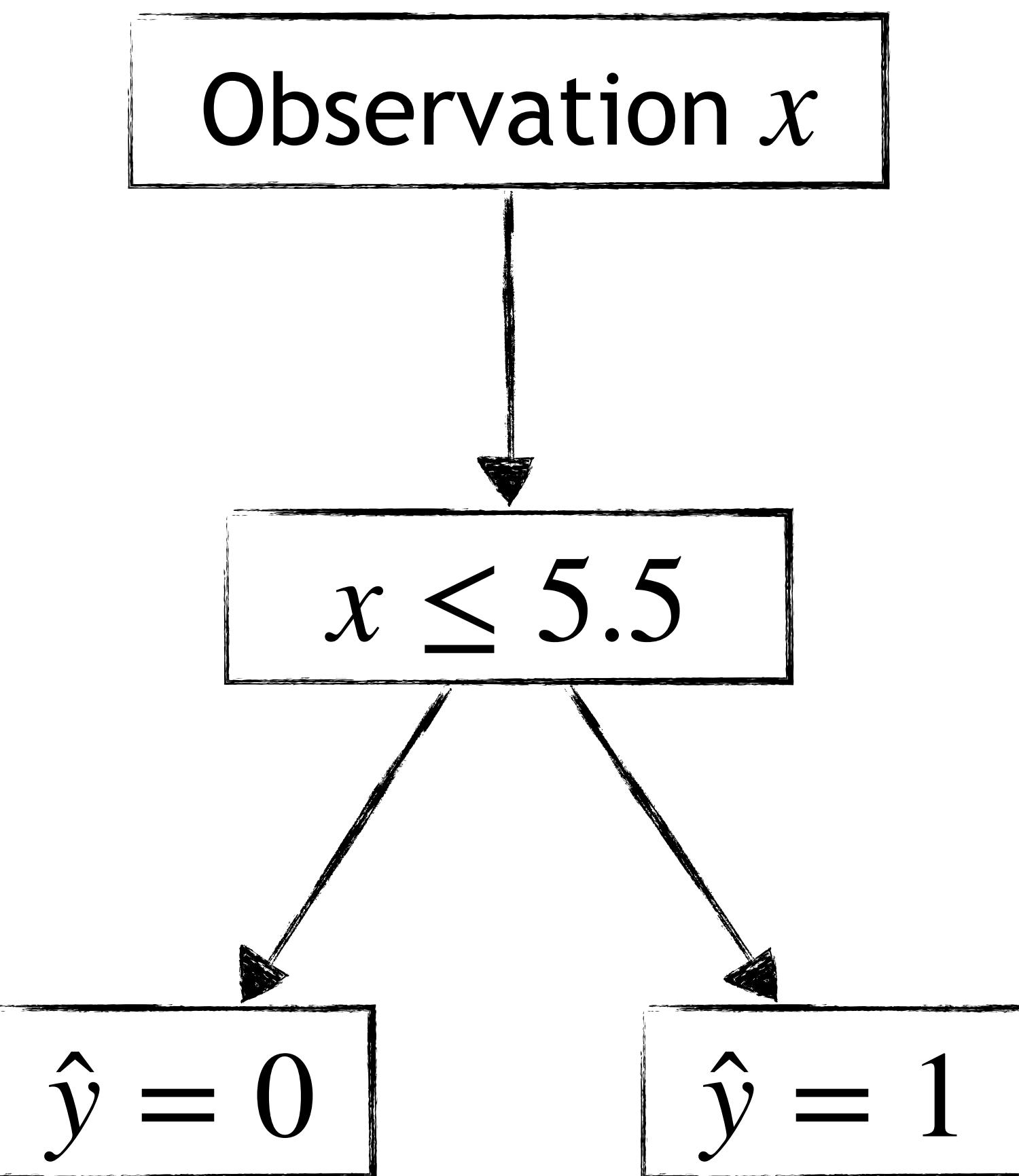
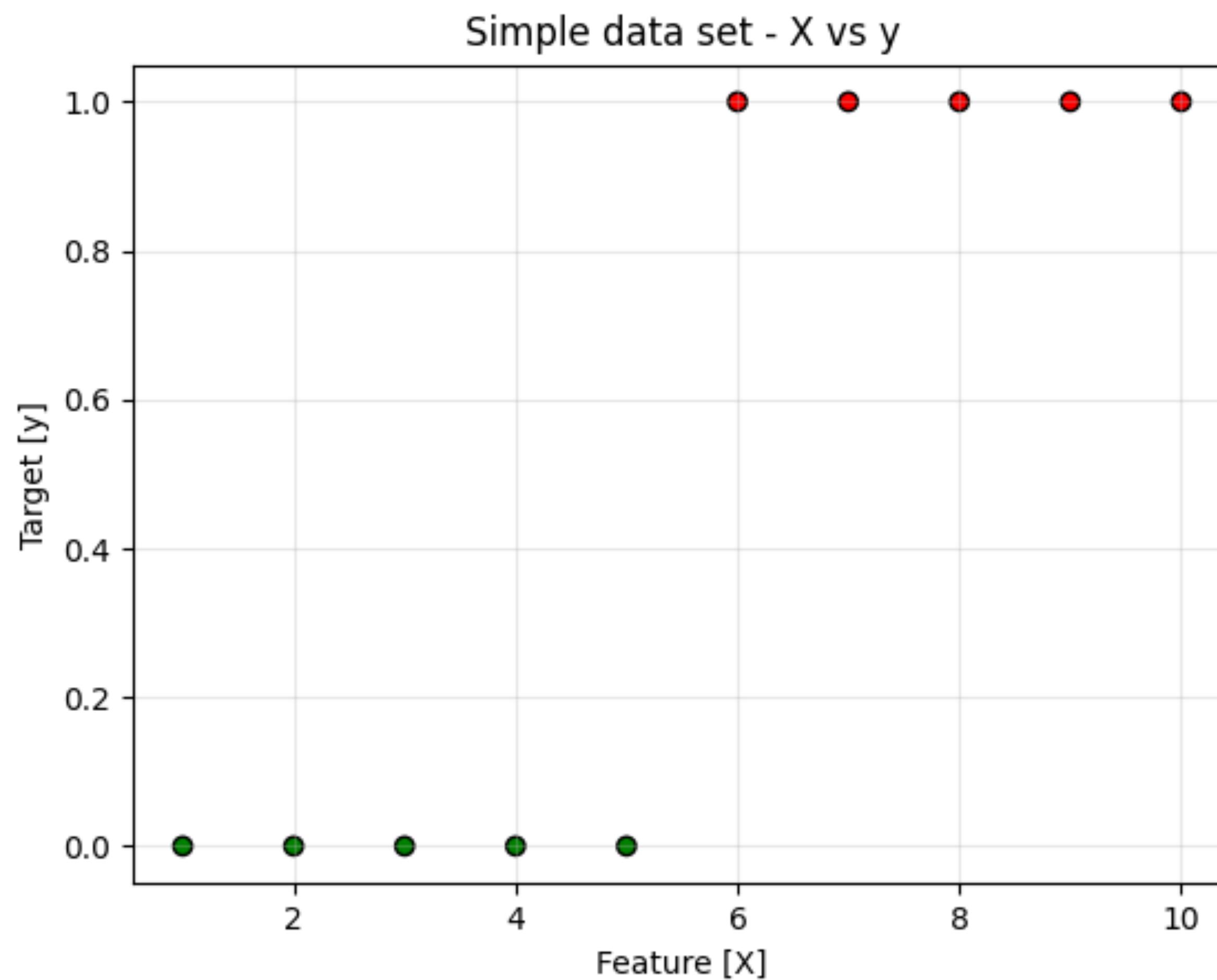
Idea : Suppose you have a binary classification task $\{0,1\}$ with a single continuous feature. Your data set has the following format :

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



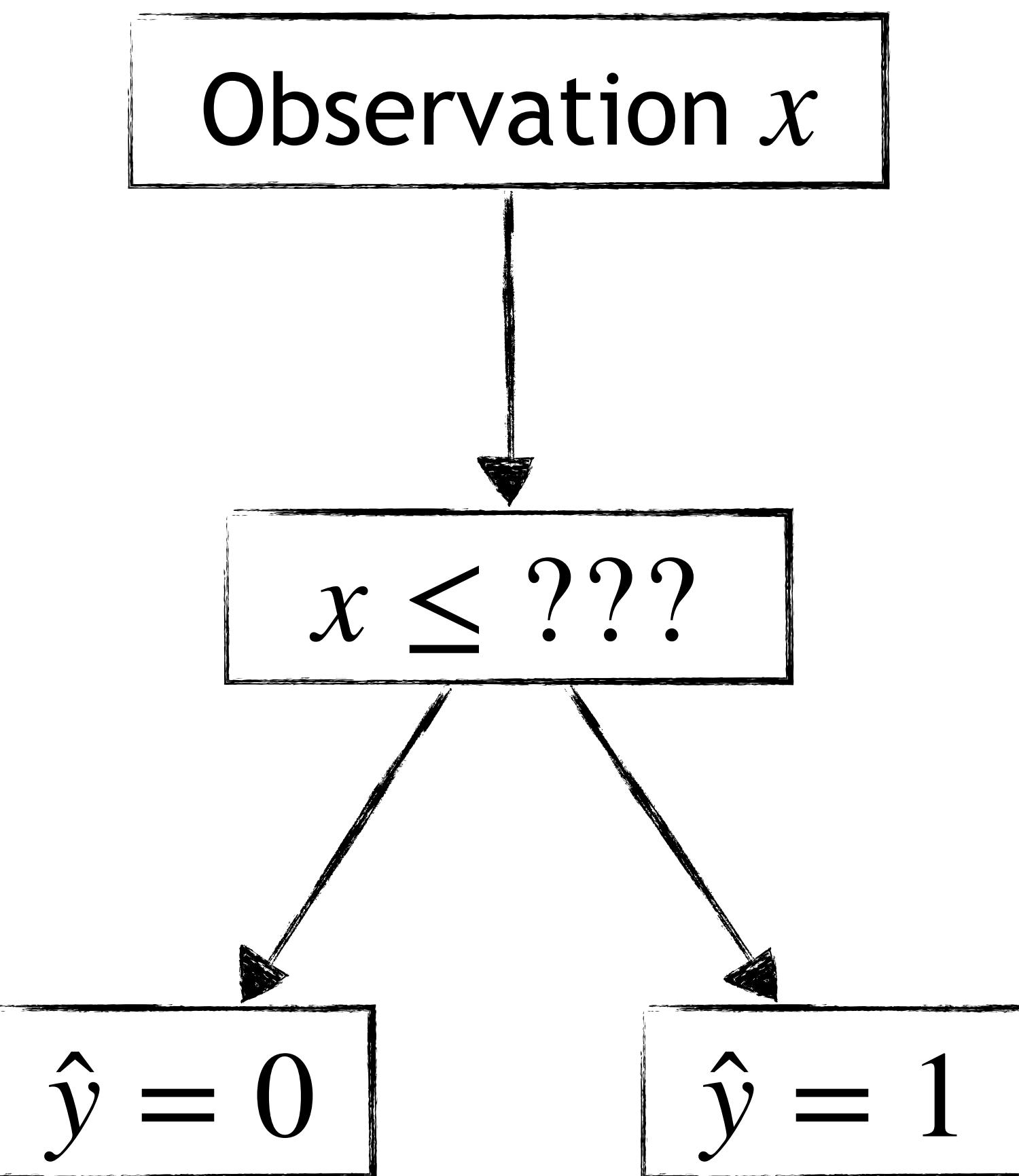
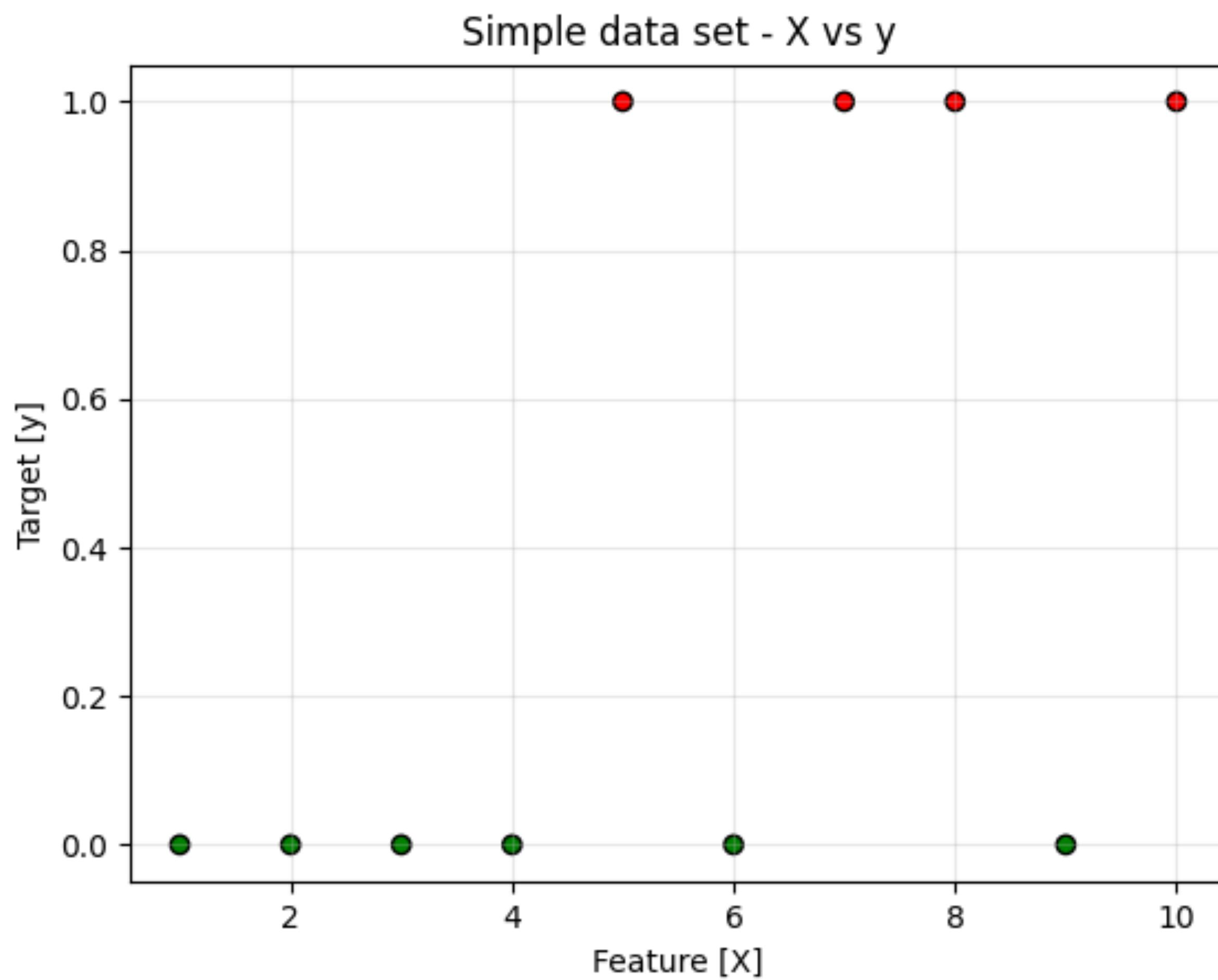
Decision tree

- How does it work ? -



Decision tree

- How does it work ? -



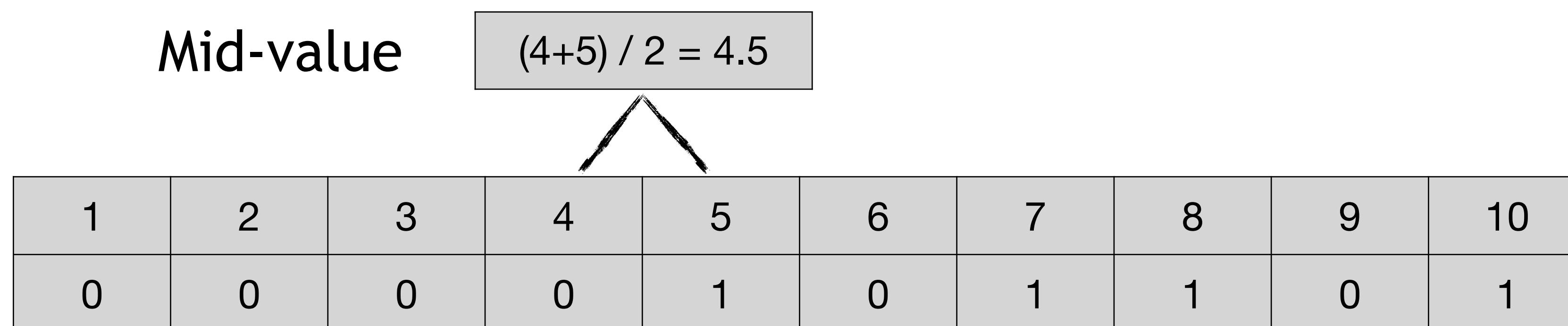
Decision tree

- 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

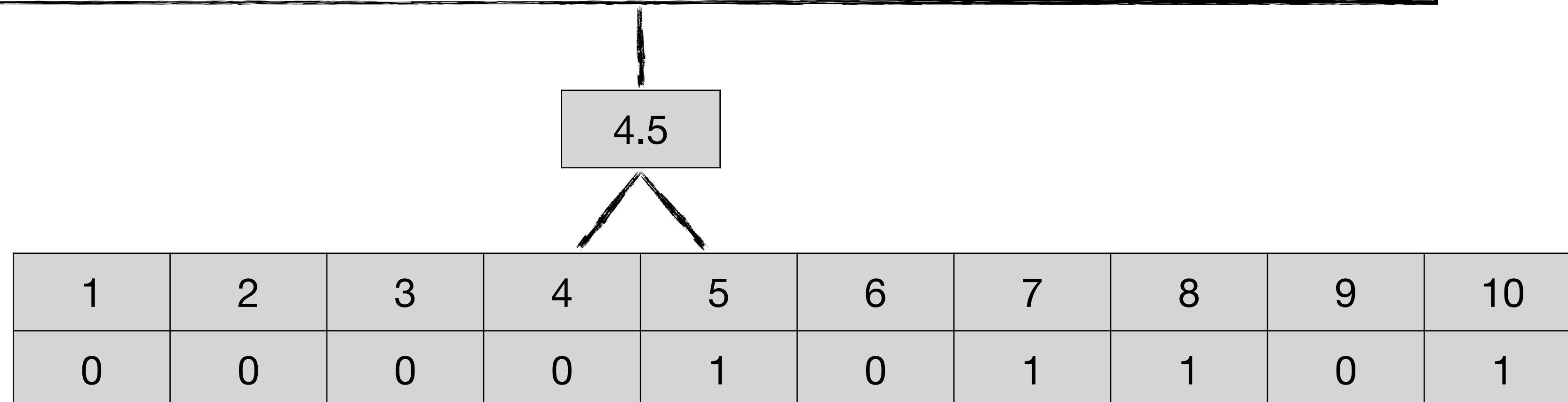
- How does it work ? -



Decision tree

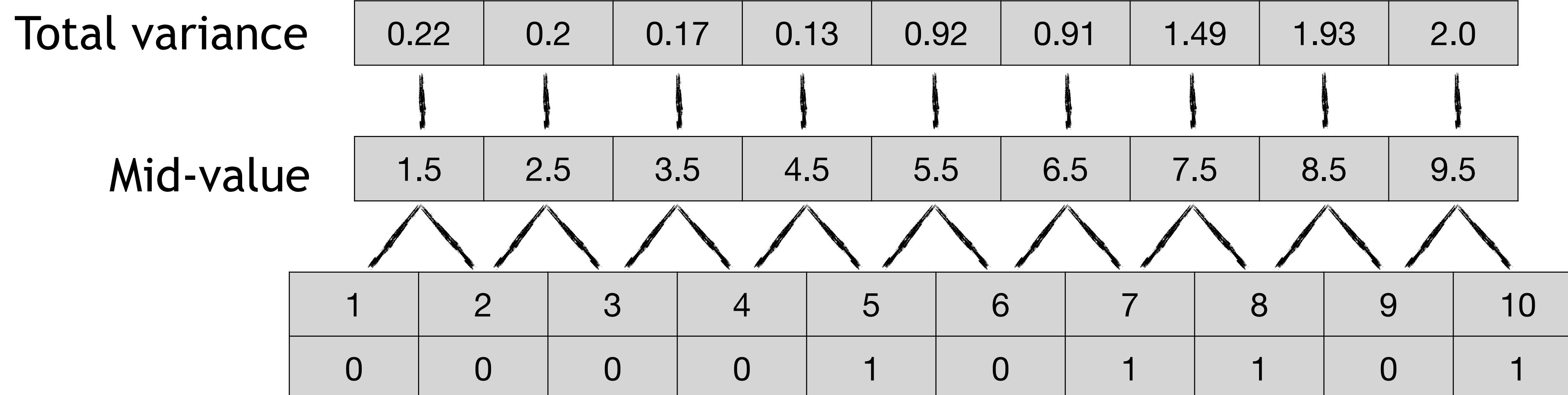
- How does it work ? -

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



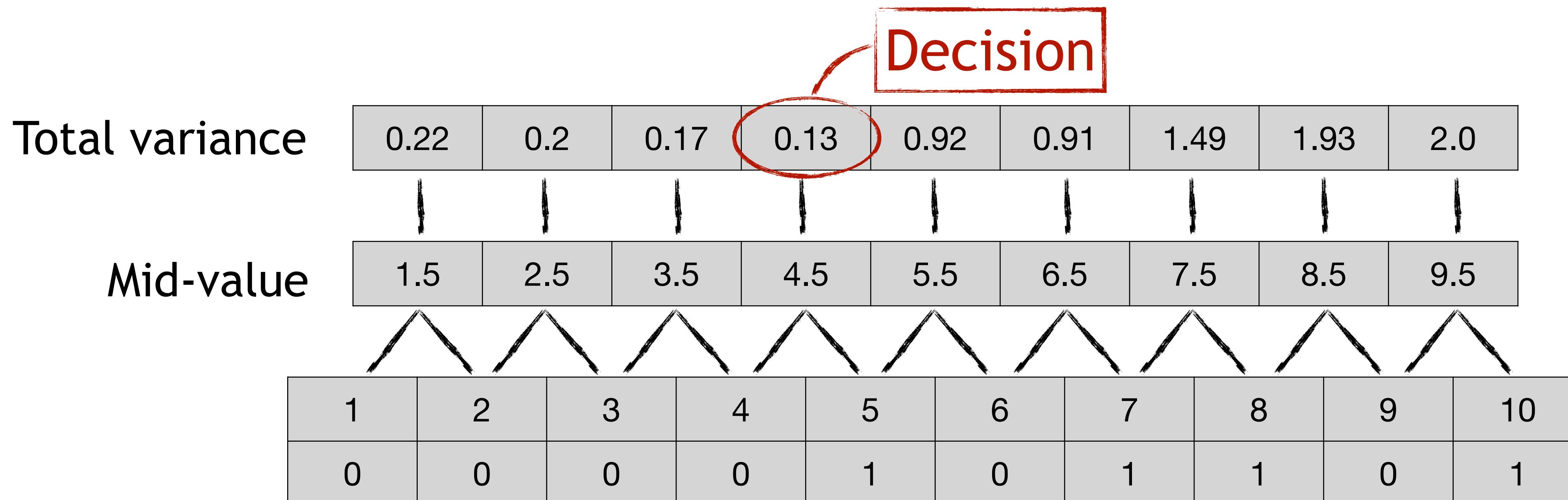
Decision tree

- How does it work ? -



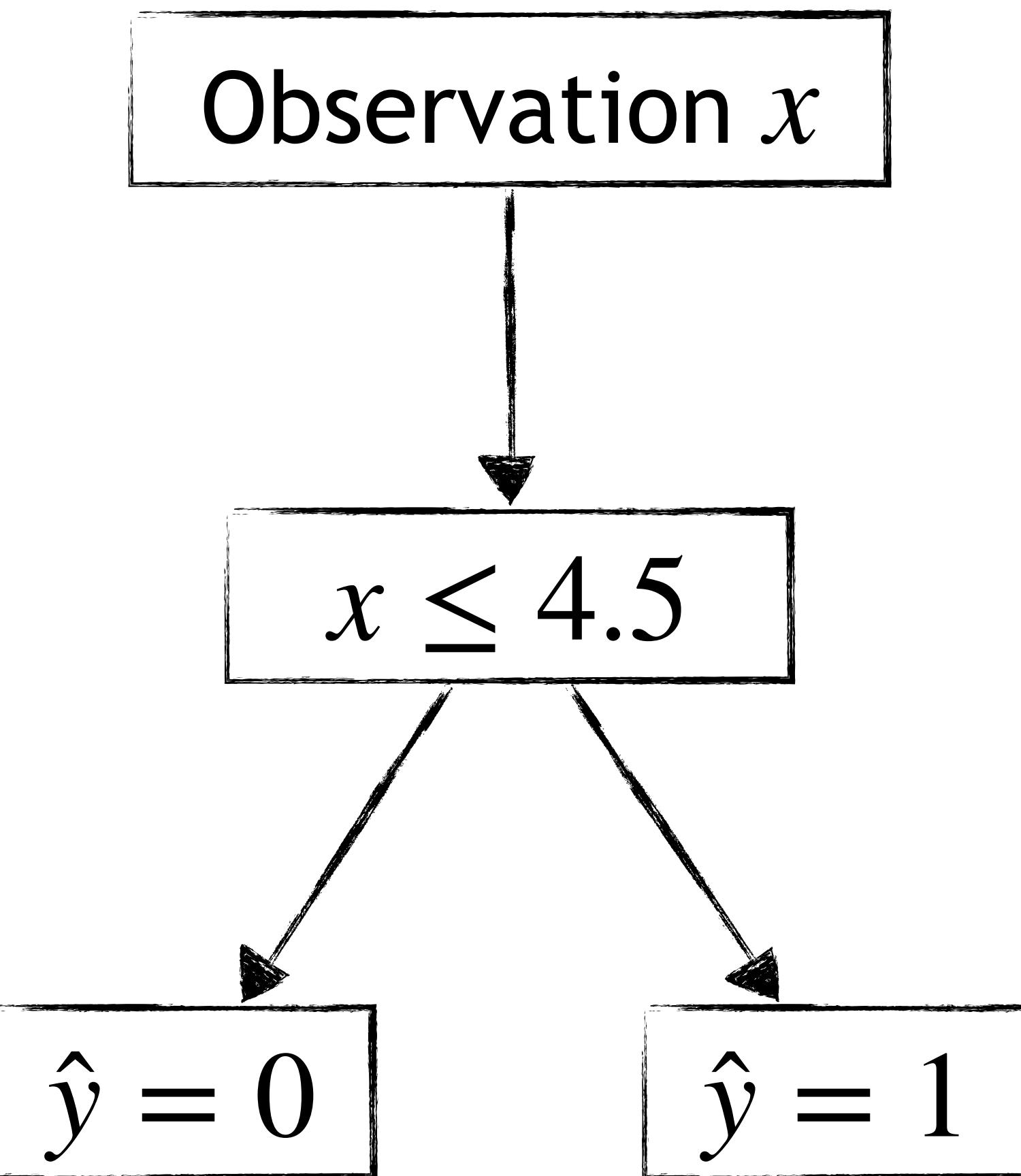
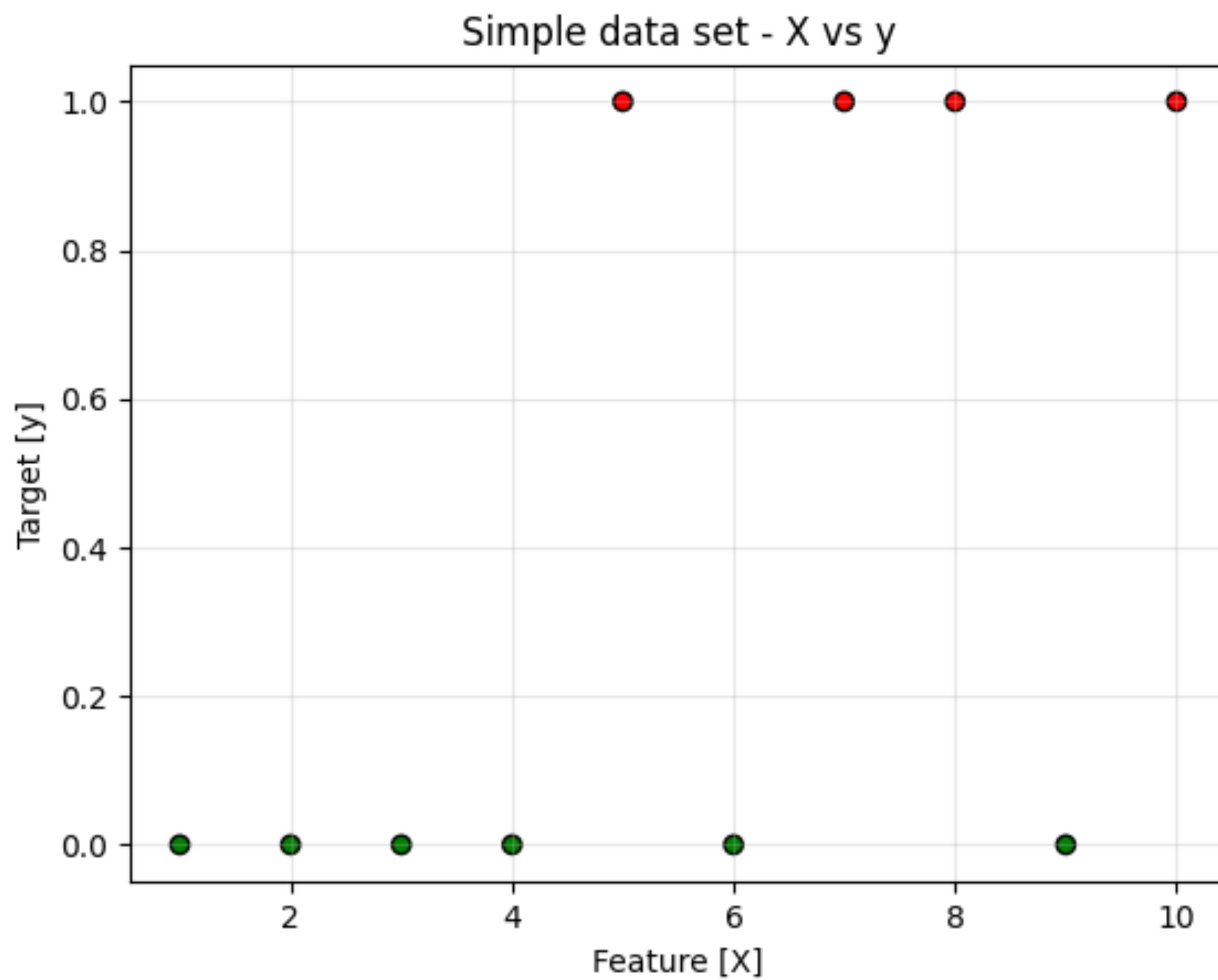
Decision tree

- 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

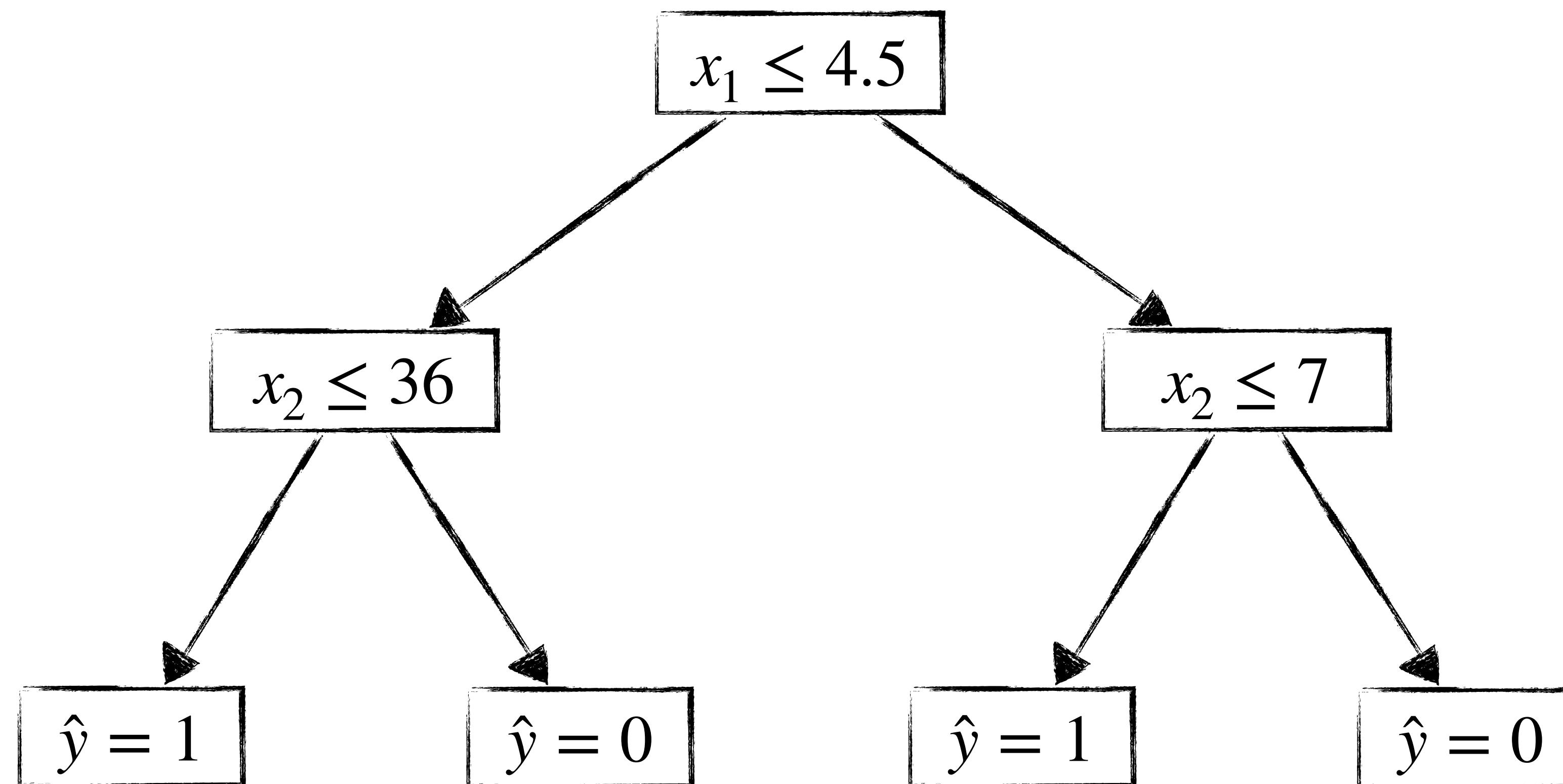
- And with more than one feature ? -

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

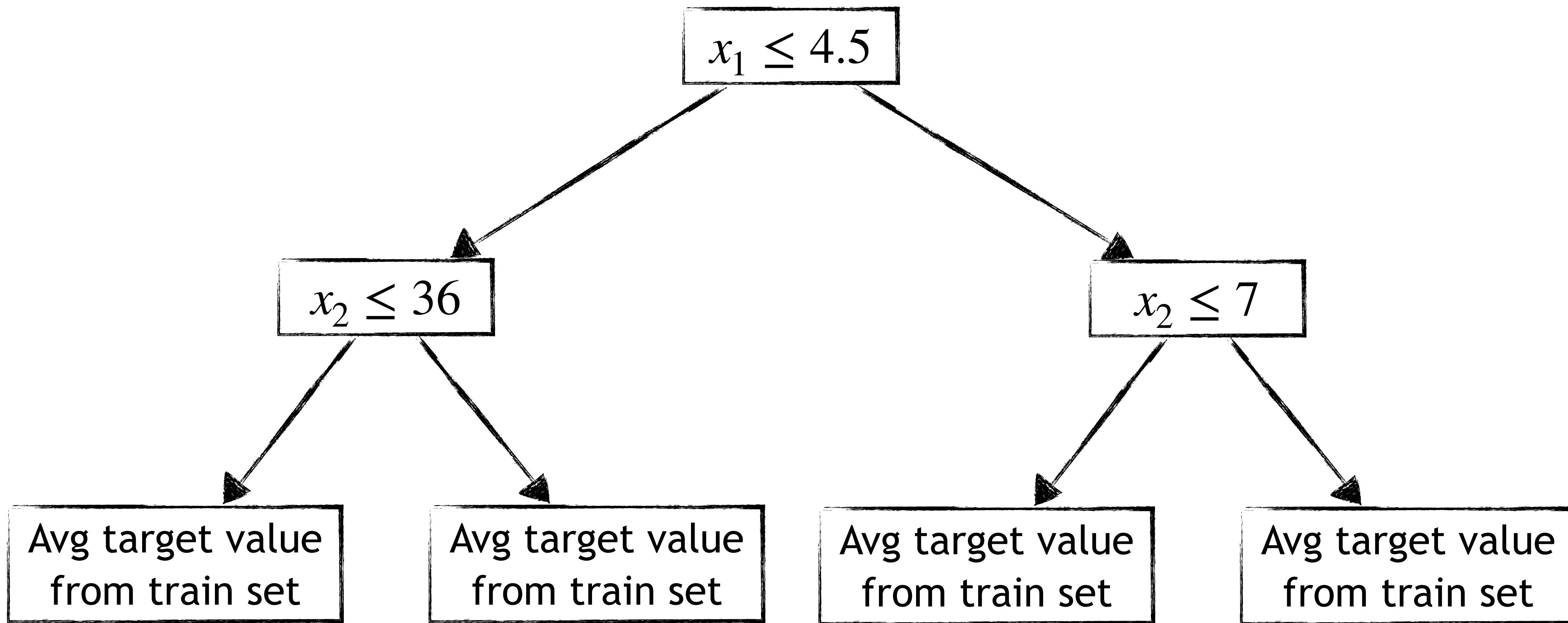
- And regression ? -

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 prediction given by the data observed in the train set.

Decision tree

- And regression ? -



Decision tree

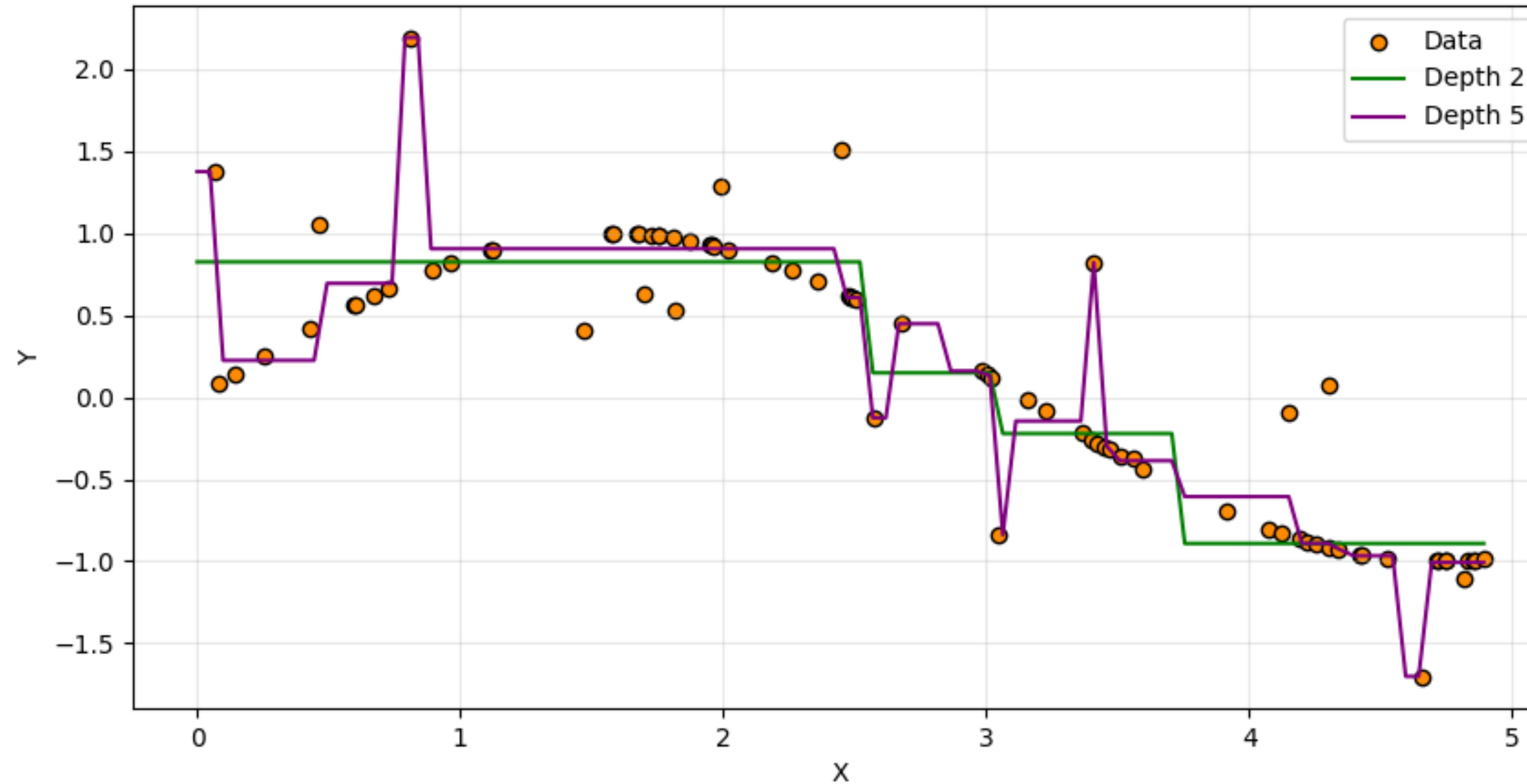
- Practice -

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 -

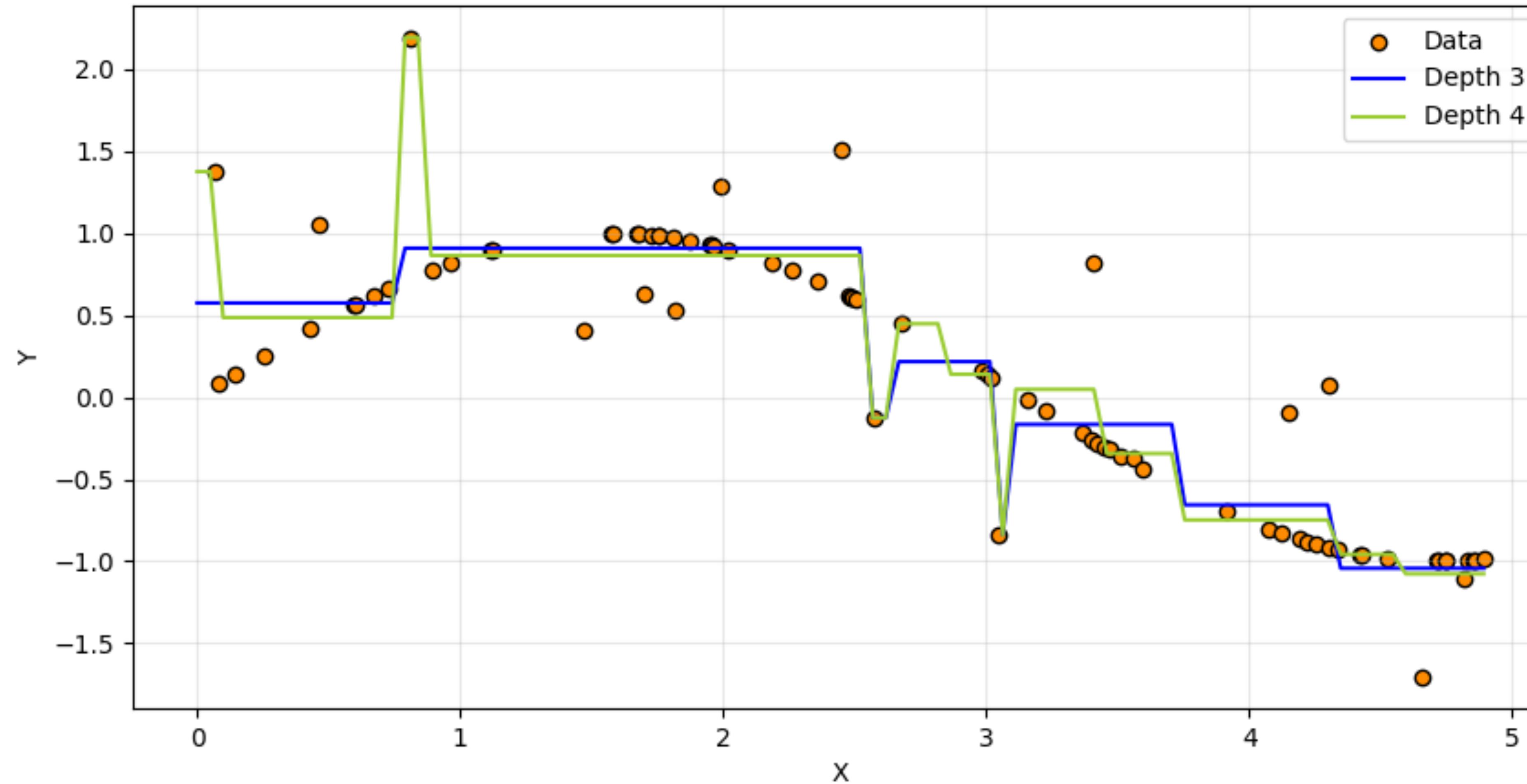
Tree data



Decision tree

- Practice -

Tree data



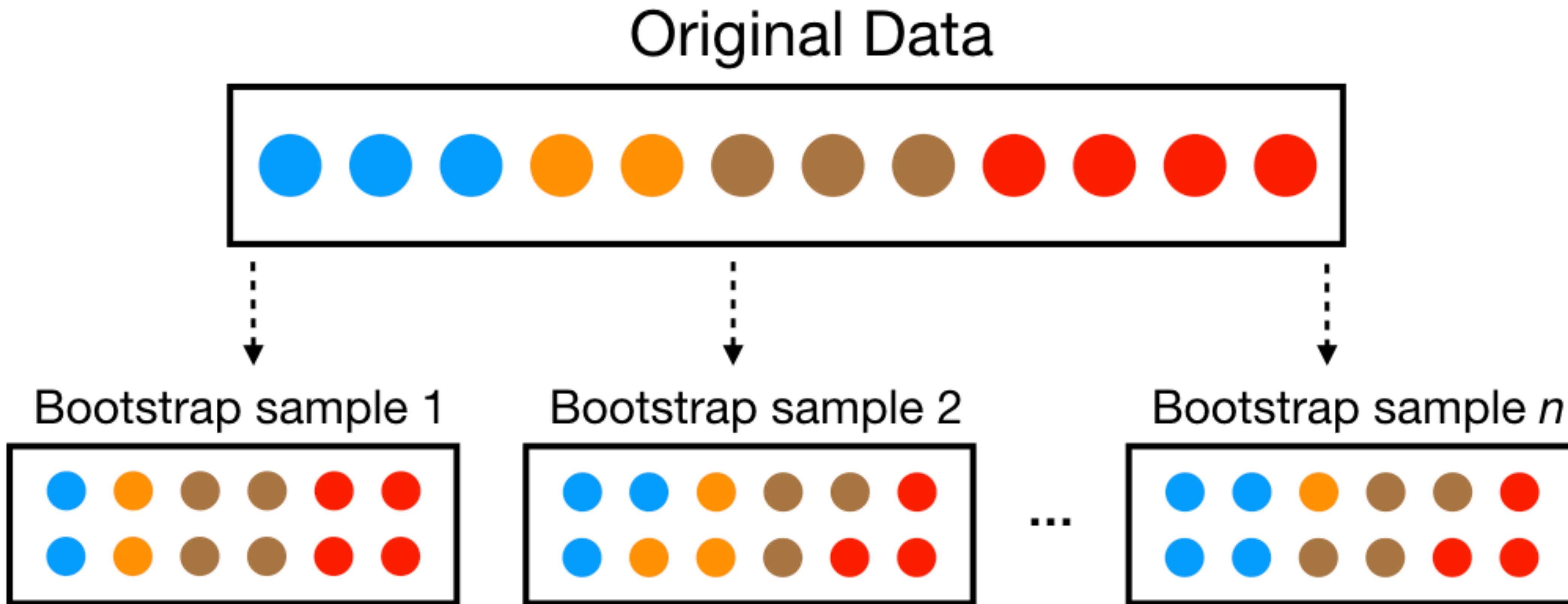
Random forest

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.

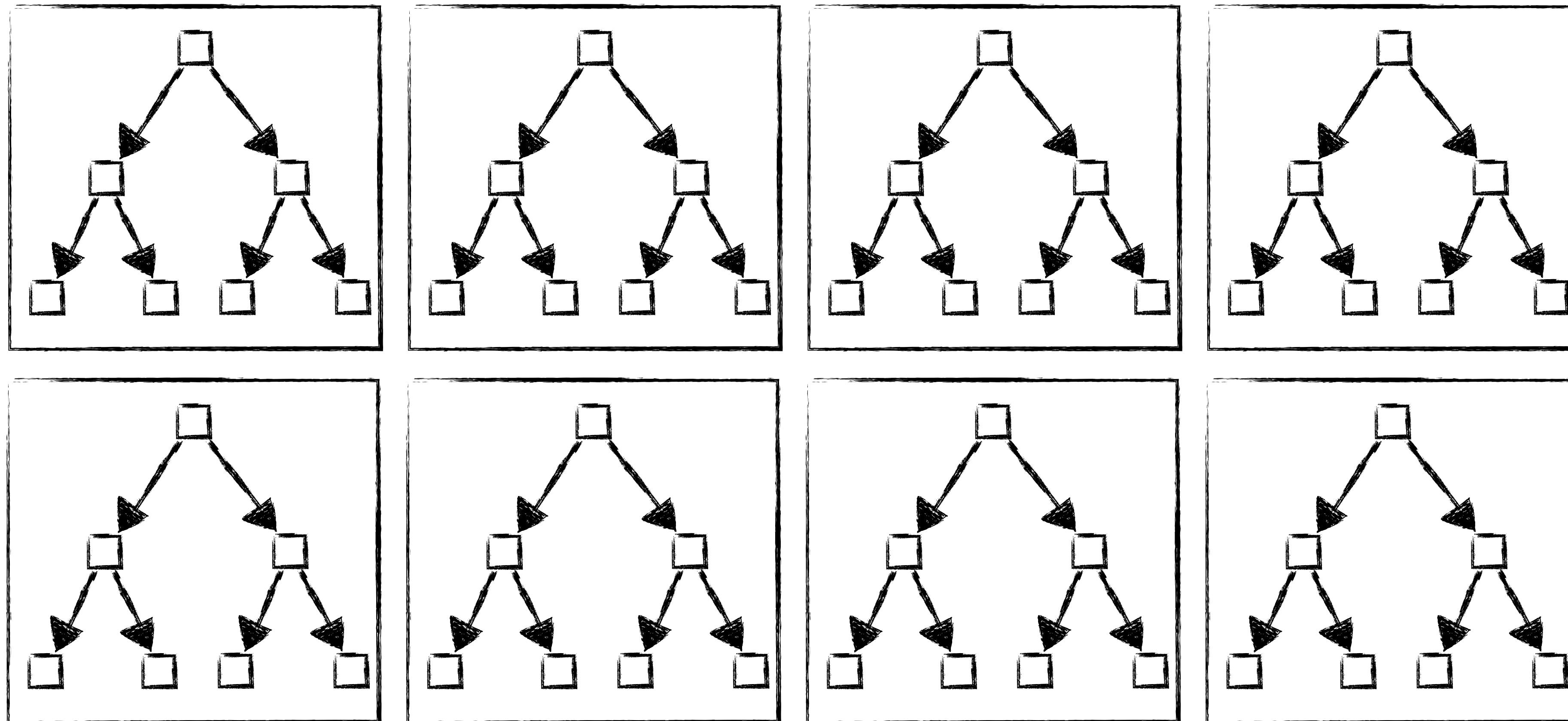
Bootstrap sample

- Random sampling with replacement -



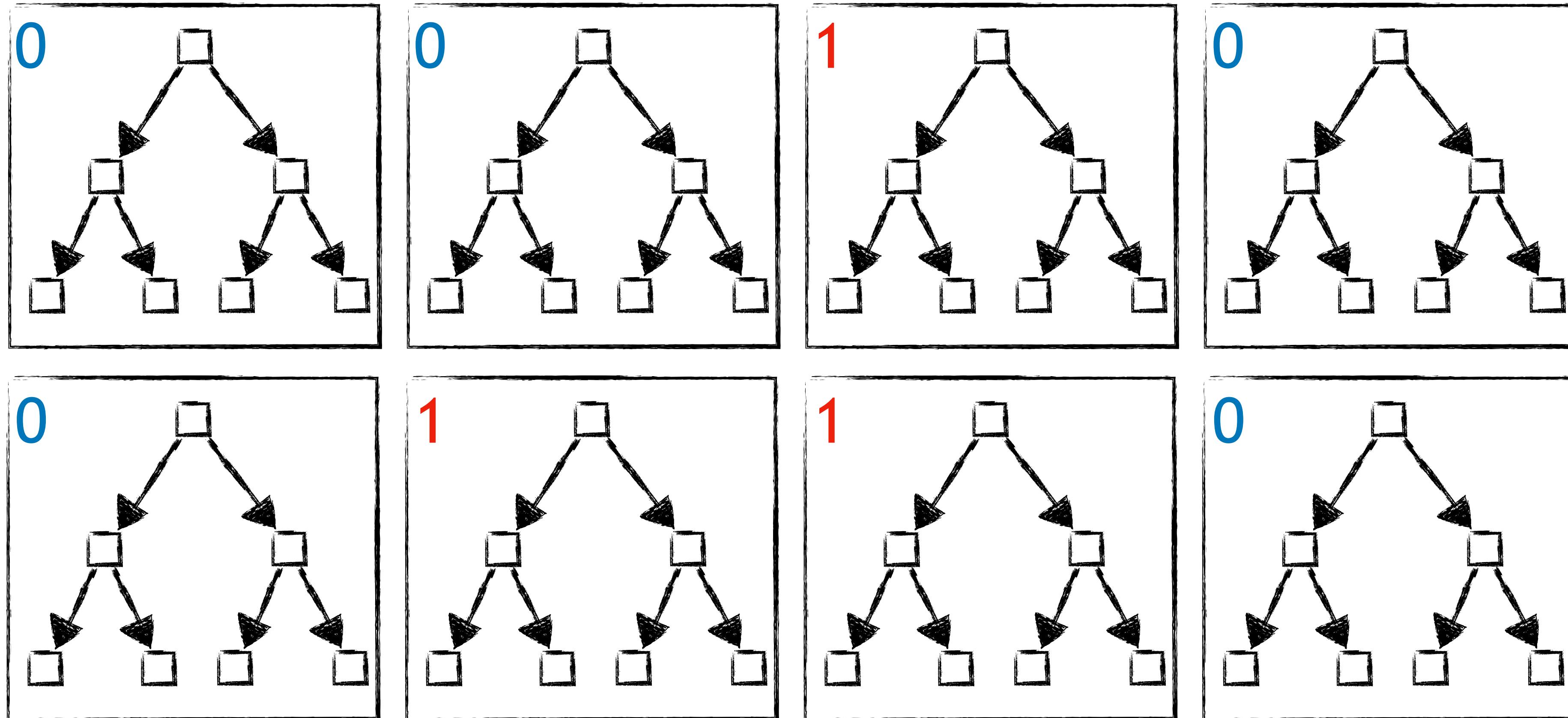
Random forest

- How does it work ? -



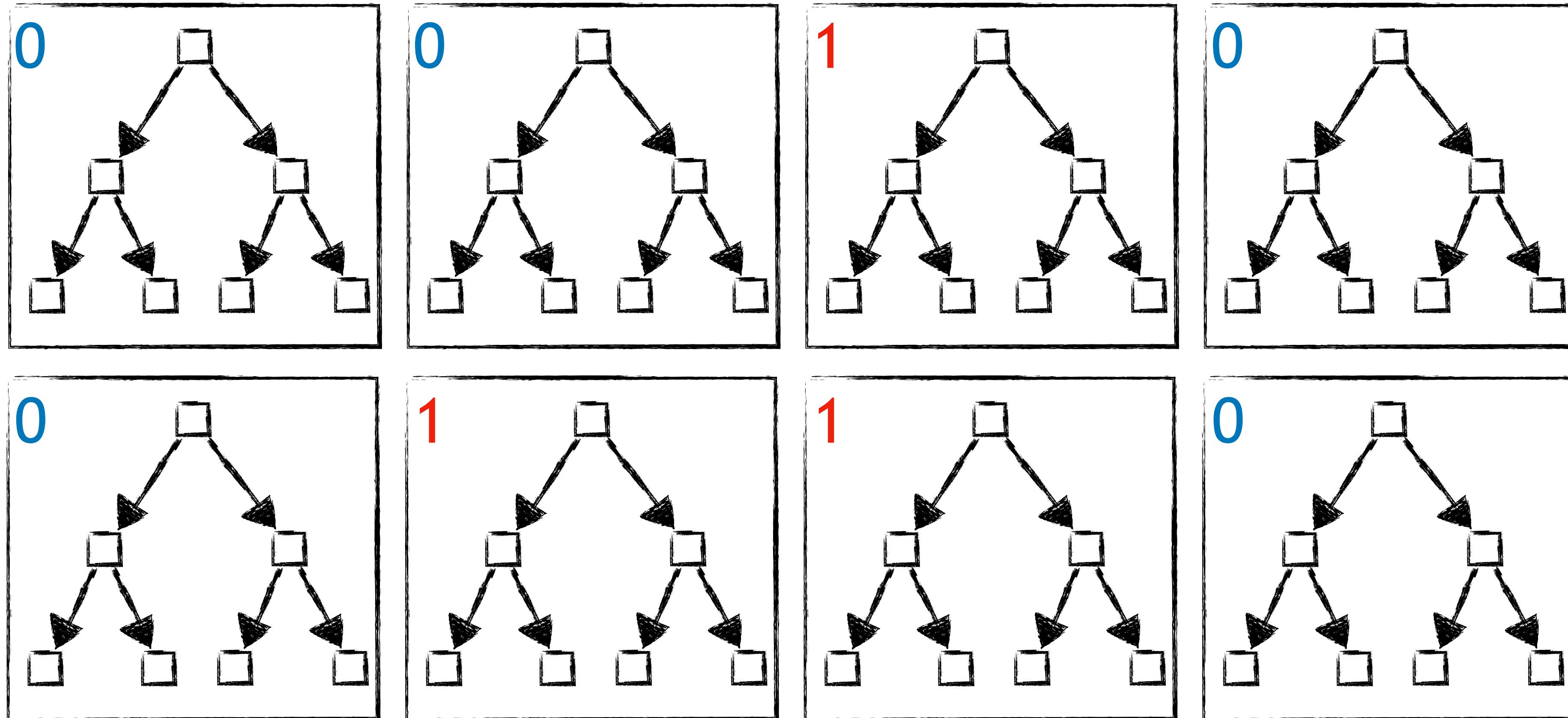
Random forest

- How does it work ? -



Random forest

- How does it work ? -



Voting

- Content -

Random forest

- Practice -

- Content -

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.

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

Not voting - Add log-odds

- Content -

XGBoost

- Practice -

- Content -

Conclusion

Decision trees

Random forests

XGBoost

- ❖ Monday : Metrics and training process
- ❖ Tuesday : Linear models
- ❖ Wednesday : Support Vector Machine models
- ❖ Thursday : Decision trees
- ❖ Friday : Time series analysis & semi-supervised
 - ▶ What ?
 - ▶ graph de performance (tol)