A gentle introduction to Machine Learning

Davide Albanese

- **We won’t talk about feature selection** (i.e. how to select the most important/predictive variables);
- **we won’t talk about feature extraction** (i.e. how to transform the variables into informative and non-redundant features) -> data dependent;
- **we won’t see algorithms in detail** -> no math here!
- **we won’t talk about deep learning!**

We will focus on **supervised classification** tasks and we’ll talk about:

- model selection;
- generalization and model assessment;
- how to write a basic predictive classification pipelines using Python and *scikit-learn.*

Machine Learning

Machine learning is programming computers to **optimize a performance criterion** using **example data** or **past experience**.

We need learning in cases where we cannot directly write a computer program to solve a given problem (e.g. vector sorting), but need example data or experience:

- when human expertise does not exist (e.g. spoken speech recognition, where we are unable to explain how we do it);
- when the problem to be solved changes in time…
- … or depends on the particular environment.

Machine Learning

Supervised Learning: learn from labeled data (e.g. case/control studies), learn the mapping between the data and the labels.

- **Classification**: when the labels are categorical (e.g. healthy/sick, benign/malignant, male/female, etc.)
- **Regression**: when the labels are continuous variables (e.g. temperature, pH, etc.)

Unsupervised Learning: learn from unlabeled data, discover some hidden structure in the data.

- **Clustering**: grouping similar samples according their similarity;
- **Dimensionality reduction**: transform the variables into informative and non-redundant features (e.g. PCA)

Machine Learning

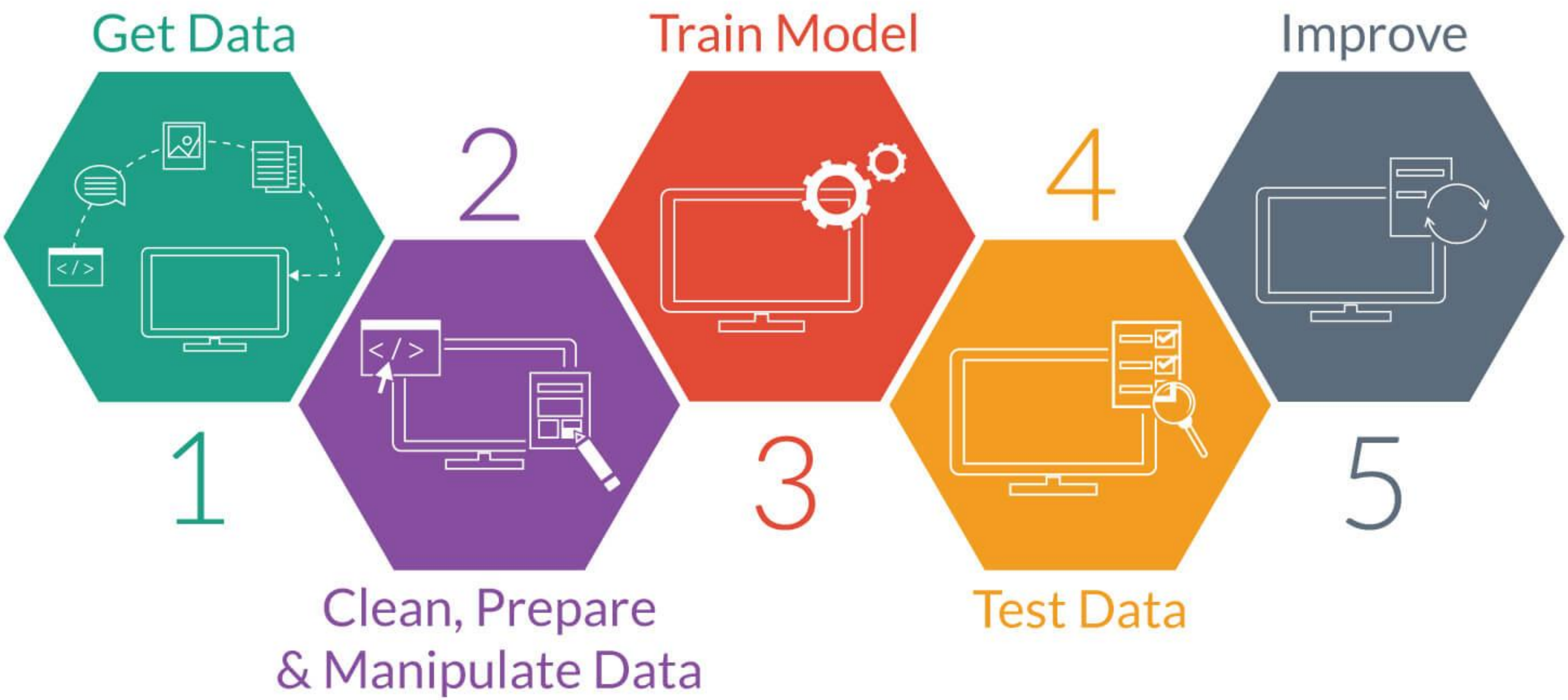
Supervised Learning: learn from labeled data (e.g. case/control studies), learn the mapping between the data and the labels.

- **Classification**: when the labels are categorical (e.g. healthy/sick, benign/malignant, male/female, etc.)
- **Regression**: when the labels are continuous variables (e.g. temperature, pH, etc.)

Unsupervised Learning: learn from unlabeled data, discover some hidden structure in the data.

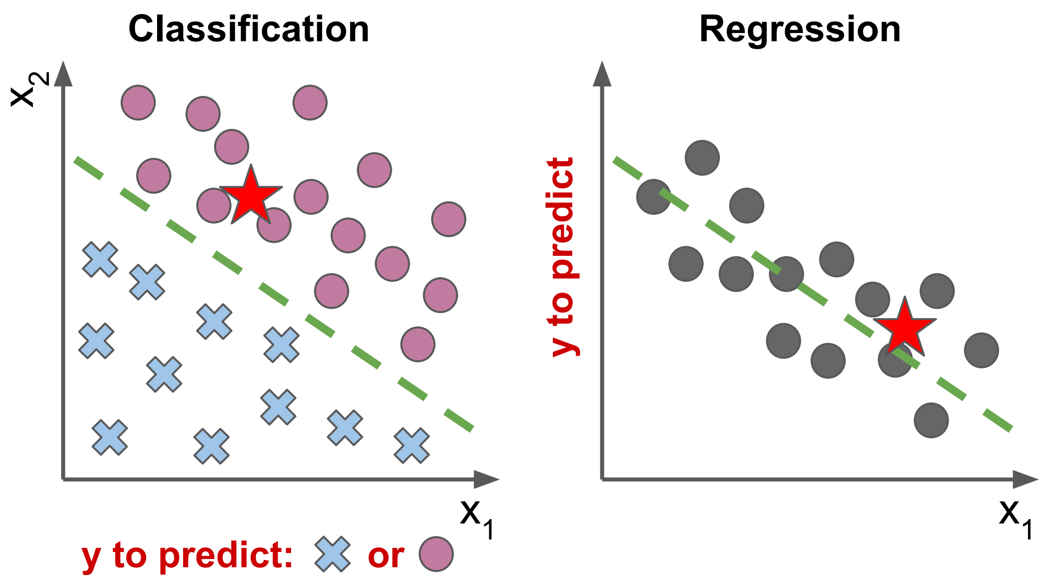
- **Clustering**: grouping similar samples according their similarity;
- **Dimensionality reduction**: transform the variables into informative and non-redundant features (e.g. PCA)

Steps to Predictive Modelling



Supervised Learning - problem setting

- **Classification**: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data
- **Regression**: the additional attribute we want to predict is a continuous variable



- x_1, x_2: **features** (a.k.a. variables, attributes, predictors, covariates)
- *points*: **samples** (a.k.a. observations, instances)
- *y*: **target** (a.k.a. labels, response, outcome) what we want to predict

Supervised Learning - problem setting

Train (fit)

X matrix

Features

	x_1	**x_2**	**x_3**
S1	0.2	1.4	0.3
S2	2.1	0.5	1.6
S3	1.1	1.3	2.1
S4	0.8	1.0	2.1
S5	0.9	1.6	1.7

Samples

y vector

classification	regression
0	**12**
0	**34**
1	**21**
1	**22**
1	**12**

Predict

New sample

	x_1	**x_2**	**x_3**
SN	1.1	1.4	1.6

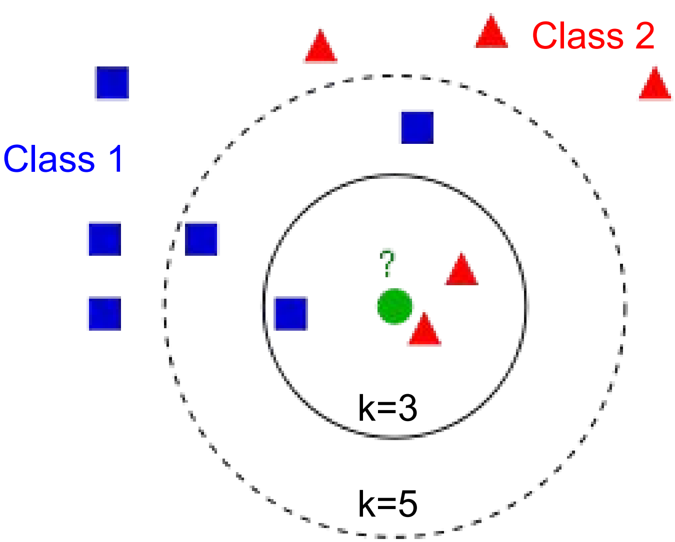
?

00_problem.ipynb

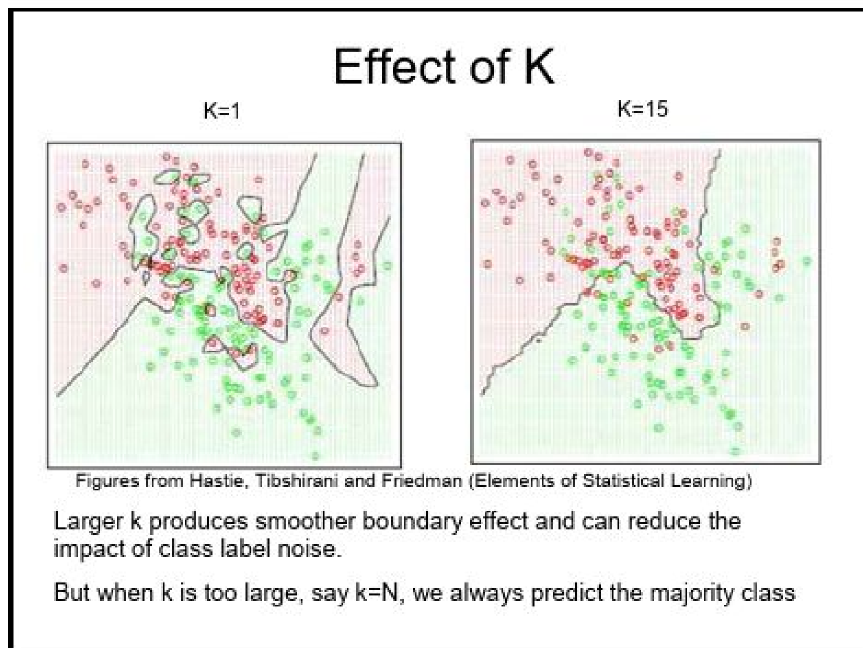
The k-nearest neighbor (k-NN) classifier

The k-nearest neighbors (k-NN) is a simple algorithm that stores all available data points and classifies new instances based on a similarity measure (e.g., distance functions).

k-NN is a type of instance-based learning (a.k.a lazy learning).



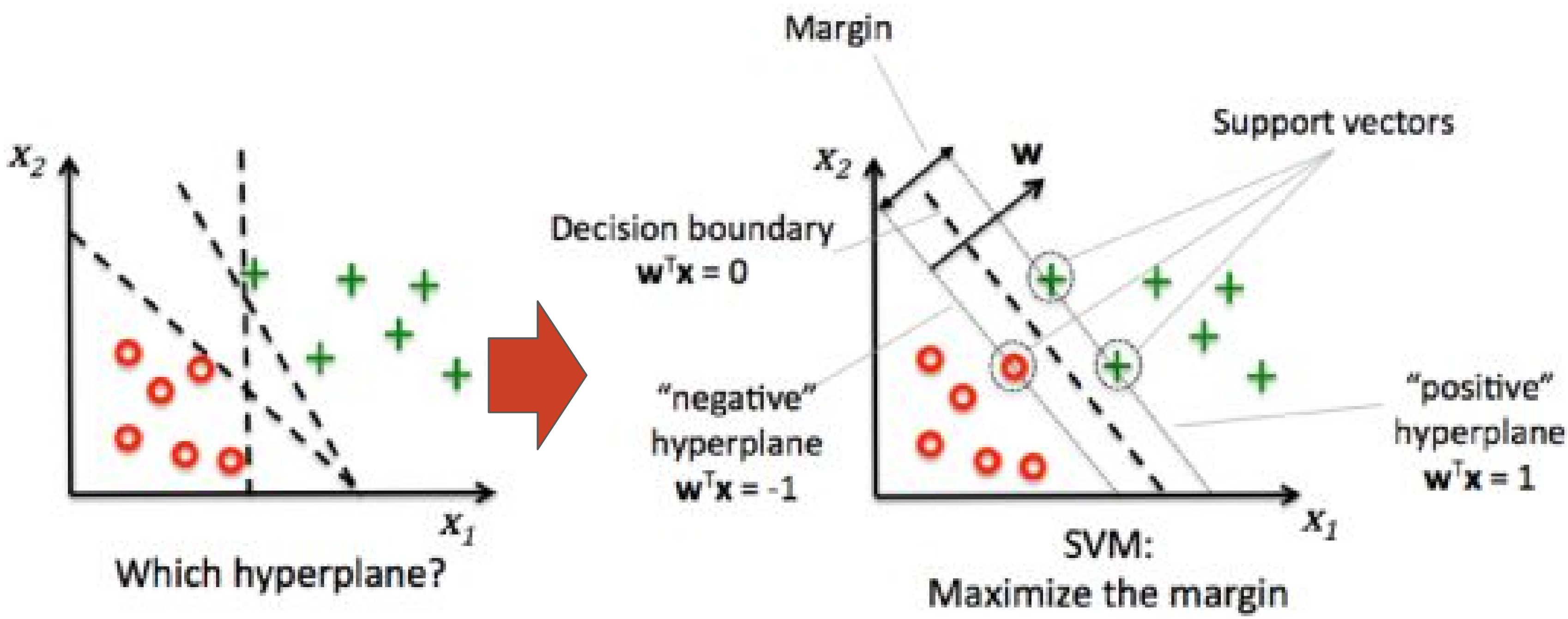
From wikipedia.org



The Support Vector Machine (SVM)

The SVM is a powerful and widely used linear classifier.

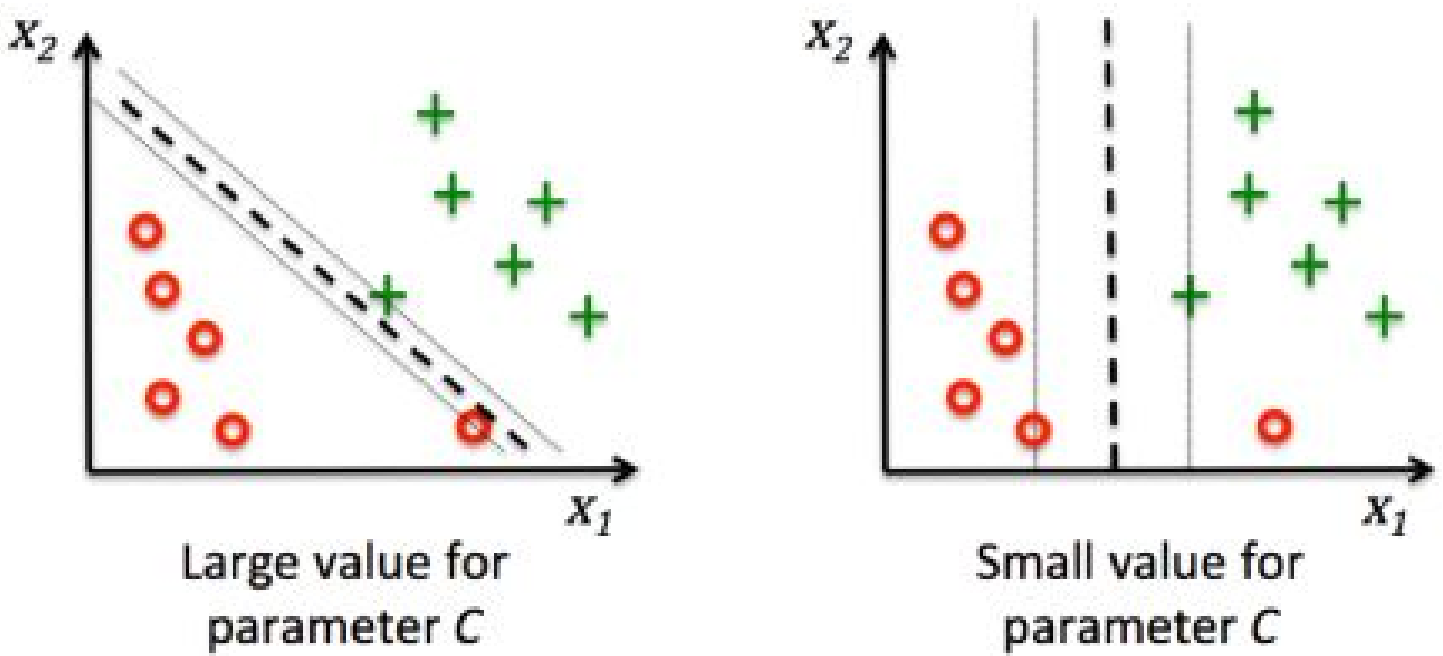
The key idea is that one reasonable choice as the best hyperplane is the one that **represents the largest separation (margin)**, between the two classes.



From: http://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_Support_Vector_Machines_SVM.php

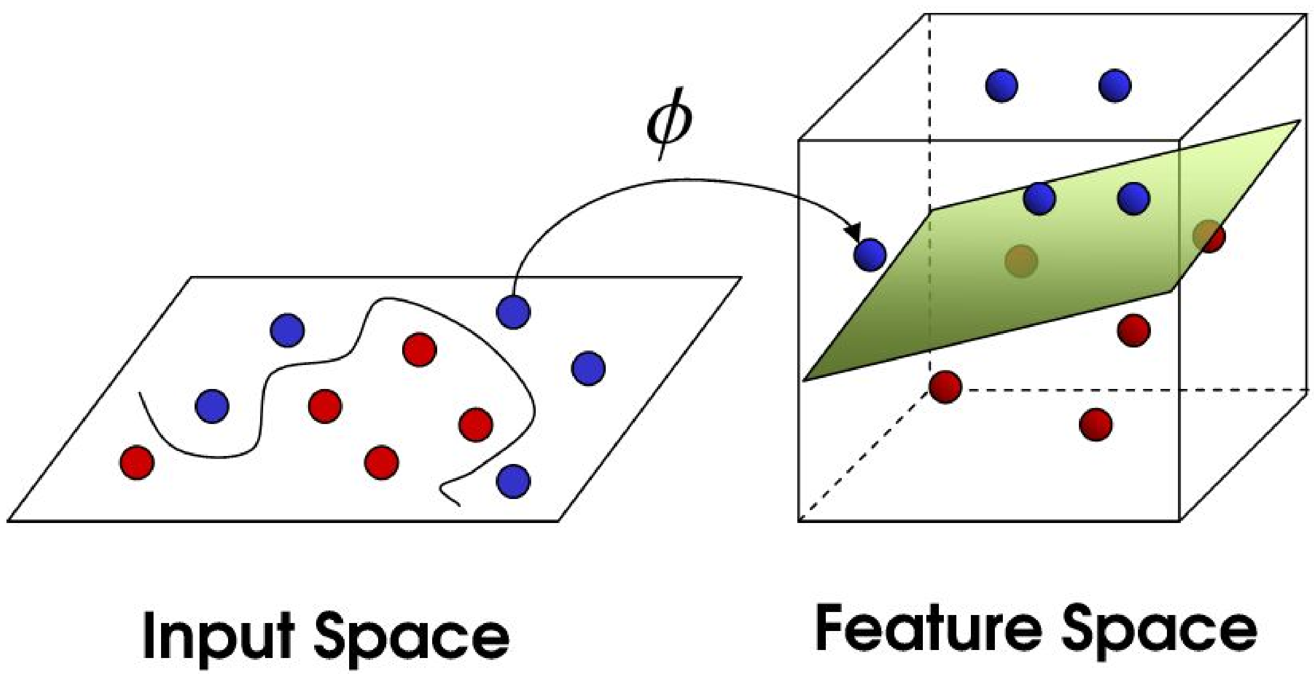
The Support Vector Machine (SVM)

The SVM's penality term *C* controls the tradeoff between the size of the margin and the misclassification error (bias-variance tradeoff).



The Support Vector Machine (SVM)

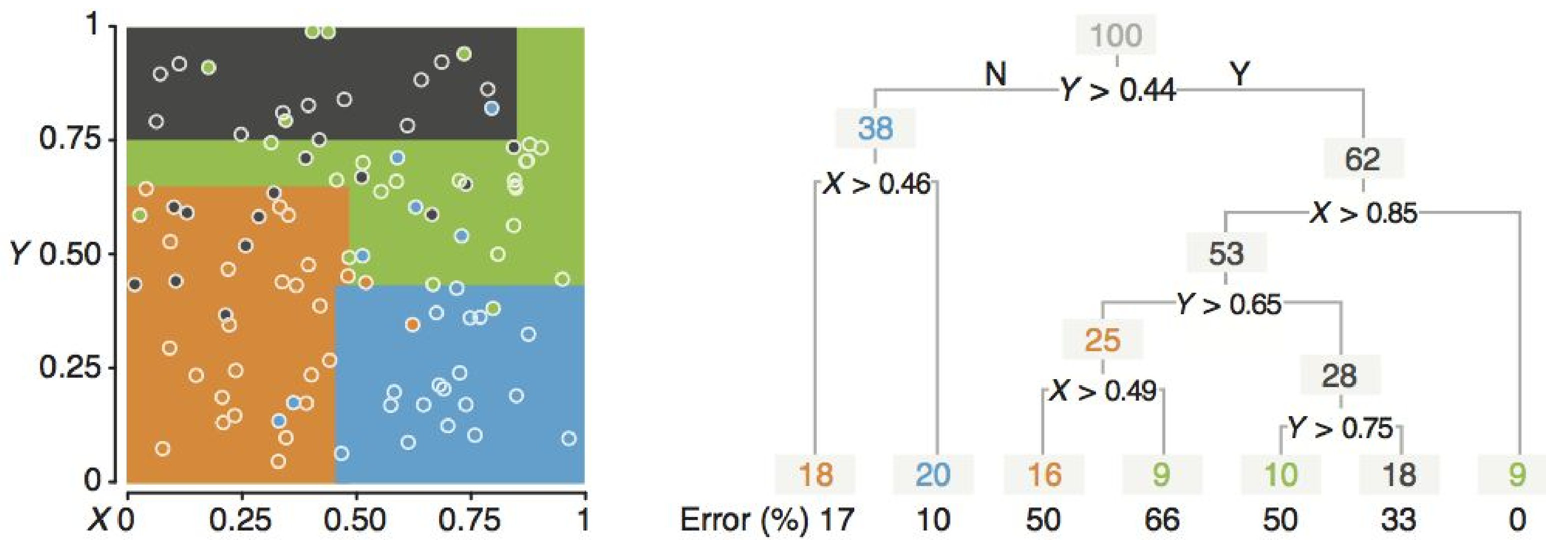
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, mapping their inputs (with a kernel function) into high-dimensional feature space.



The classification tree

A classification decision tree is built by partitioning the features to reduce class mixing at each split.

- at each iteration we select the partition in order to maximize the **information gain,** which measures how well the classes are separated by the split of set S into subsets S1 and S2 with n1 and n2 points.
- a stopping criterion -> a split does not improves the relative accuracy by at least alpha (complexity parameter).



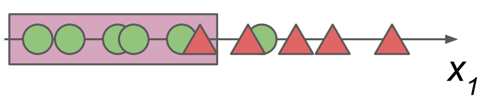
01_classifiers.ipynb

Curse of dimensionality

Usually -omics data are highly dimensional (e.g. 20.000 genes in the human genome = 20.000 dimensions!). In ML and in applied mathematics, the **curse of dimensionality** refers to the problem caused by the exponential increase in volume when we add extra dimensions to a mathematical space.

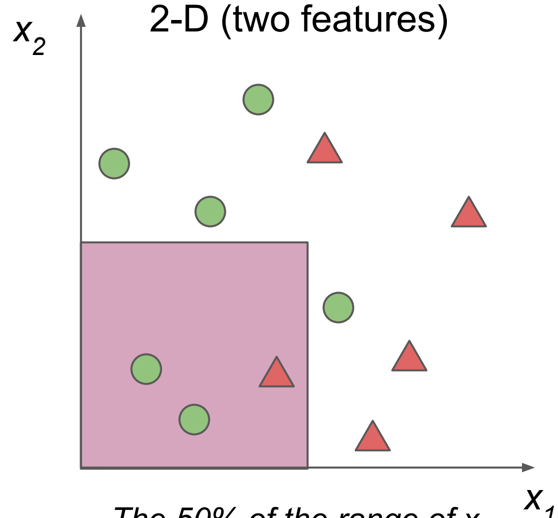
As the number of features (dimensions) grows, the amount of samples we need to build a "good" classifier grows exponentially.

1-D (one feature)



The 50% of the range of x_1 covers the 60% of samples.

2-D (two features)

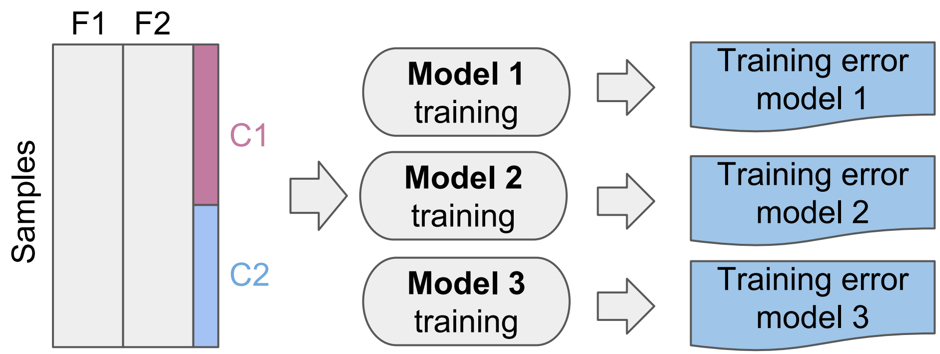


The 50% of the range of x_1 and x_2 cover the 30% of samples.

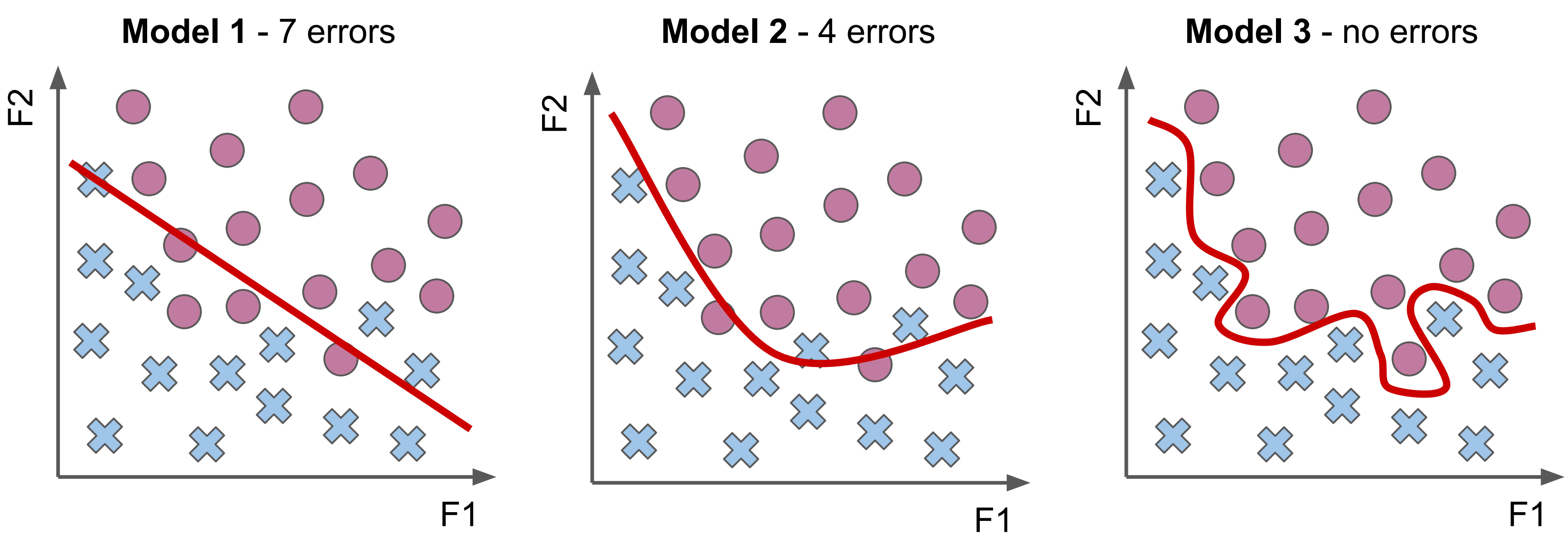
In general:

- more features -> performance of a classifier decreases
- increase in running time of the ML algorithms

Overfitting and underfitting

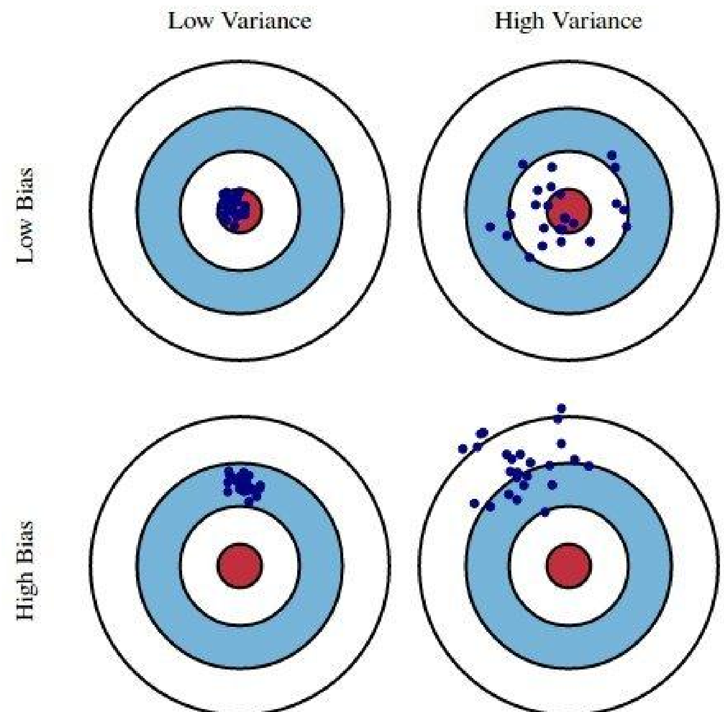


Which model is the best?

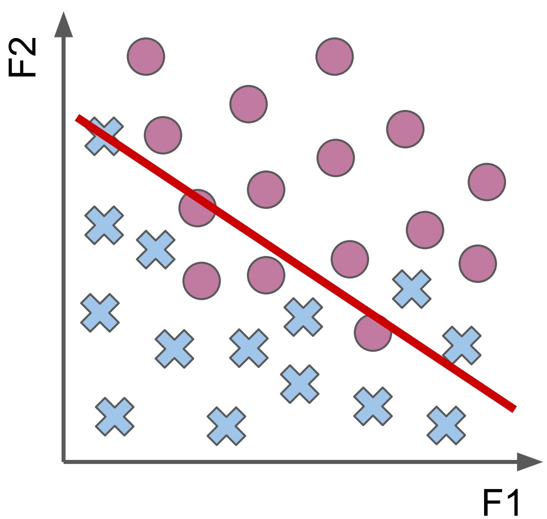


Overfitting and underfitting

We say that the model 1 **underfits** the data



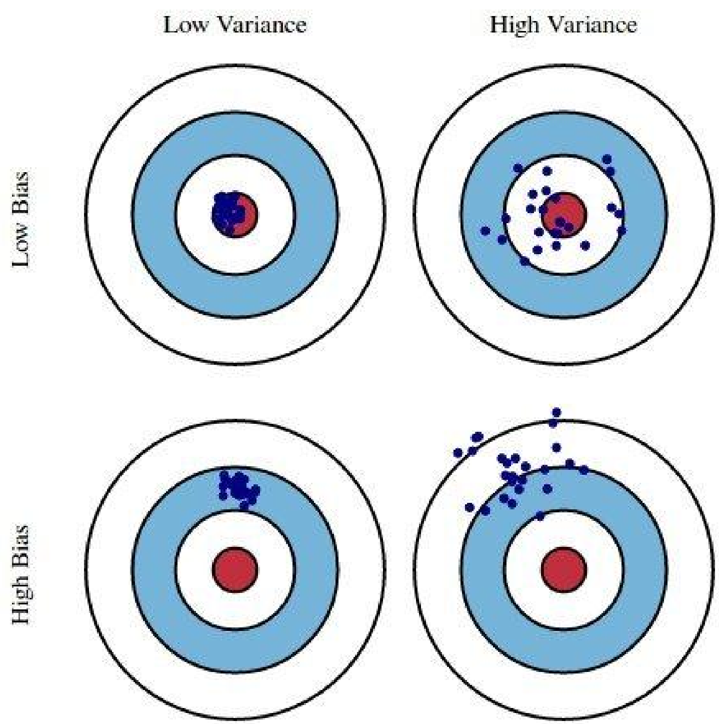
Model 1 - 7 errors



High bias: the model is incapable of capturing the underlying model.

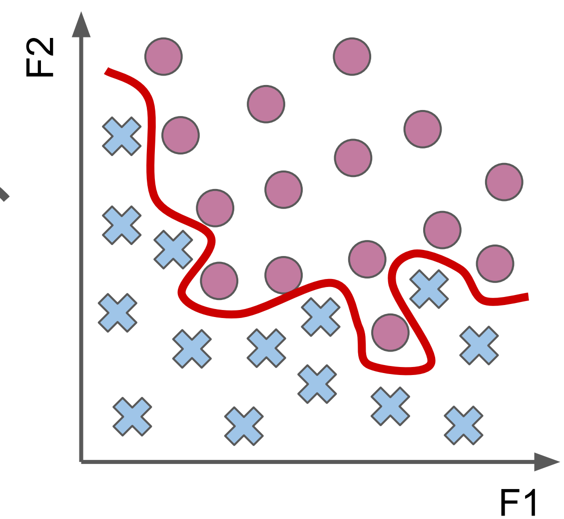
Low variance: the model is robust to noise.

Overfitting and underfitting



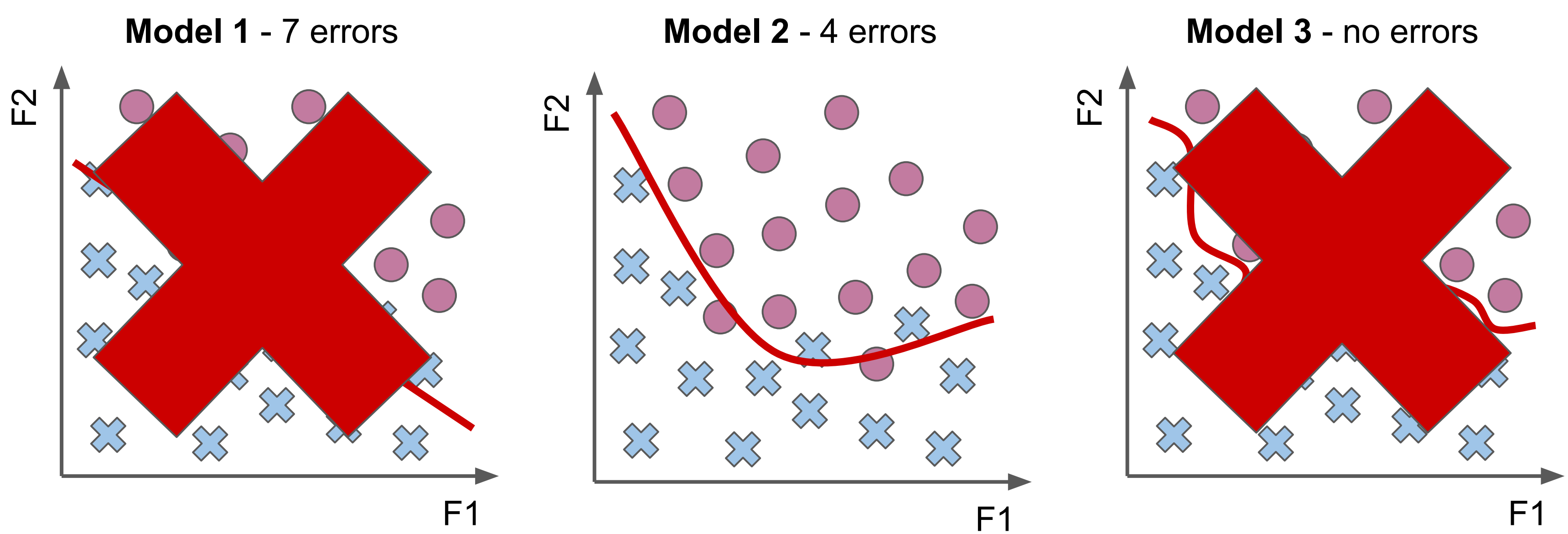
We say that the model 3 **overfits** the data

Model 3 - no errors



Overfitting and underfitting

We can say that the Model 2 has the best bias-variance tradeoff



How to limit overfitting?

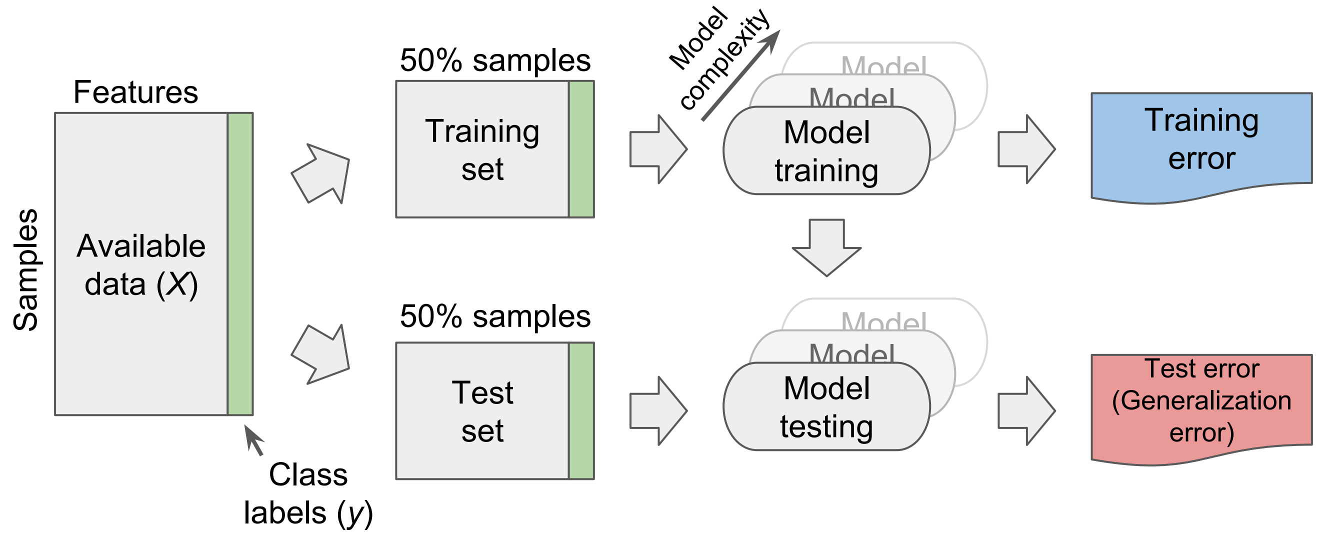
Under- and overfitting are common problems in both regression and classification. An appropriate level of complexity is needed to avoid both underfitting and overfitting.

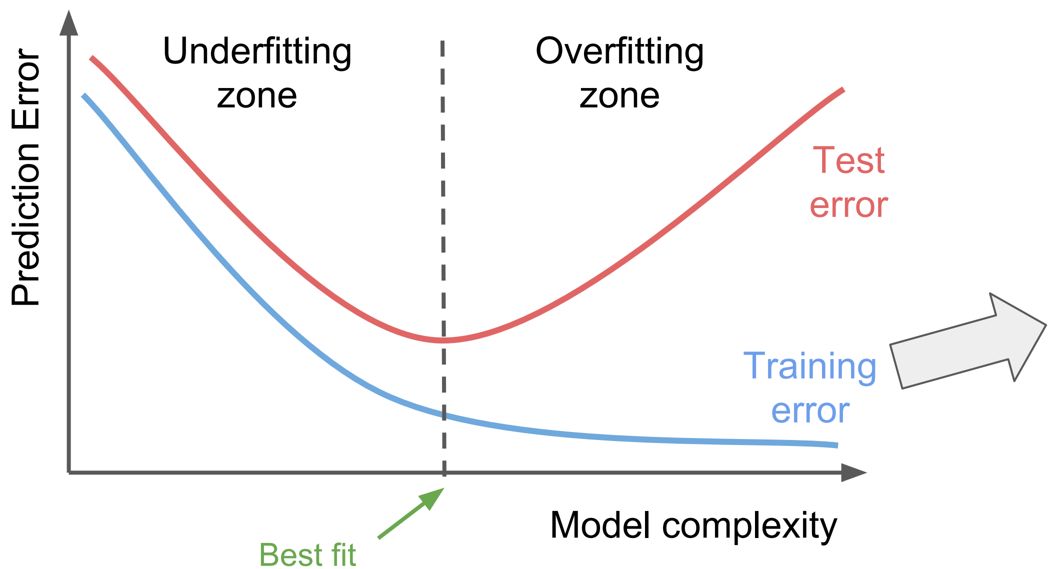
In turn, both **bias** and **variance** are affected by model complexity, which itself is a function of **model type** (e.g. linear, polynomial), **number of inputs** and **number and magnitude of parameters.**

How to limit overfitting:

- collect more samples;
- use ensembling methods that “average” models;
- choose simpler models / penalize complexity (e.g. regularization, a method that controls a model’s complexity by penalizing the magnitude of its parameters).

Generalization error





The goal of learning is to find a model which is able to **generalize**.

Training error does not provide a good estimate of how well the model will perform well on new unknown samples.

02_overfitting.ipynb

Model selection and assessment

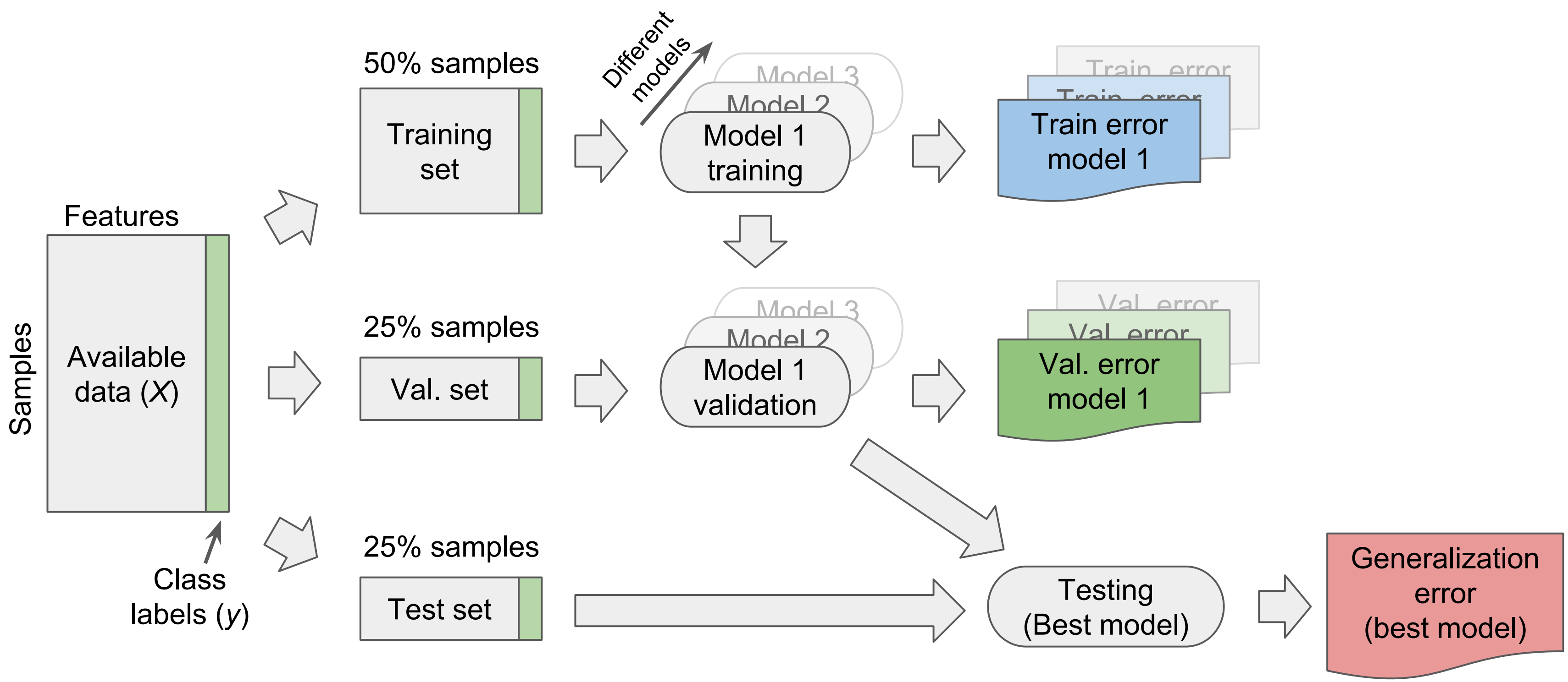
Note that there are in fact two separate goals:

- **Model selection**: estimating the performance of different models (e.g. different algorithms or different parameters of the same model type) in order to **choose the best one**;

- **Model assessment**: having chosen the best model, estimating its prediction error (**generalization error**) on new data.

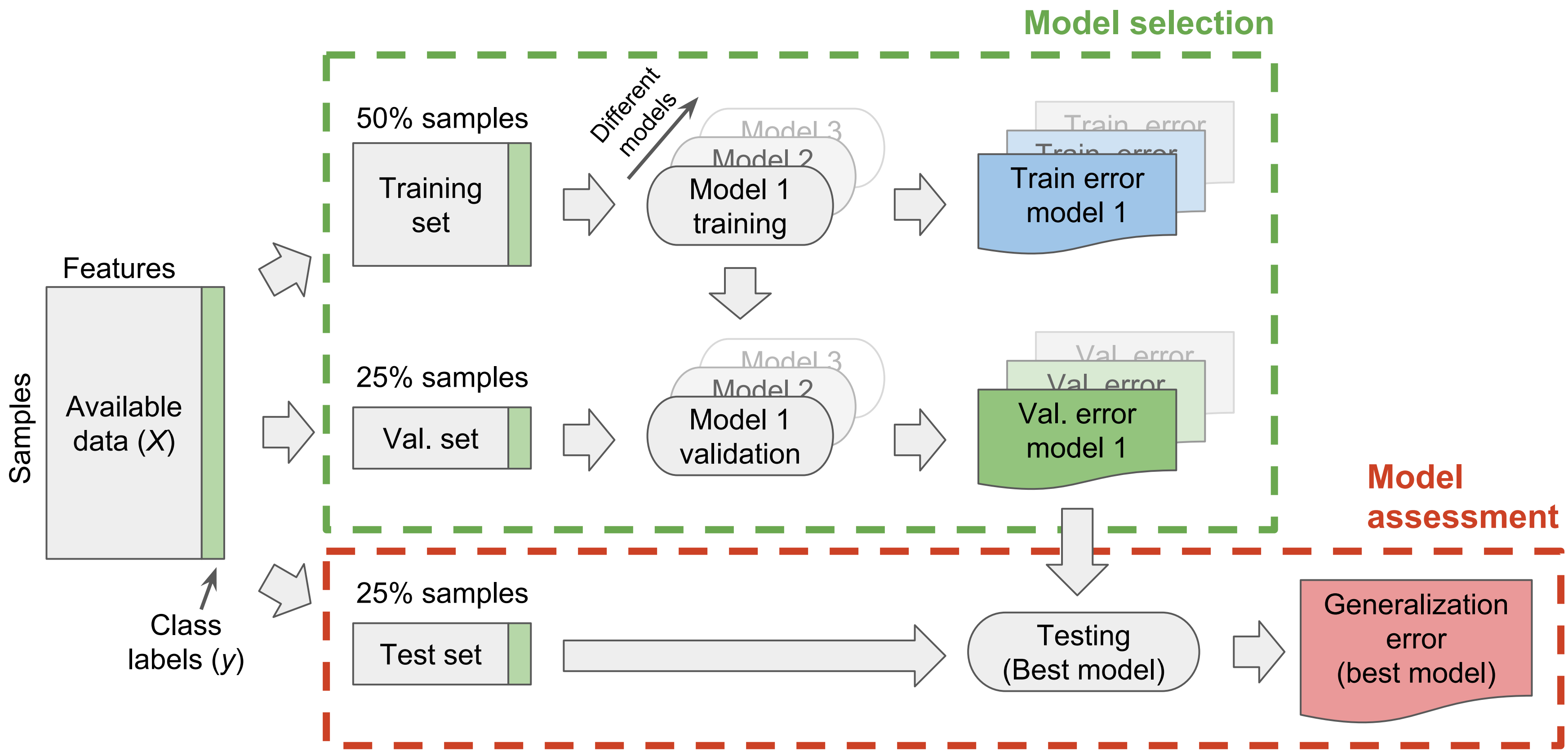
Model selection and assessment

If we have enough samples, the best approach for both problems is to randomly divide the dataset into three parts: a **training set**, a **validation set** and a **test set**.



Model selection and assessment

If we have enough samples, the best approach for both problems is to randomly divide the dataset into three parts: a **training set**, a **validation set** and a **test set**.



Model selection and assessment

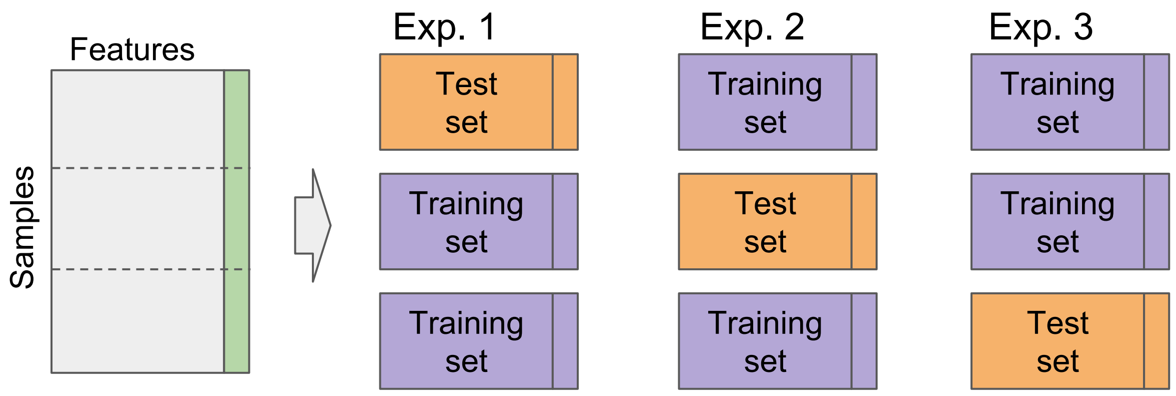
However, by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and **the results can depend on a particular random choice of training, validation and test sets**.

A solution to this problem is a procedure called **cross-validation** (**CV** for short).

Cross validation

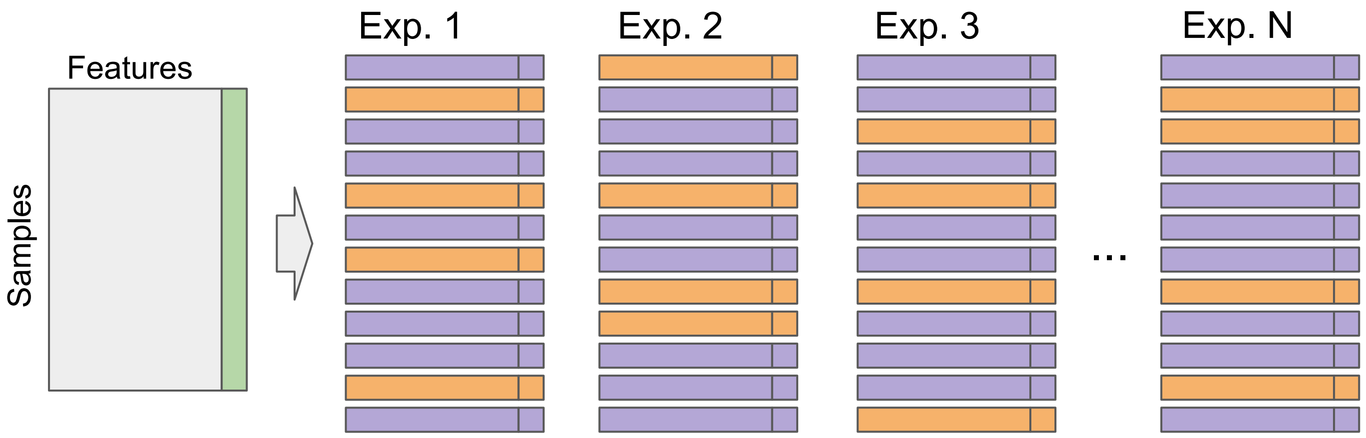
K-fold CV (3-fold CV here).

- k=1: **Leave-one-out (LOO) CV**



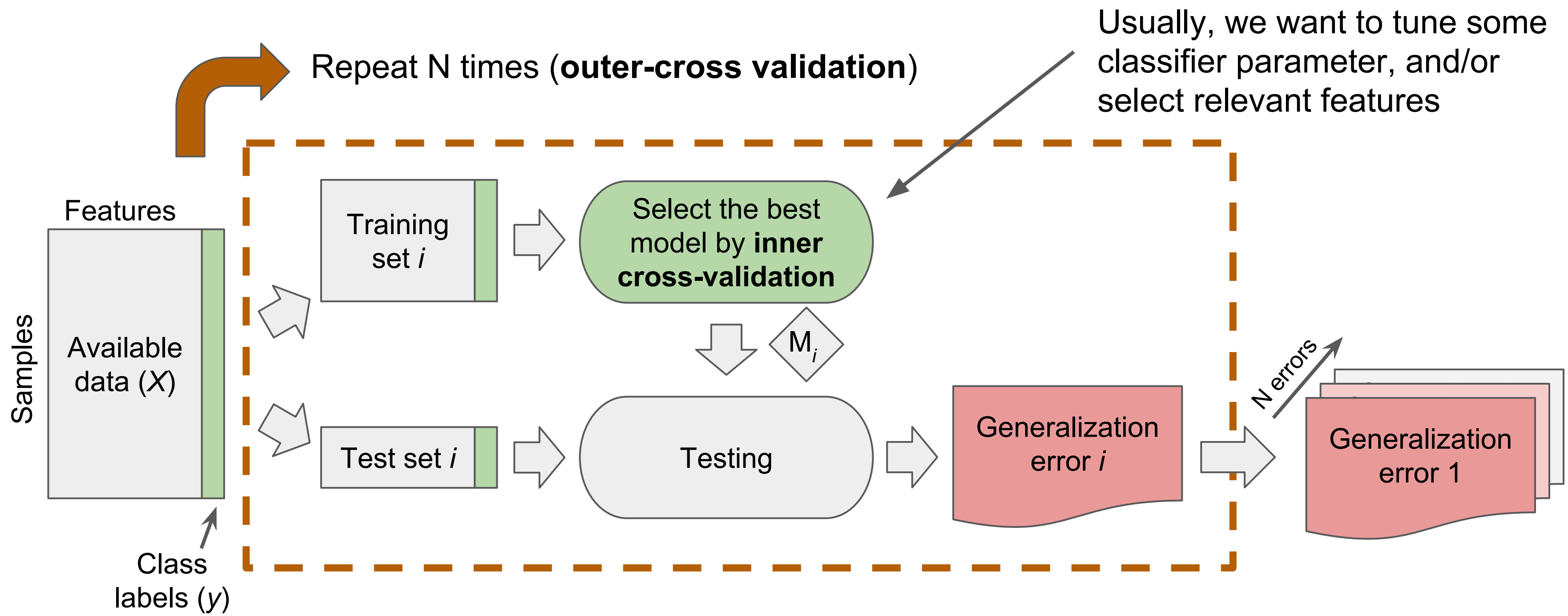
Random permutations CV
a.k.a. Shuffle & Split, random subsampling

- N iterations
- test-set 25%

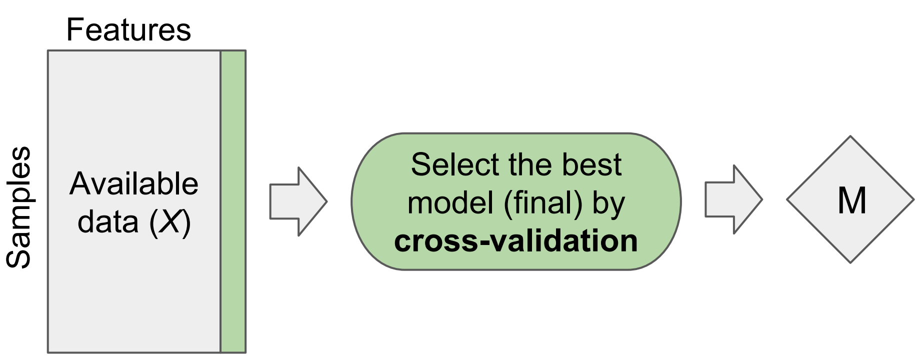


Model assessment and final model

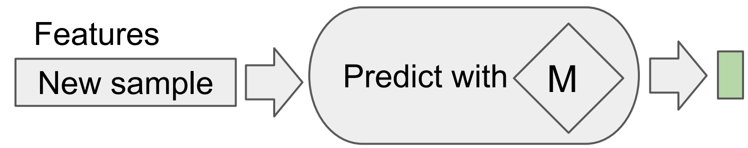
1. Model assessment



2. Final model



3. Predict on new samples



03_cross_val.ipynb