Model Validation and Selection

Dr. Umberto Michelucci
umberto.Michelucci@toelt.ai

Source

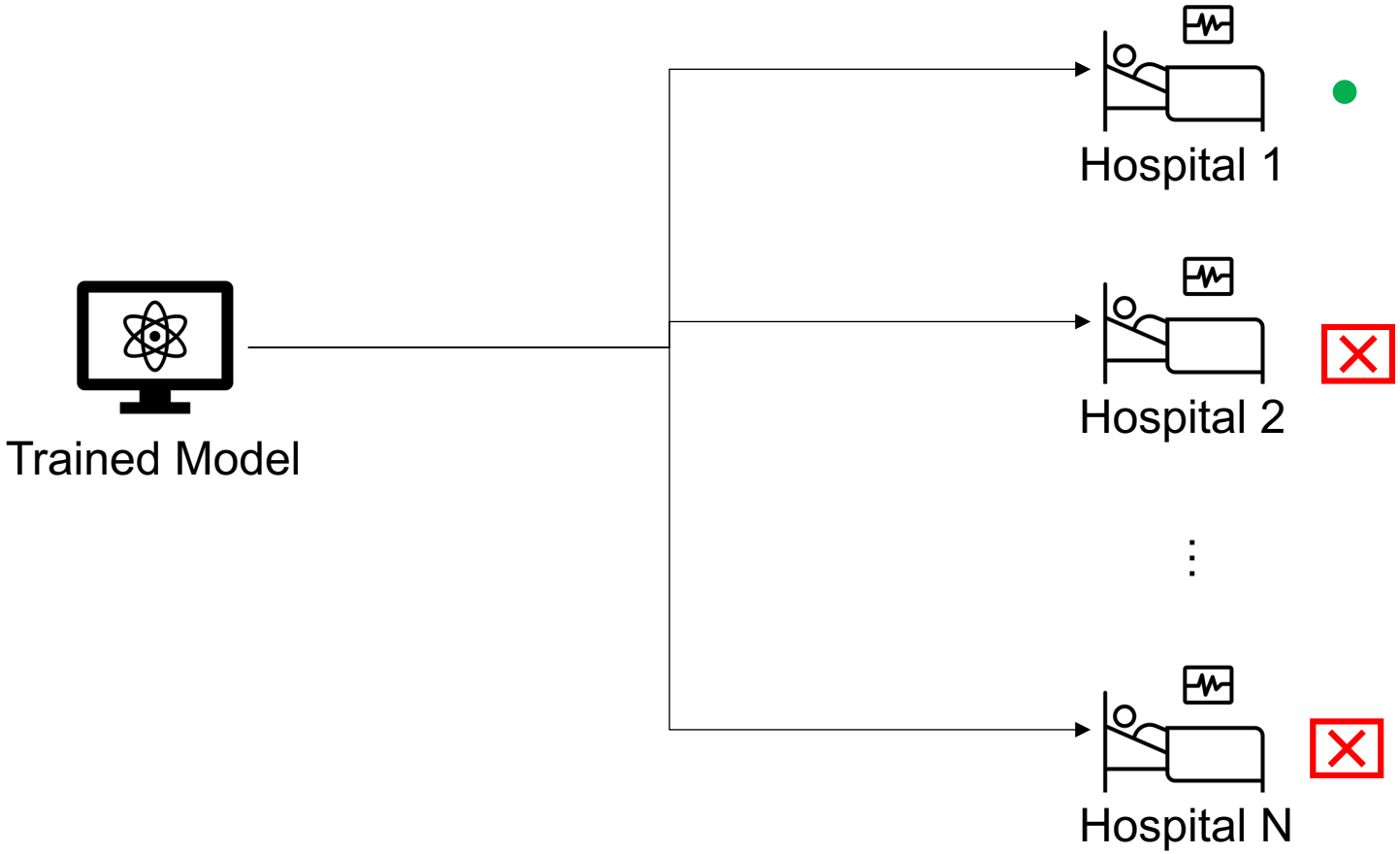
You will find in this presentation definition, figures, etc. with numbering that may seem not to make any sense. The reason is that they are taken from my upcoming book: *Fundamental Mathematical Concepts for Machine Learning in Science,* Umberto Michelucci, Springer Nature (available in 2024).

Model Validation

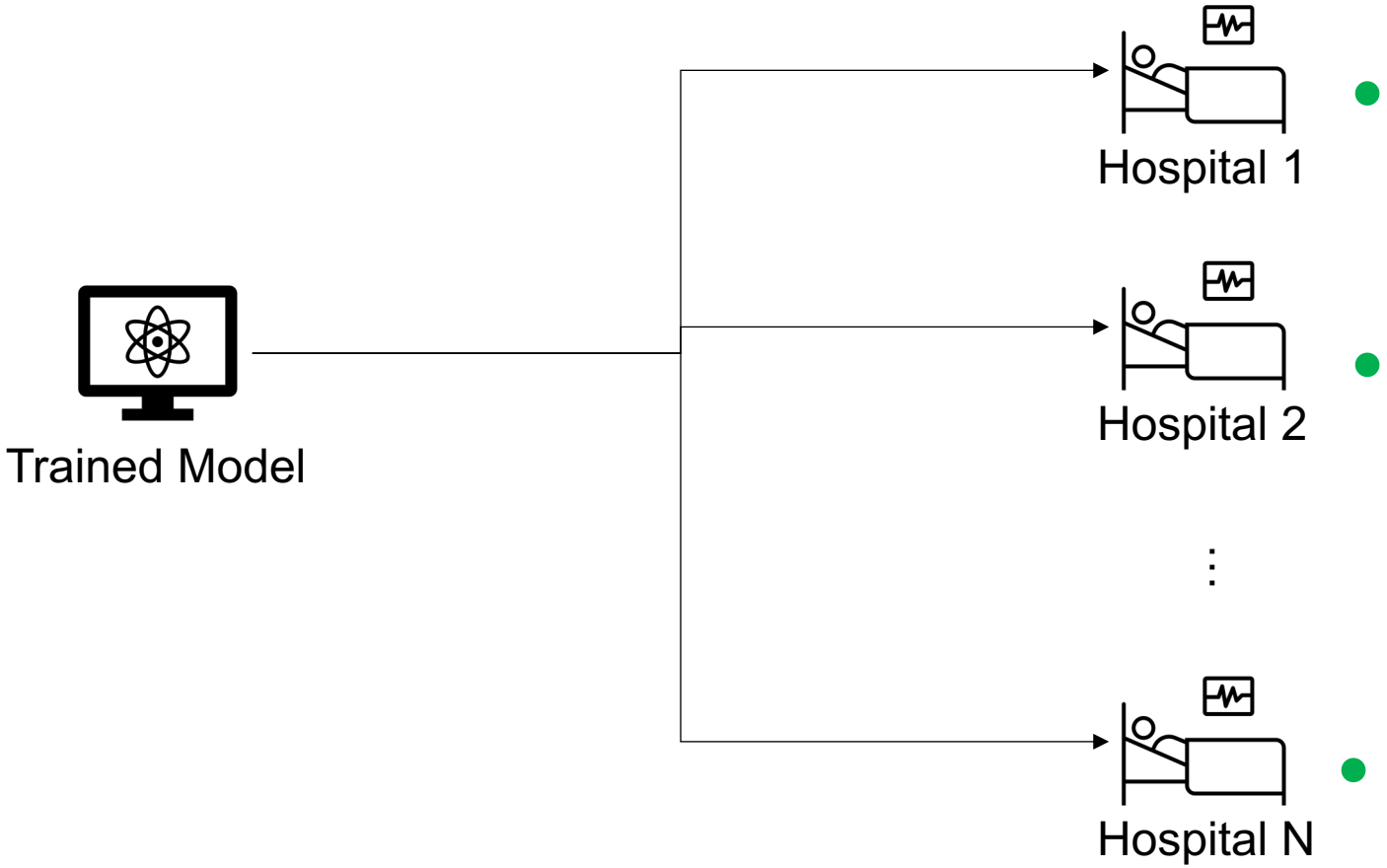
Model validation refers to **the process of confirming that the model actually achieves its intended purpose.**

Quiz: what is the intended purpose of a machine learning model?

Model Validation



Model Validation



Challenges

- You have often a very limited dataset at disposal
- You do not have the data from Hospital 2, 3, …, N

How can you check that the model is working with the data of all hospitals?

Or in “machine learning” terminology, how can you verify that your model generalise well?

Model Validation

Model validation refers to **the process of confirming that the model actually achieves its intended purpose.**

The intended purpose of a model is, in almost all cases, to be able to find patterns in data with an expected level of performance on unseen data (generalisation) statistically similar to the one used for training.

Why “validation” is important

Assessing Generalisation Properties: validation helps in evaluating how well a model performs on unseen data. This is essential for understanding the model’s ability to generalize beyond the training dataset (as we mentioned).

Detecting Overfitting: overfitting occurs when a model learns the noise and fluctuations in the training data to an extent that it negatively impacts the performance on new data. Validation helps in detecting overfitting by providing a check on the model’s performance on a separate dataset that was not used during training

Model Comparison: through validation, different models and their parameters can be compared and fine-tuned. This process allows for the selection of the best model and parameter set that offers the optimal balance between bias and variance.

Why “validation” is important

Trust: in practical applications, the validation process is crucial for stakeholders to trust the model. Demonstrating robust performance on validation data assures that the model is reliable and effective in real-world scenarios.

Regulatory Requirements: in certain industries, like finance and healthcare, model validation is not just a good practice but also a regulatory requirement to ensure that the models are fair, transparent, and reliable.

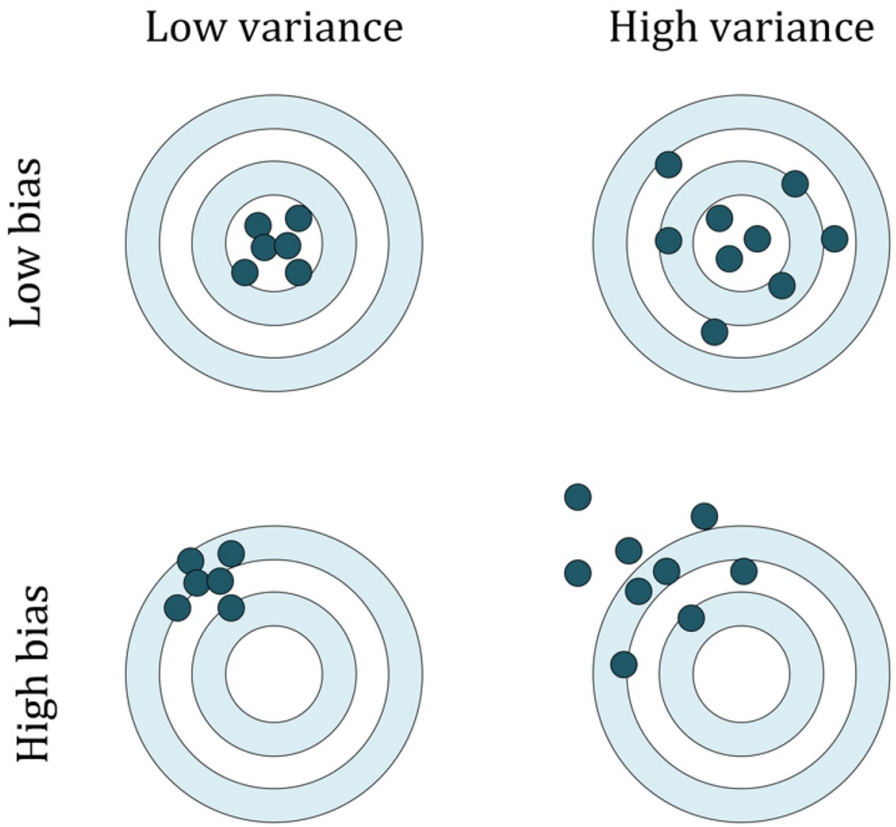
Why “validation” is important

Requirements: your model may need to satisfy specfic requirements as

- **Speed**
- **Energy Consumption**
- **Memory Consumption**

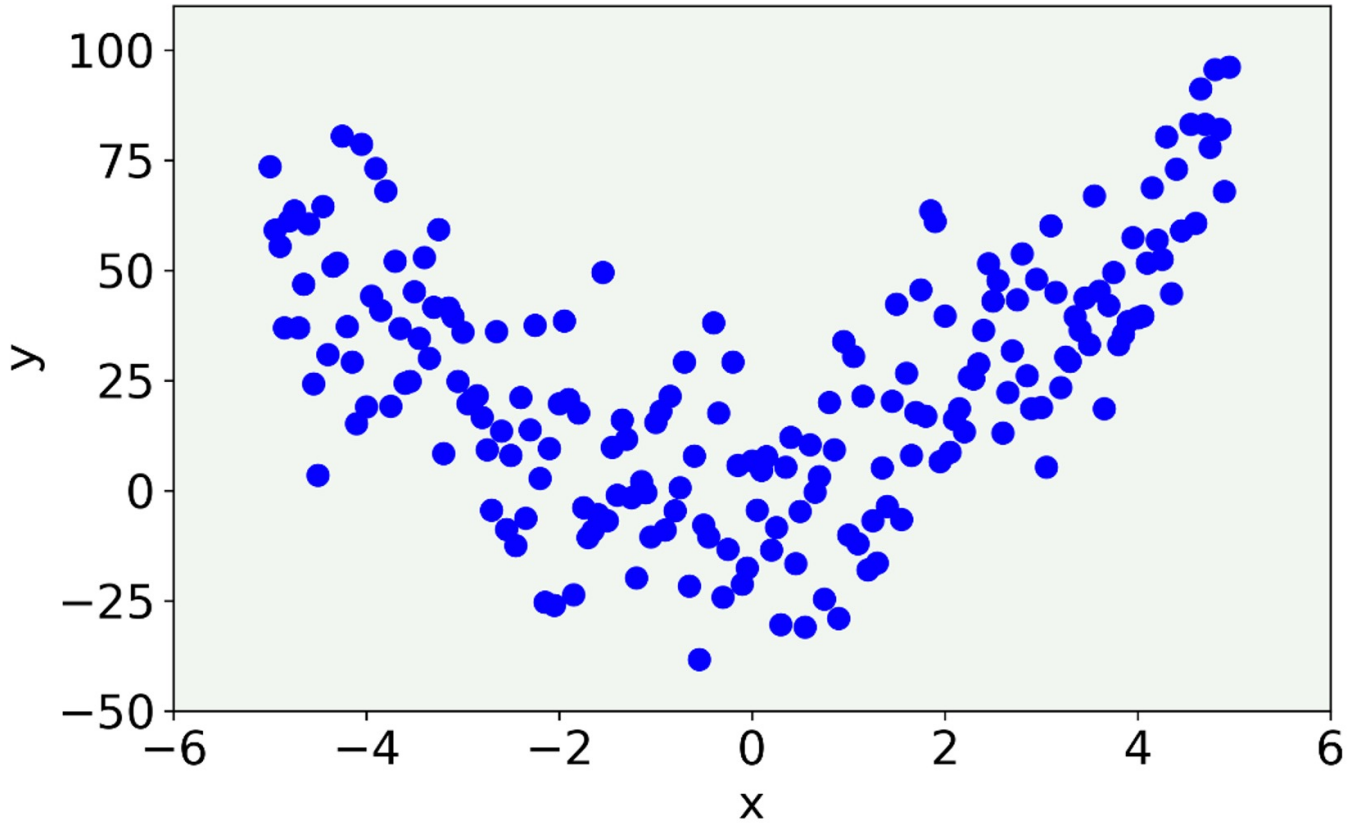
This is typical, for example, for applications for space.

Some terminology: variance and bias



U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

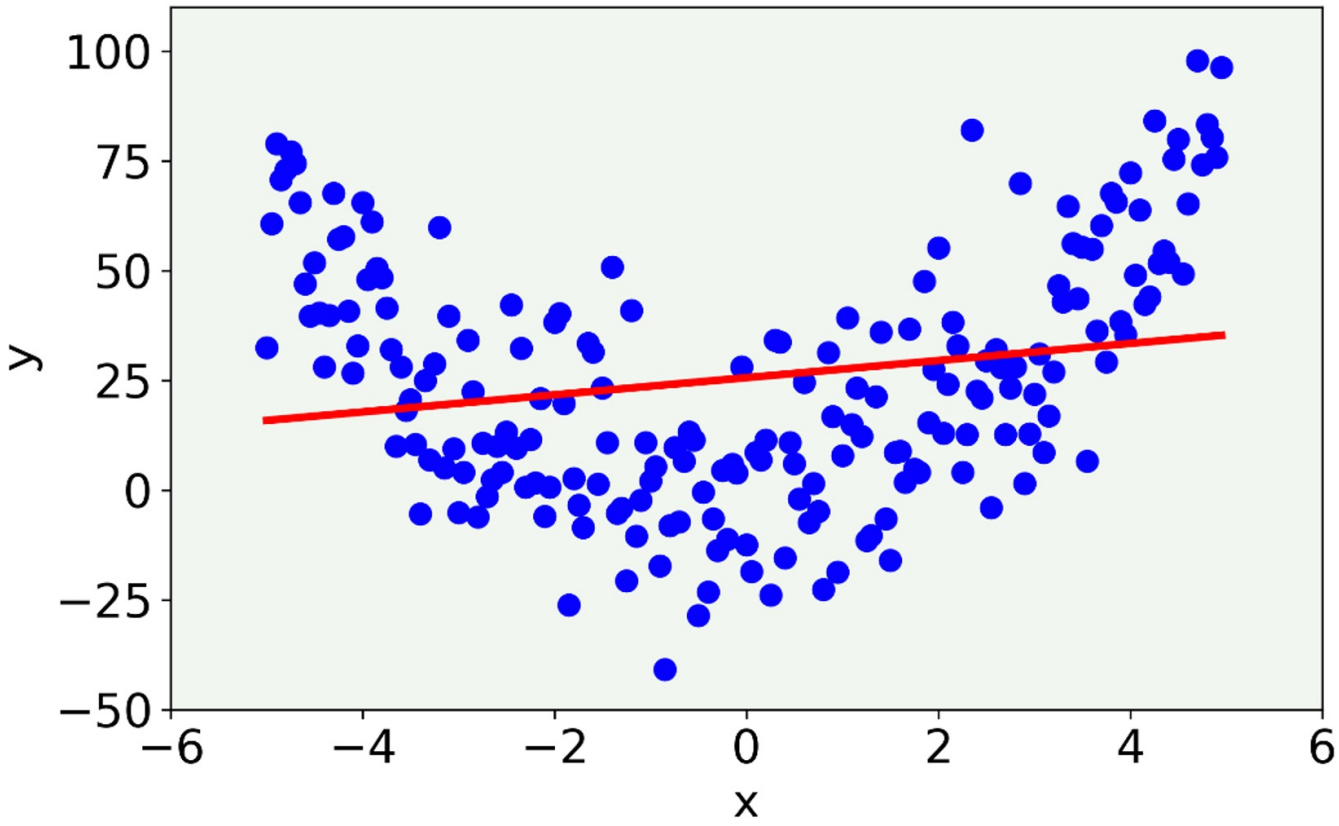
Overfitting I



Let us consider a set of data generated from a quadratic formula with some noise added.

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

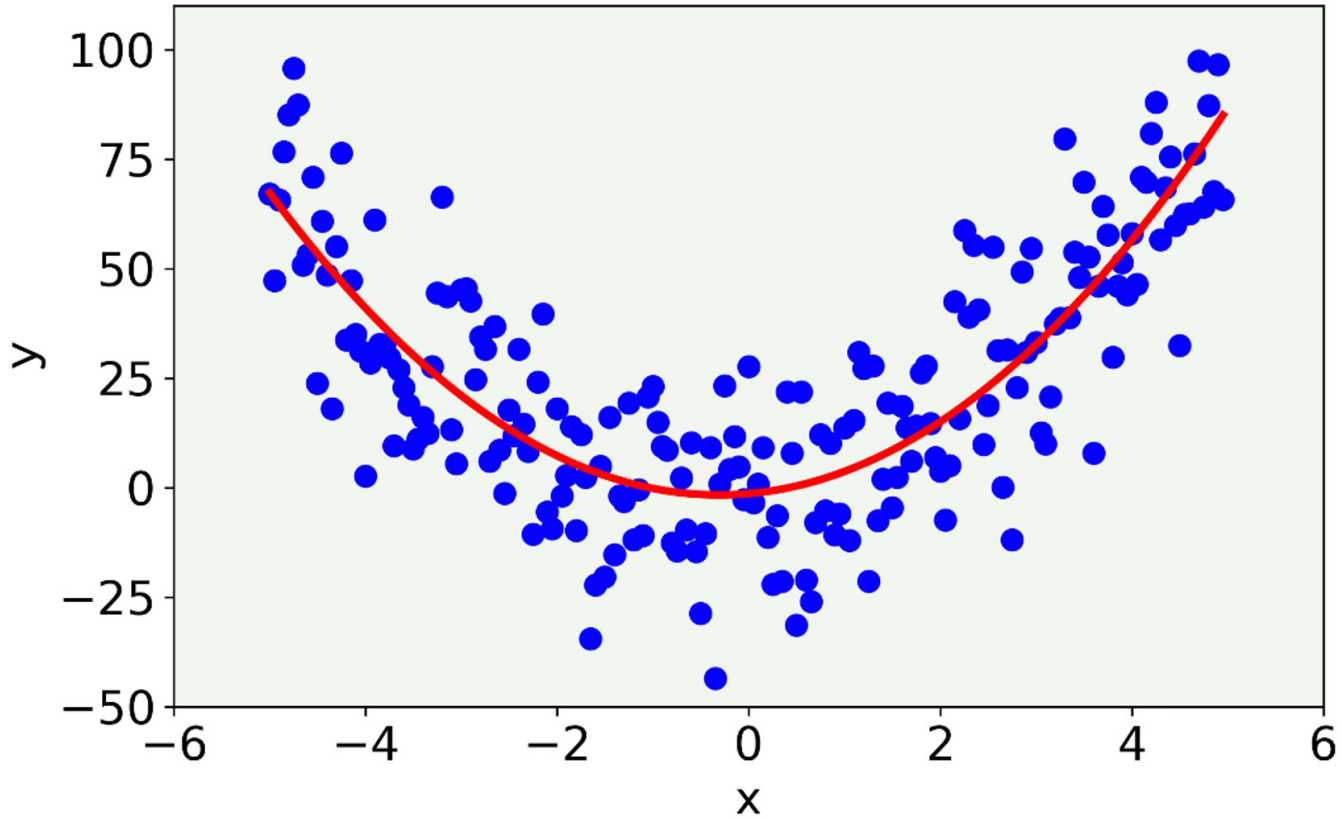
Overfitting II



The linear model misses the main features of the data being **too simple**. In this case the model is said to have **high bias**.

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

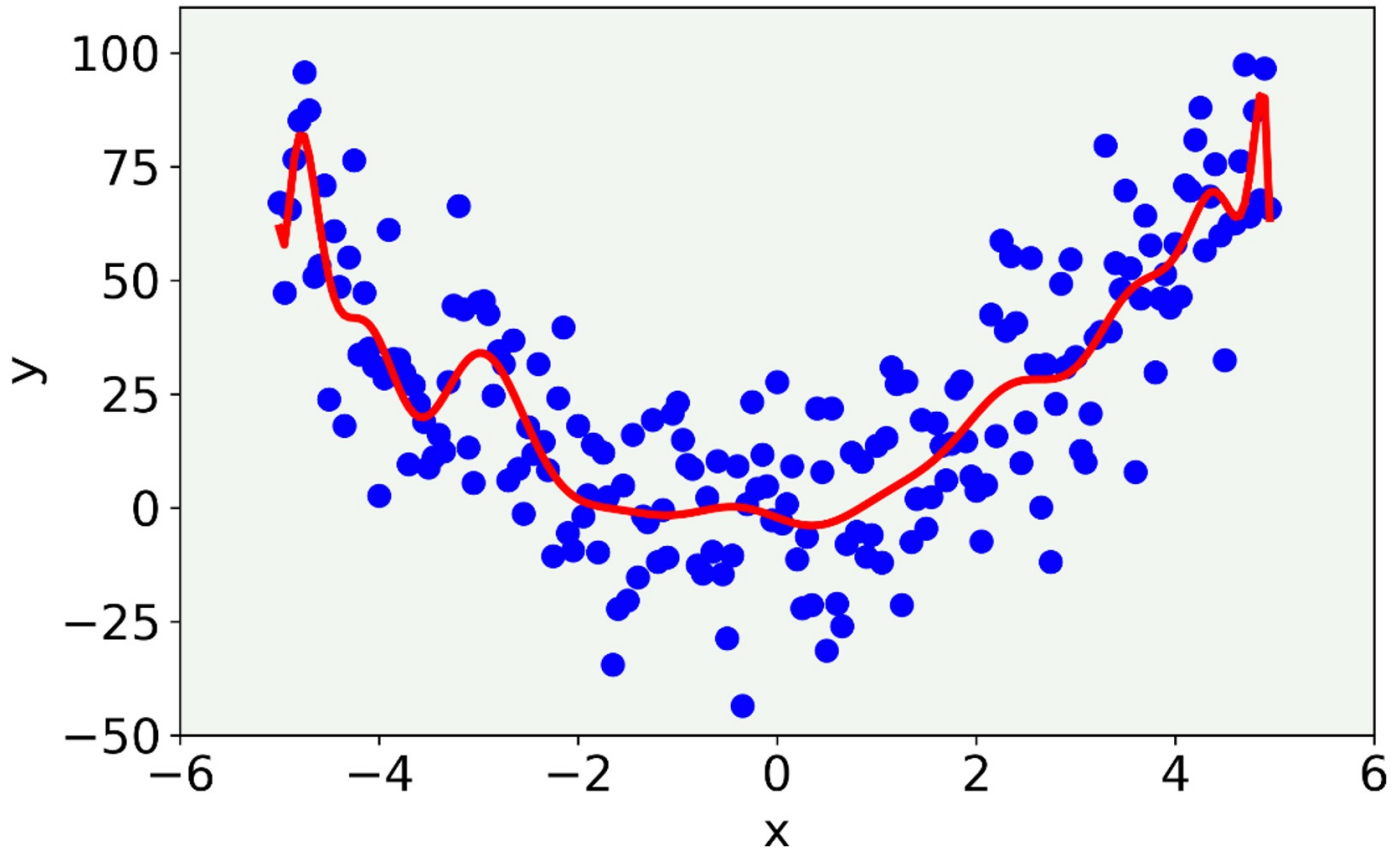
Overfitting III



The result (red line) for a 2-degree polynomial.

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

Overfitting IV

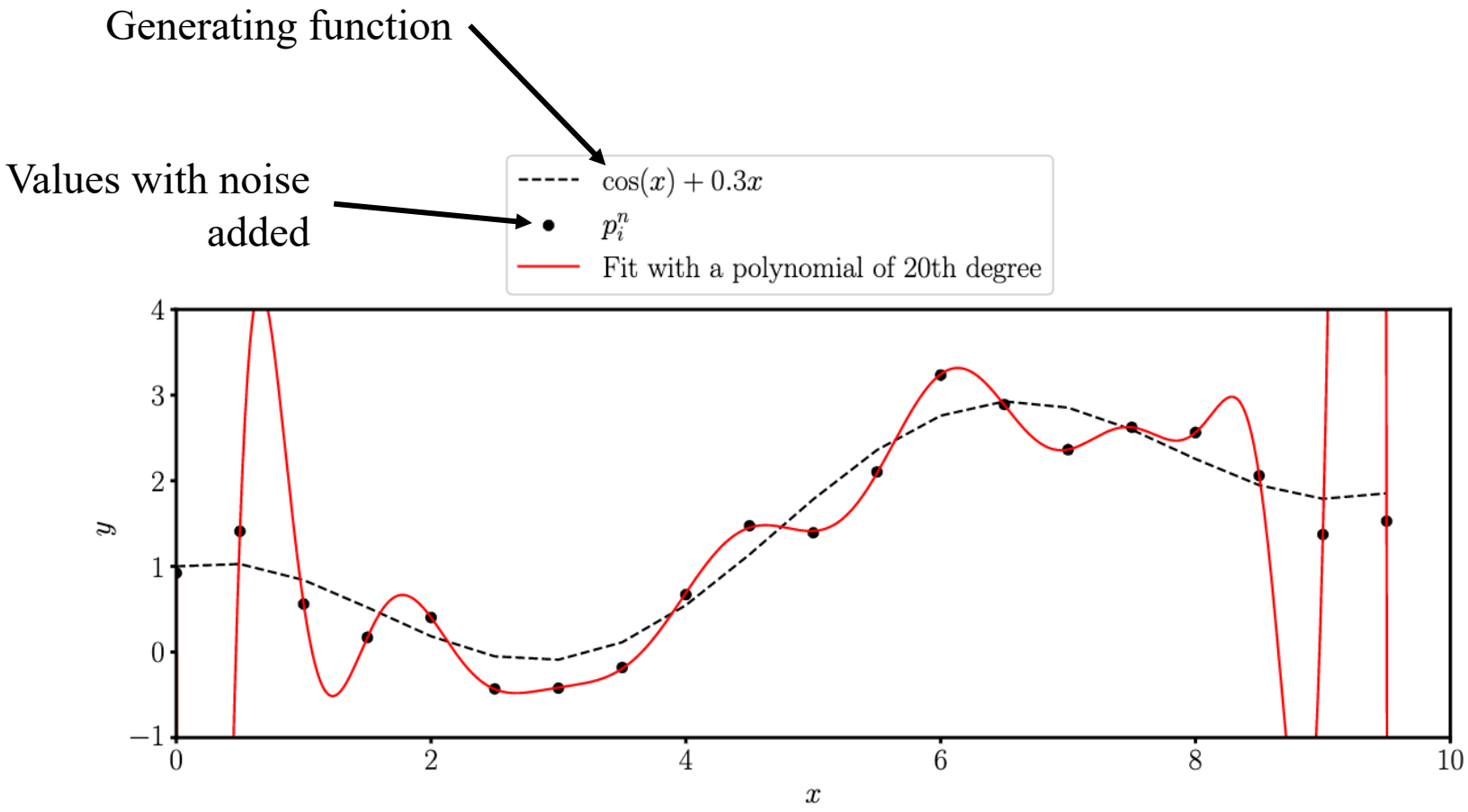


The result for a 21-degree polynomial model.

Here we talk about **overfitting**, since the model capture the noise present in the data. The model is **too complex**.

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

Another Example – model too complex



Overfitting

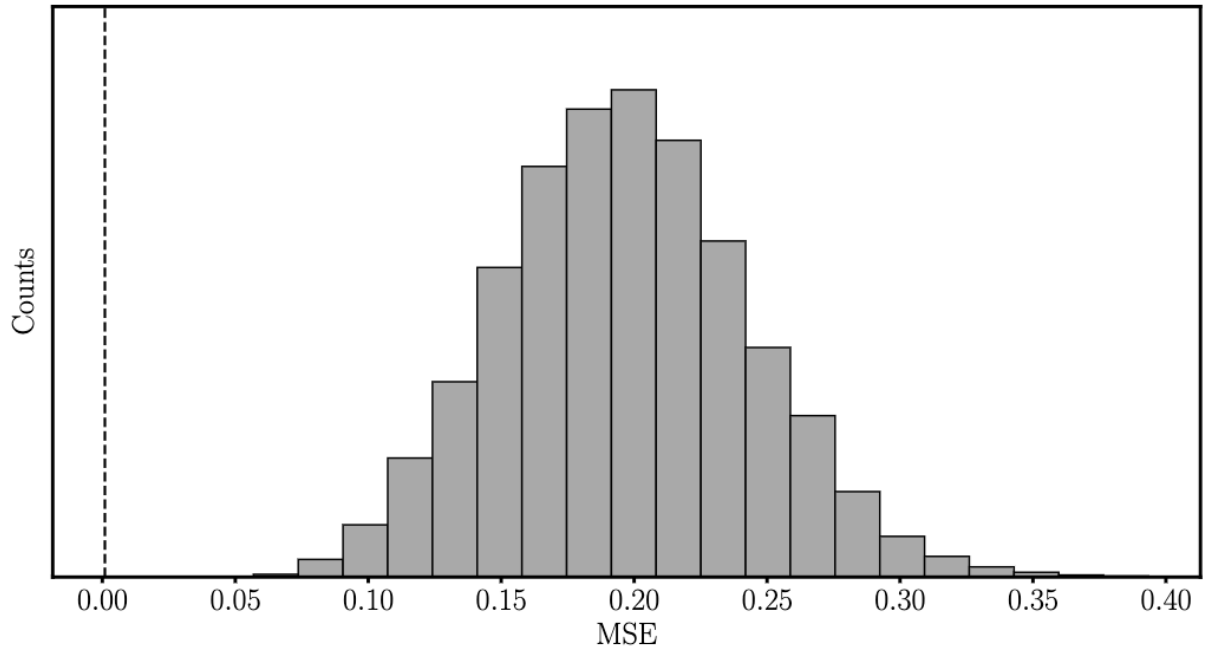
A model may, due to its flexibility, learn patterns that are due to noise, errors or simply wrong data.

How can we detect it? How can we visualise its occurrence?

When overfitting occurs, the model evaluated on unseen data will have a bad performance.

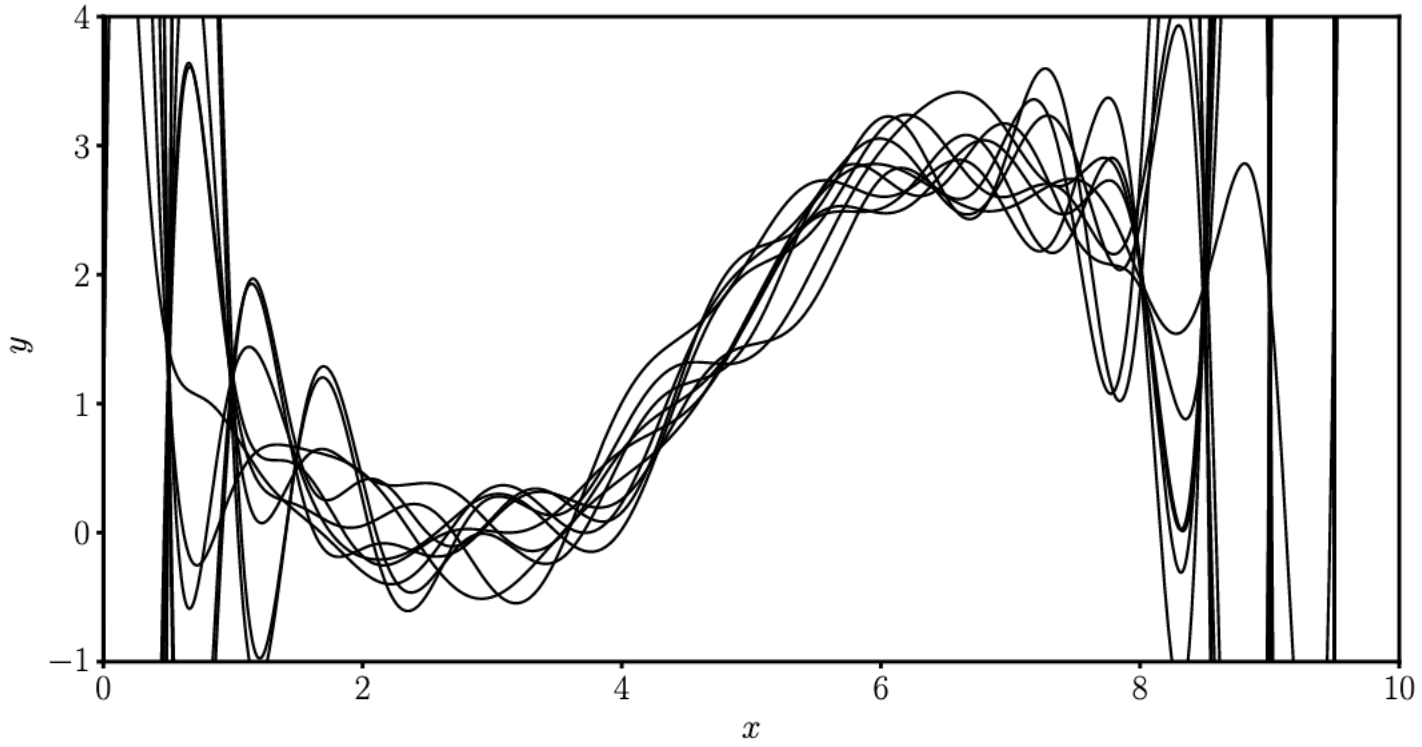
Another Example – model too complex

Distribution of the MSE obtained by fitting a 20^{th} polynomial to one dataset, and then evaluating the MSE between the polynomial and 10000 datasets obtained by generating each time new random noise. The dashed vertical line is the MSE evaluated on the training dataset.



Another Example – model too complex

The result of fitting a 20^{th} polynomial to multiple dataset of points obtained by generating each time different random noise.
This regine is called: **HIGH VARIANCE**



Overfitting VI

In 99.9999% of cases a visualisation is not possible, so how can we detect it?

→ To detect it we need to test our model on unseen data. To achieve this, we need to split our dataset in two portions.

Definition of Cross-Validation

Cross-Validation (CV)

Cross-validation is a statistical method used to estimate the skill of machine learning models. It is primarily used to assess how the results of a predictive model will generalize to an independent data set.

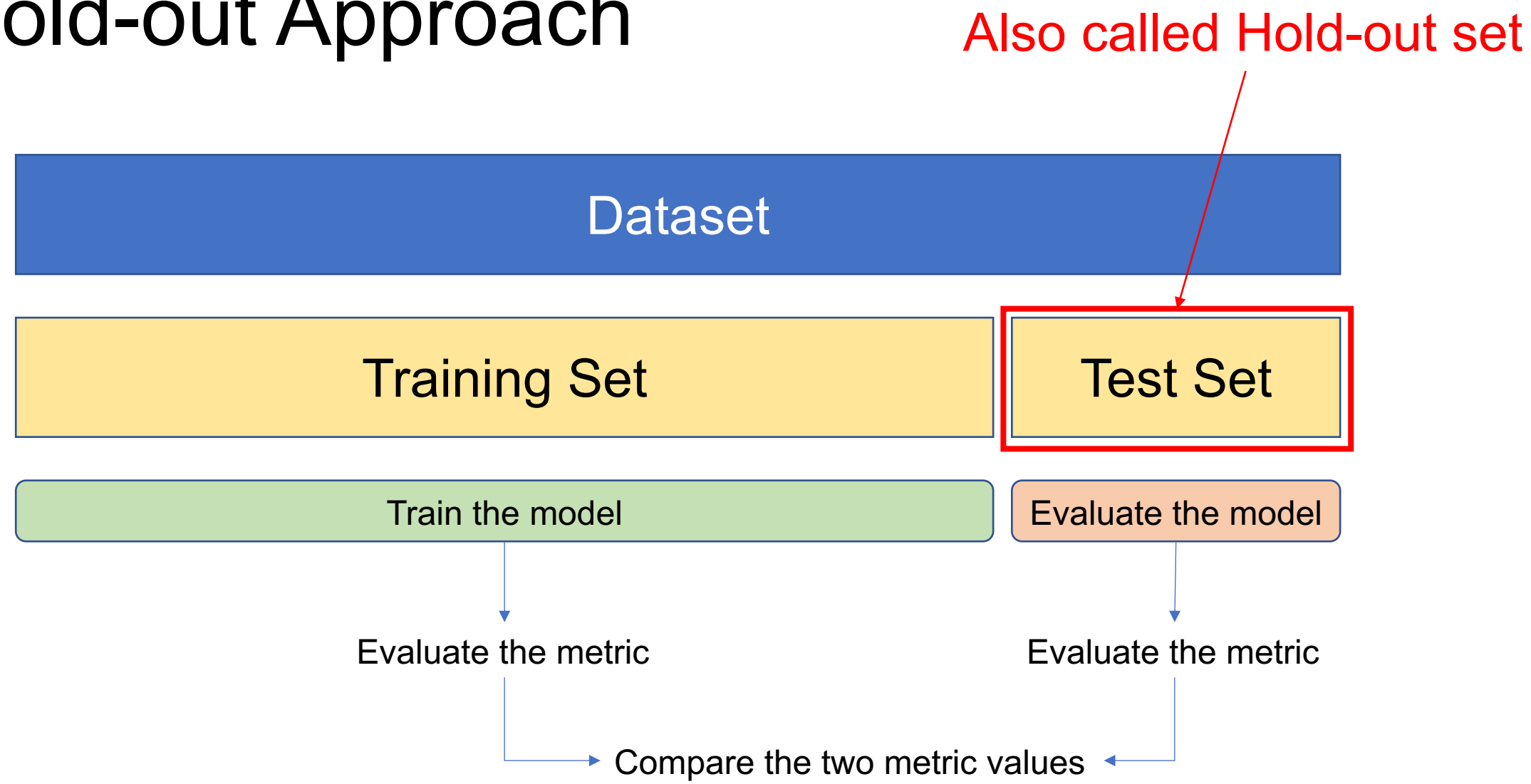
The basic idea is to divide the total data into multiple subsets or "folds." The model is then trained on a subset of the folds (training set) and tested on the remaining fold (validation set). This process is repeated multiple times, with each fold being used once as the validation set.

Model Validation - Methods

- Hold-out Approach
- Monte-Carlo Cross Validation
- K-Fold Cross Validation
- Leave-one-out Cross Validation
- y-Randomisation

Hold-out Approach

Hold-out Approach



Hold-out Approach

If the *train* and *test* value of (for example) the error is similar → the model generalise well (kind of, at least it behaves similary on the two sets)

If the *train* and *test* value of (for example) the error is different → the model generalise badly (kind of, it behaves differently on the two sets)

Error	Case A	Case B	Case C	Case D
Train set error	1%	15%	14%	0.3%
Dev set error	11%	16%	32%	1.1%

- **Case A**: here we are overfitting (high variance), because we are doing very well on the training set, but our model generalizes very badly to our dev set
- **Case B**: here we see a problem with high bias, meaning that our model is not doing very well generally, on both datasets
- **Case C**: here we have a high bias (the model cannot predict very well the training set) and high variance (the model does not generalize well on the dev set).
- **Case D**: here everything seems ok. Good error on the train set and good on the dev set. That is a good candidate for our best model.

Bias-Variance Trade-off

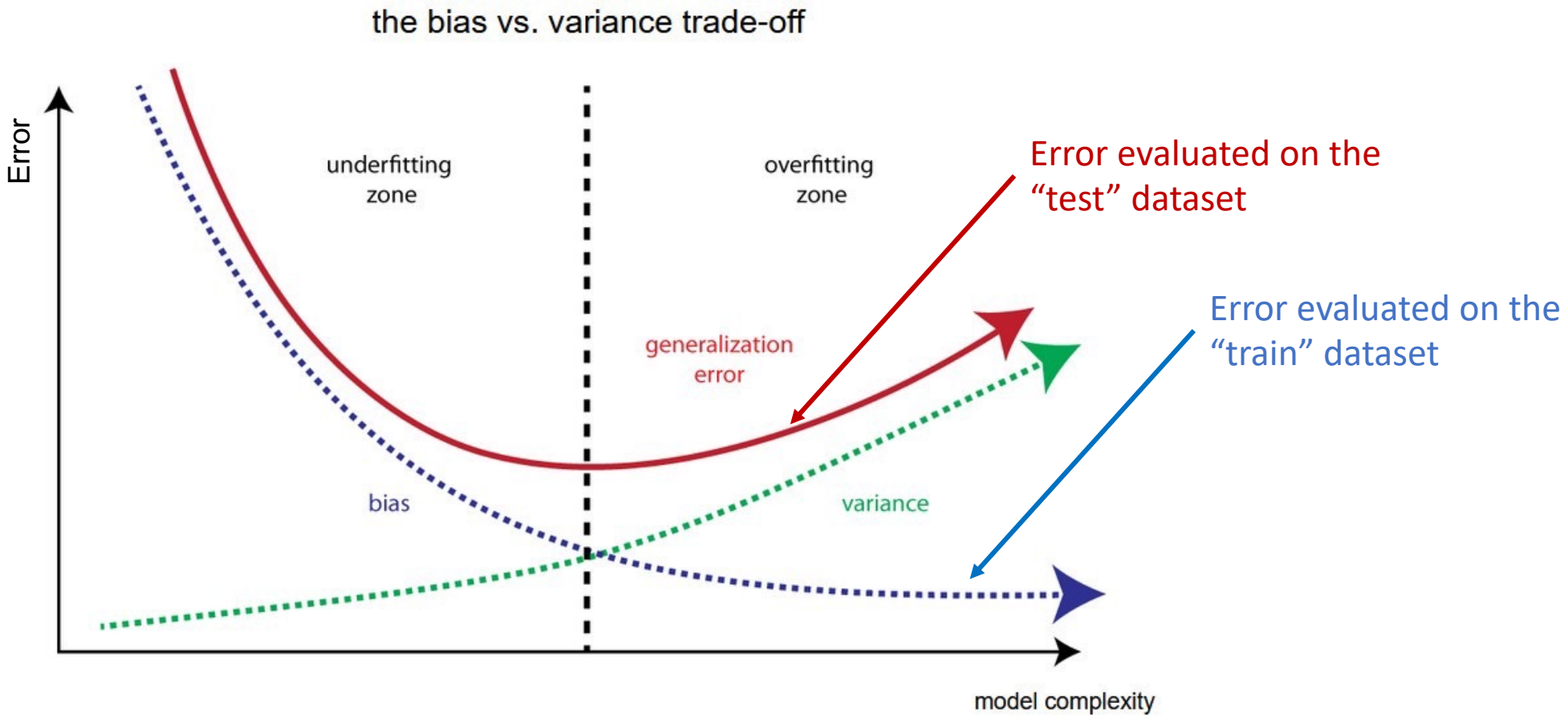


Image © Towarddatascience.com

Essence of overfitting

Note The essence of **overfitting** is to have unknowingly extracted some of the residual variation (i.e. the noise) as if that variation represented underlying model structure. The opposite is called **underfitting** when the model cannot capture the structure of the data

Effect of randomness in the data – Hands-on Experiment

Code in GitHub

Under ”Day 1/04 - Model Validation/code"

Stratified Sampling

When using the hold-out approach you should look out for situation where by splitting the dataset you end up with a D_T that is at all not representative of the original dataset (or the original research question).

Hold-out approach – a possible problem

1. Consider a dataset with 1000 inputs x_i with $i = 1, \dots, 1000$ and 1000 labels y_i where $y_i = \{0,1\}$ (imagine a supervised binary classification problem).
2. Imagine that $y_i = 0$ for $i = 1, \dots, 500$ and $y_i = 1$ for $i = 501, \dots, 1000$.
3. Now imagine to split the dataset in two parts:
   1. Training: $i = 1, \dots, 800$
   2. Test: $i = 801, \dots, 1000$
4. The test dataset will only have class 1, and thus it is impossible to assess the model performance sensibly

Hold-out approach – a solution (stratified sampling)

The solution to the previous problem is the so-called «stratified sampling».

In general **stratified sampling** is a method of sampling from a population (in our example, the original dataset D) which is partitioned into sub-populations (in a classification problem the sub-populations can be the classes)

You split the dataset in such a way that each portion has the same proportions of the classes (for example by shuffling the data before splitting)

Splitting done right

In general you should split the dataset in such a way that each portion is representative of the entire dataset (or the research question you are trying to solve).

Data Leakage

What is data leakage

Data leakage in machine learning and data science refers to a situation in which information from outside the training dataset (for example from the test dataset) is inadvertently used to create or train the model.

This leakage of information typically results in a model that performs unrealistically well on the training data, but poorly on unseen, real-world data.

Types of data leakage

Preprocessing Leakage: If data preprocessing (like normalisation, feature selection) uses the entire dataset rather than just the training set, information from the test set can leak into the model.

Model Selection Leakage: When model selection or hyperparameter tuning is done using the test set, it leads to an overfitting on the test data, providing a misleadingly high performance estimate (more on that later).

Temporal Data Leakage: In time-series data, using future data in the training process leads to leakage as the model gets access to information it would not have in a real-world scenario.

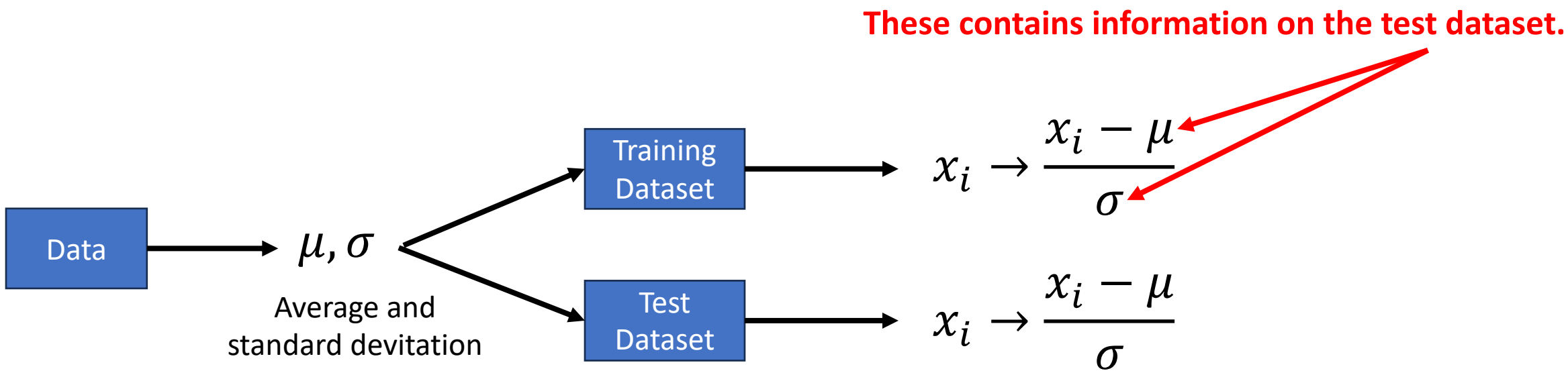
How to deal with data leakage

Preprocessing within Splits: ensure all preprocessing steps are conducted within each dataset portion (for example in the hold-out approach in the training and test portions). This means applying transformations after the split and separately to each portion.

Do not use any test data: do not use any information from the test dataset (as averages, standard deviations, missing data or imputation approaches) on the training dataset.

Data Leakage
Example of Zero Average-Standard Deviation of one

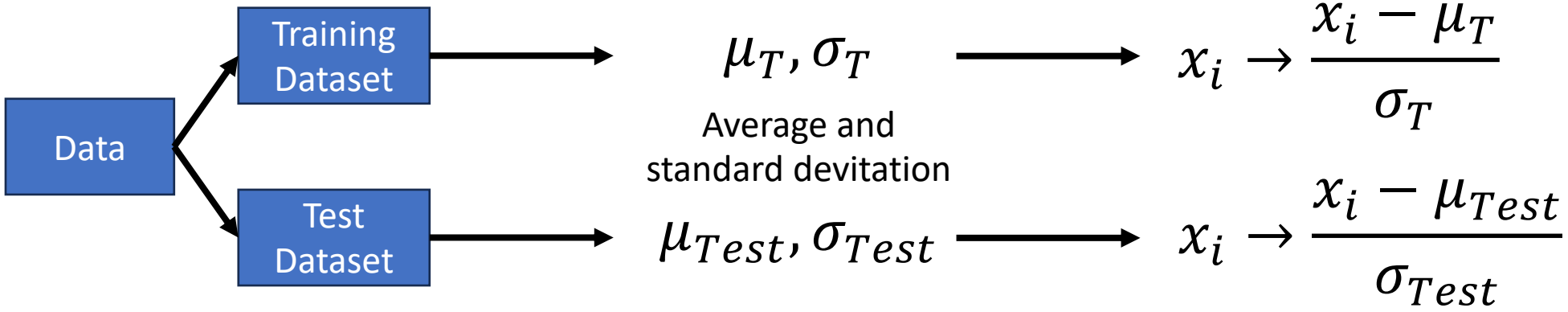
Normalise the data to have an **average of zero** and a **standard deviation of one**.



Data Leakage

Preprocessing within splits
Example of Zero Average-Standard Deviation of one

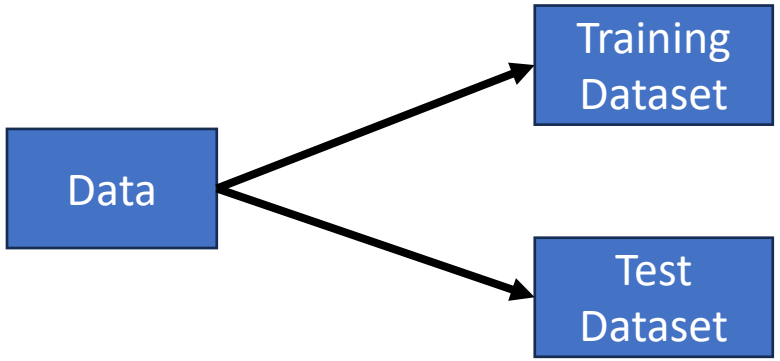
Normalise the data to have an **average of zero** and a **standard deviation of one**.



HUGE ASSUMPTION: the split has been done so that the two datasets are ”statistically similar”.

Data-agnostic Preprocessing
Example of Zero Average-Standard Deviation of one

Normalise the data to have an **average of zero** and a **standard deviation of one**.



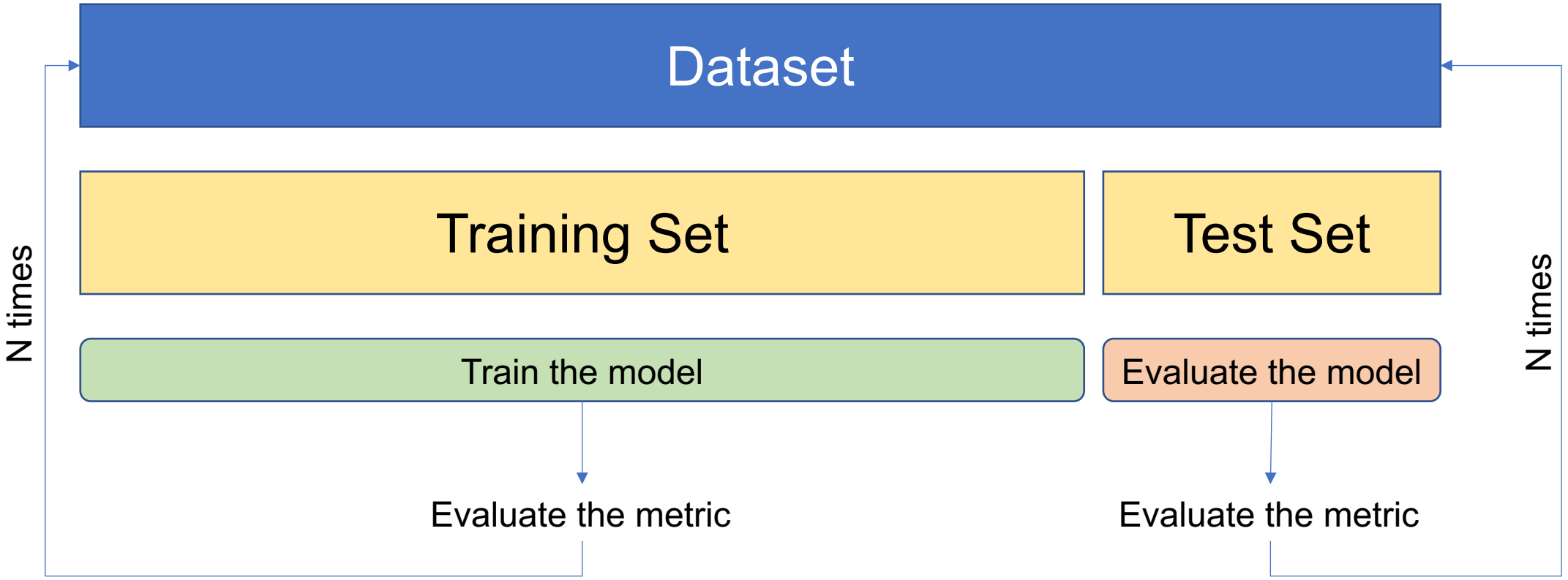
$$x_i \rightarrow \frac{x_i - \mu_P}{\sigma_P}$$

$$x_i \rightarrow \frac{x_i - \mu_P}{\sigma_P}$$

μ_P and σ_T are determined by previous knowledge about the data and problem (and not determined by the data). For example one may take the average of age (hypothetical feature) from available statistics.

Monte-Carlo Cross Validation

Hold-out Approach with multiple splits (Monte Carlo Cross-Validation)



Monte-Carlo CV

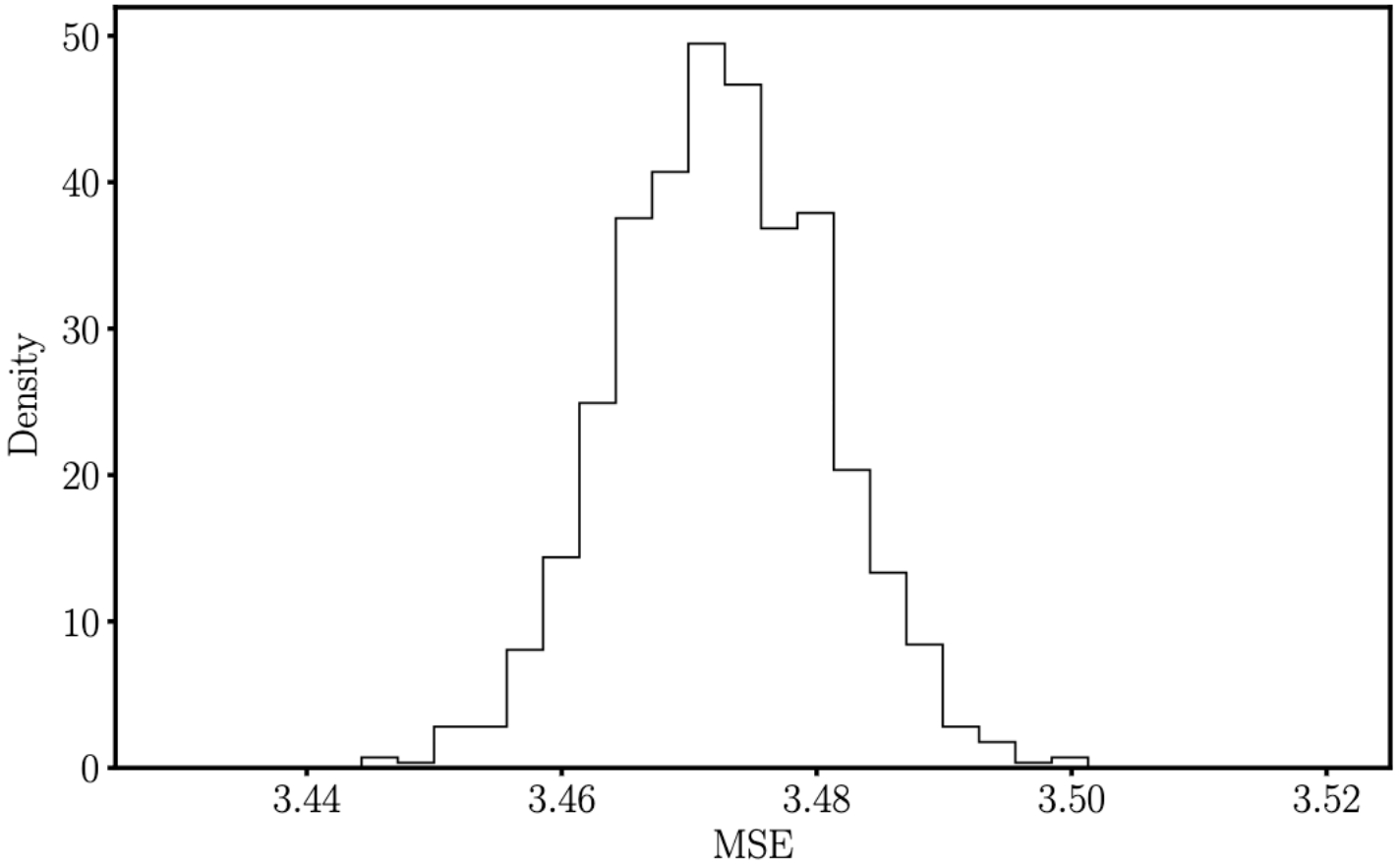
Algorithm 1 Monte Carlo Cross-Validation approach Algorithm Study

set N to some value
$j \leftarrow 1$
p is the percentage of data used for validation.
while $j < N$ **do**
    Split the dataset in two parts: $D_T^{[j]}$ and $D_V^{[j]}$ in proportions $1 - p$ and p for training and validation respectively
    Train a linear regression model $M^{[j]}$ on $D_T^{[j]}$
    Evalute the MSE on $D_V^{[j]}$ and save its value in $\text{MSE}^{[j]}$
    $j \leftarrow j + 1$
end while

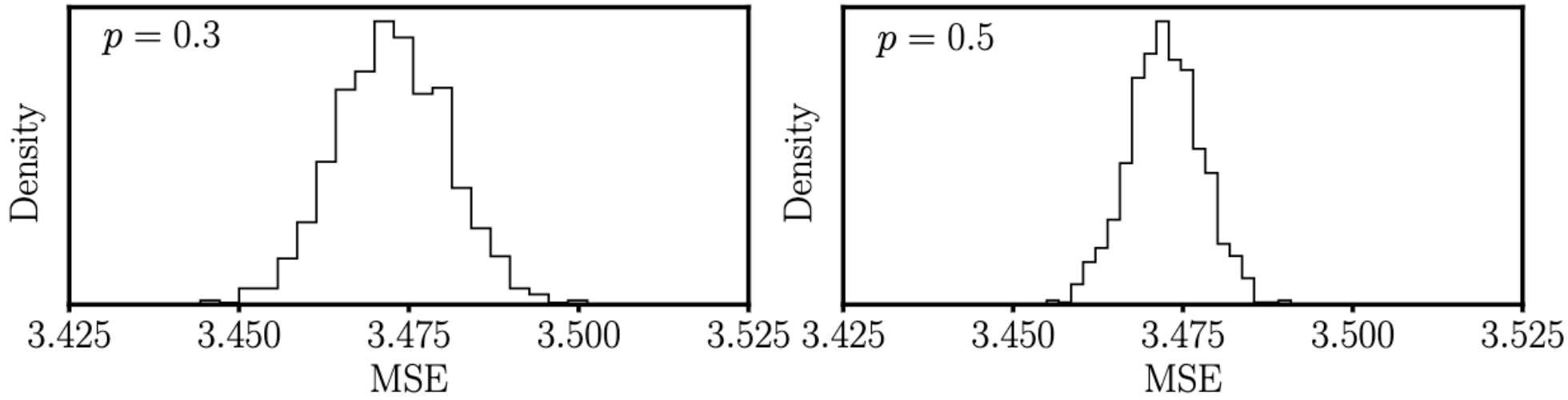
Monte-Carlo CV

The MSE distribution obtained with previous algorithm with $N = 5 \cdot 10^4$ and $p = 0.3$

Yes: the distribution is very similar to a Gaussian (normal) one (due to the **Central Limit Theorem)**



Monte-Carlo CV



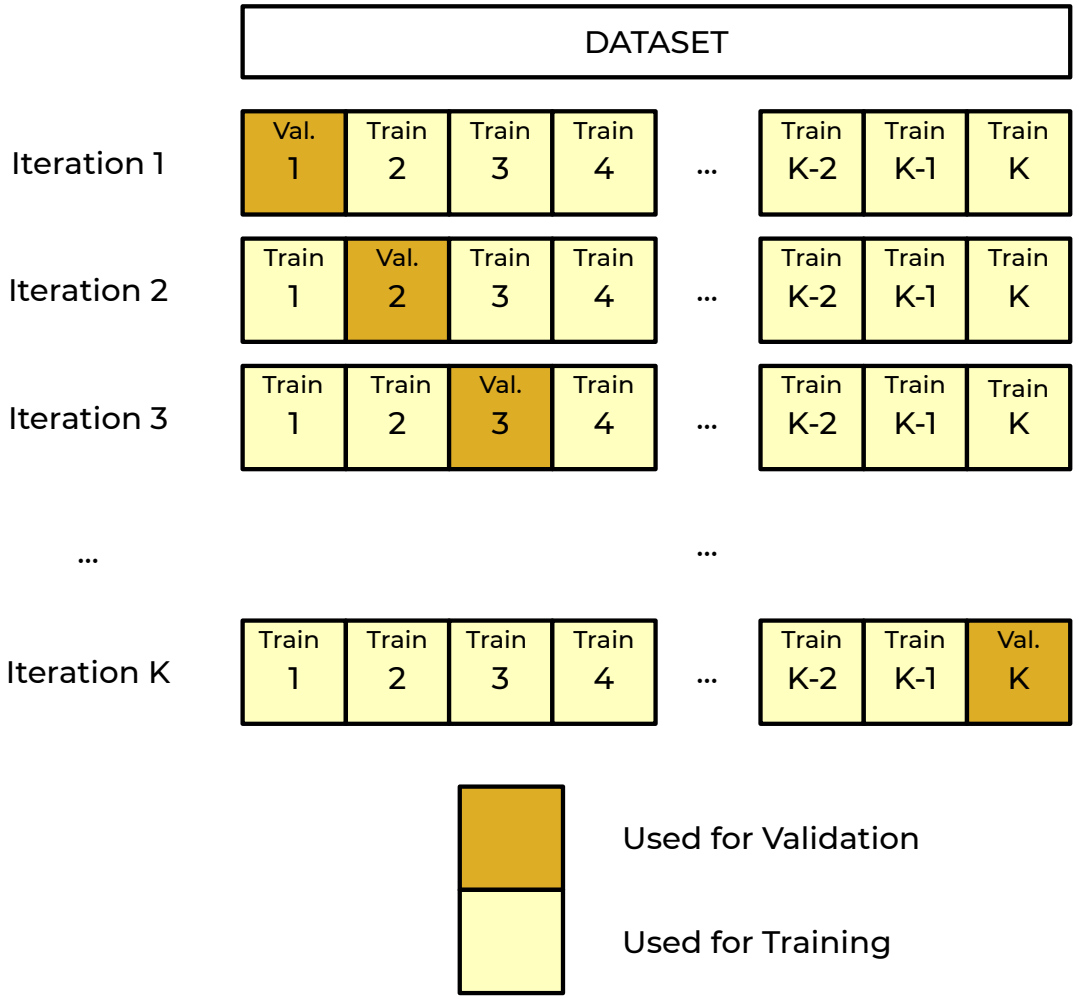
The MSE distribution obtained with Algorithm 1 with $N = 5 \cdot 10^4$ and $p = 0.3$ (left panel) and $p = 0.5$ (right panel)

The standard deviation depends on the size of the test dataset (central limit theorem). So be very careful!

Monte-Carlo CV

With the Monte Carlo Cross-Validation approach, you are not assessing the performance of one single model, but you are assessing the potential performance of a generic model type (for example linear regression) when assessed on a validation dataset of a given size.

K-Fold Approach



You get K values of the chosen metric.

→ You can evaluate mean and variance of the metric for the training and for the test set.

Some tips and remarks

→K-Fold is typically used when the dataset is small. Typically one chooses K=5-10 (the higher the more computational requirements you have)

→Leave-one-out (K-Fold with K=1) is used for very small datasets

→Hold-out with multiple split is often used

→Comparison of models (and thus of mean and variance) requires statistical tests (like the t-test) (that we cannot cover in this lecture)

Which CV method to choose

k-Fold Cross-Validation:

- *When to use:* k-fold is suitable for datasets that are large enough to be split into multiple smaller sets, ensuring each fold can serve as a meaningful test set. It works well with datasets that are fairly balanced, meaning that each class or outcome of interest is roughly equally represented across the dataset. When the goal is to obtain a more accurate estimate of model performance than what a simple hold-out approach would provide, k-fold cross validation helps in averaging the performance across multiple subsets. Additionally, when comparing different models or different configurations of the same model, k-fold cross validation provides a more reliable basis for comparison than a single train-test split (more on that later).

Which CV method to choose

Stratified k-Fold Cross Validation:

- *When to use:* when dealing with datasets where some classes are underrepresented, stratified k fold ensures that each fold maintains the same proportion of each class as in the full dataset. This is crucial for maintaining the integrity of the model's performance across classes. It's most beneficial in classification problems where maintaining the distribution of classes is important. For instance, in medical diagnosis datasets where one outcome is much rarer than another, stratified k-fold helps in preserving this imbalance across all folds. Because of the stratification, it offers a more accurate reflection of the model's ability to handle class imbalances. When tuning hyperparameters (more on this later), especially in imbalanced classification problems, stratified k-fold provides a more consistent and fair evaluation of each set of parameters.

Which CV method to choose

Leave-One-Out Cross-Validation (LOOCV):

- *When to use:* LOOCV is particularly useful when you have a very small dataset. Since it maximises the amount of training data by only leaving one sample out at a time, it is beneficial when each data point's inclusion in training significantly impacts the model's learning. LOOCV can be a good choice if the dataset is balanced and the samples are relatively homogeneous, meaning the exclusion of a single sample doesn't significantly change the nature of the training set.

Which CV method to choose

Monte Carlo Cross-Validation:

- *When to use:* it's suitable for large datasets. Monte Carlo cross-validation allows for a quicker assessment of the model performance. Unlike *k*-fold cross-validation, random subsampling allows flexibility in choosing the proportion of the dataset to be used as the training and test sets. This can be particularly useful when working with very large datasets (for example if you have enough data you could use 99% of the data for training and just 1% for validation, if 1% contains enough data and is representative of tehe entire dataset you are good to go). It offers a good balance between computational efficiency and the stability of the performance estimate. The number of repetitions can be adjusted on the basis of the available computational resources and the desired confidence in the model's evaluation.

An example

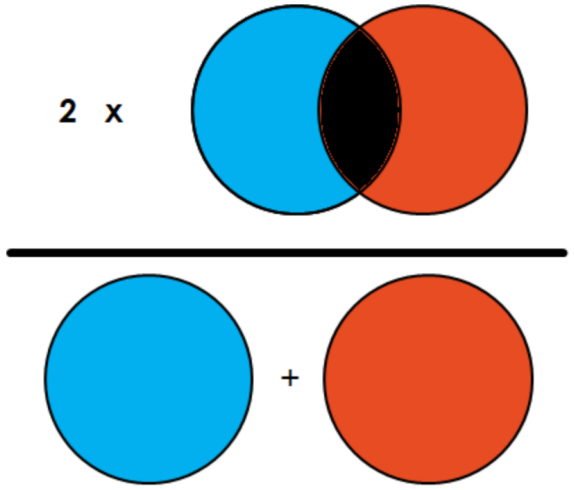


Illustration of Dice Coefficient. 2xOverlap/Total number of pixels

Image Source: towarddatascience.com

OptTTA
85.0 ± 2.5
$\mathbf{82.0 \pm 2.7}^{\dagger}$
88.8 ± 1.7

Dice (%) Results (+/- std dev.)

From: Tomar, Devavrat, et al. "OptTTA: Learnable Test-Time Augmentation for Source-Free Medical Image Segmentation Under Domain Shift." *Medical Imaging with Deep Learning*. 2021.

Model Selection

Model Selection (Definition)

Model selection in the context of machine learning and statistics is the process of choosing the most appropriate model from a set of candidate models for a given dataset.

The goal of model selection is (typically) to identify the model that best balances the trade-off between underfitting and overfitting, thereby providing the most accurate and generalizable predictions on new, unseen data.

(hyper)-Parameters

Definition 7.4 (Hyper-Parameters) Parameters that are related to the model (for example, depth in decision trees, or the learning rate) and are not influenced by the training process (thus do not depend on the training data) are called **hyper-parameters**.

Definition 7.5 (Parameters) Parameters indicates what is changed during the training process (for example, the weights of a neural network or the slope in a linear regression problem).

Model Selection Algorithm

1. Split the dataset D in three parts: D_T for training, D_V for validation and D_{test} for testing.
2. Train all q models M_i on D_T. We will assume that we want to track one single metric $m_i(\cdot)$ (for example MSE or accuracy).
3. Compute the metric on the validation dataset $m_i(D_V)$.
4. Select the model j that has the lowest value for the metric m_i (D_V).
5. Compute the metric on the test dataset $m_i(D_{\text{test}})$.
6. Compare $m_i(D_{\text{test}})$ with $m_i(D_V)$. If the two values are similar you are good and you can finally train your chosen models on the entire dataset, ready to be used on real data.

Number of datasets

***Warning* Number of Datasets**

You will need to split your dataset in three portions **only** if you are doing hyper-parameter tuning. If you simply want to validate a class of models, then two portions will be the number you need. Remember, you may overfit a model to a dataset not only by training, but also by choosing values for the hyper-parameters that works well **particularly** for the given test dataset, but would not lead to a well generalising model. This is the reason you need three portions of a dataset when you are performing hyper-parameter tuning.