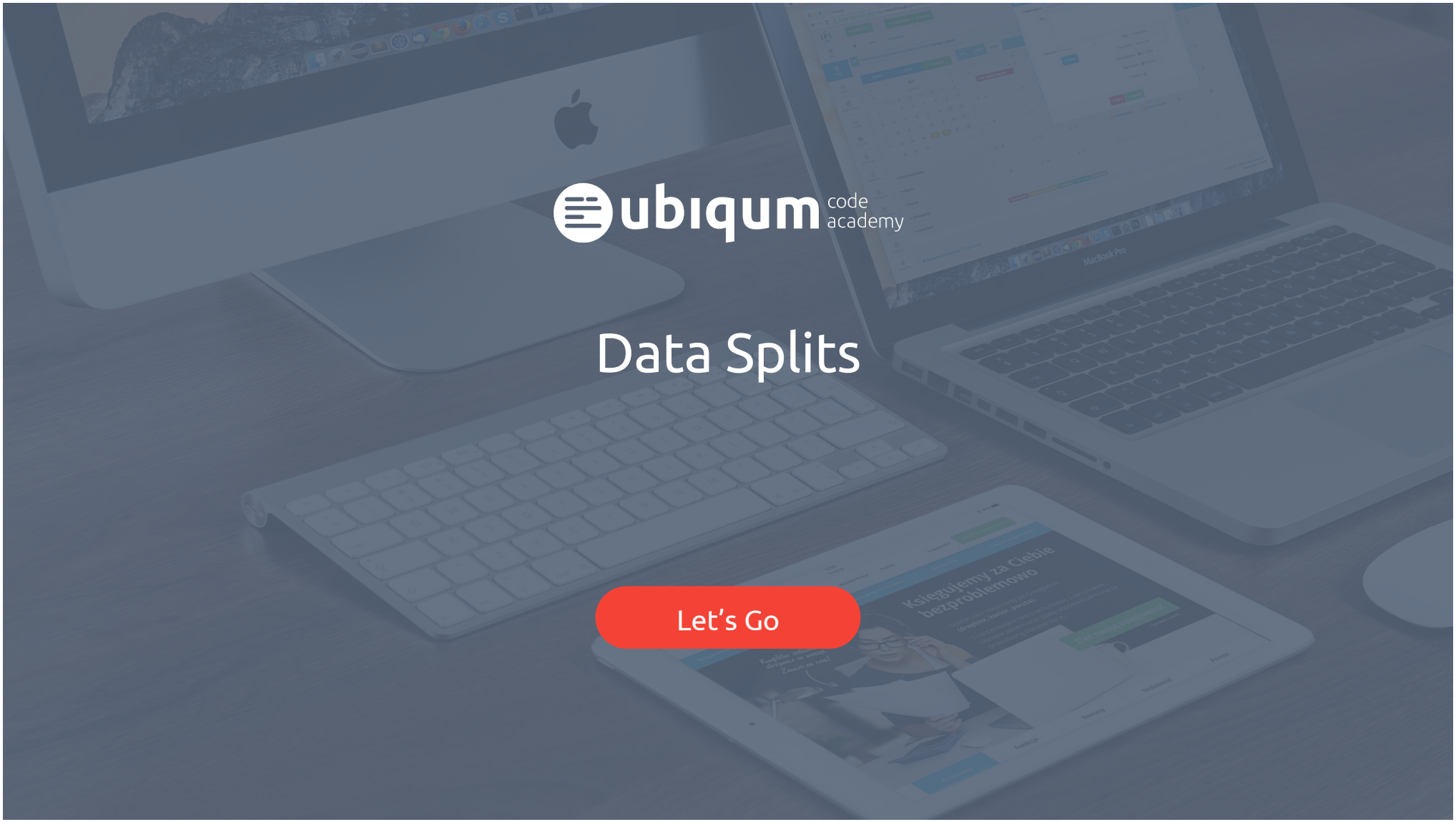

ubiqum
code
academy
Data Splits
Let's Go

Supervised Learning

Supervised learning (SL) is the machine learning task of learning a function that maps* an input to an output, based on example labelled input-output pairs.

* Mapping refers to any prescribed way of assigning to each object in one set a particular object in another set

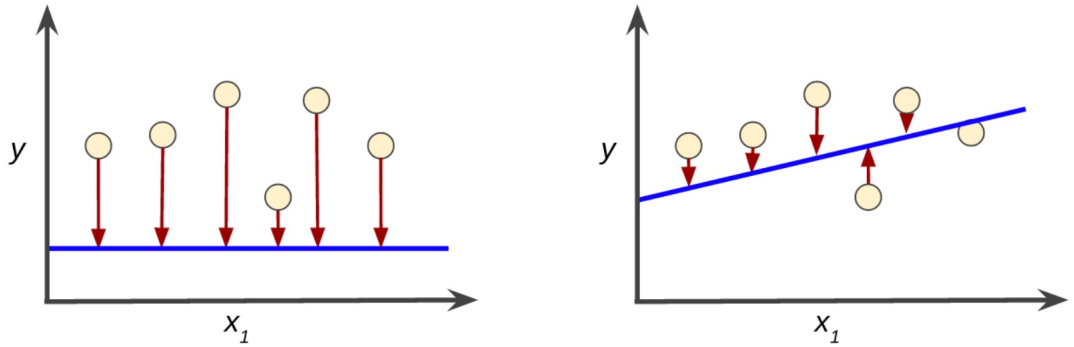
Classification VS Regression

Classification and regression are two major types of prediction problems Machine Learning tries to solve. Fundamentally, they differ because:

- classification is about predicting a label (true/false, low/medium/high, etc.)
- regression is about predicting a quantity (age, height, etc.)

Training a model

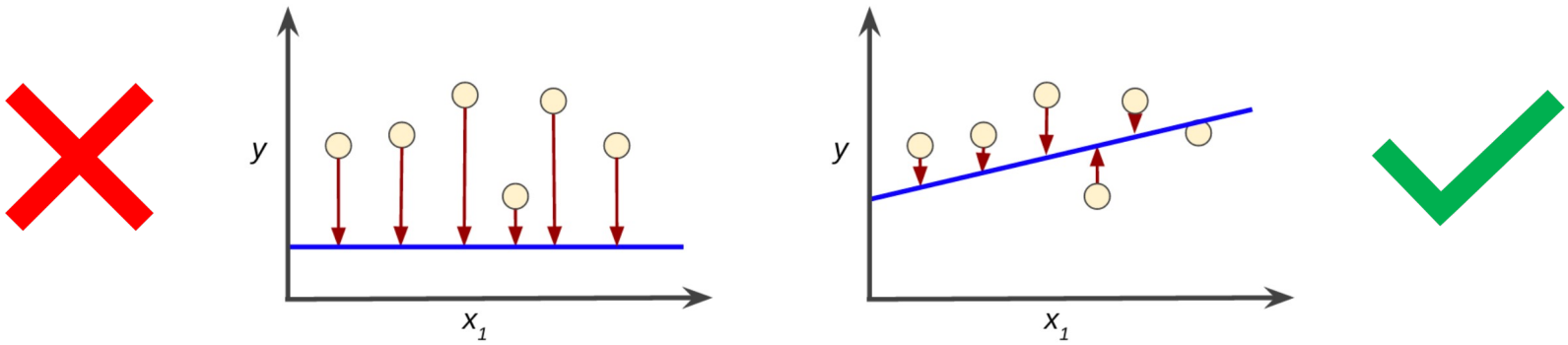
Regardless the type of problem we are asked to solve, the first step is to **fit** or **train** a model, which means making your algorithm learn the relationship between predictors (independent variables) and outcome (dependent variable), so that you can later predict the future values of the outcome.



* The red arrows represent loss, while the blue lines represent our predictions

The Best Fitted Model

However, our goal is not to just create a model. We must do so and find one that best minimizes loss (the error made by the model when predicting the values of Y based on values of X), not just any model.



* The red arrows represent loss, while the blue lines represent our predictions

To recap…

We are not interested in past data. Our goal is to make sure our model is reliable and able to predict, as precisely as possible, future values of the outcome (dependent variable).

In other words, the algorithm has to find the best possible prediction function.

We achieve this by:
a. Evaluating the performance of our model
b. Comparing the performance of several duplicates of the model (each using a different algorithm) and choosing the best performing one

Overfitting

Learning the parameters of a prediction function and then testing it on the same data is a methodological mistake.
A model that would just repeat the labels of the samples that it has just seen would have, in fact, a perfect score, but would fail to predict anything useful on yet-unseen data.

This situation is called **overfitting**.

Avoiding Overfitting

In order to avoid overfitting, it is common practice, when performing a (supervised) machine learning experiment, to hold out part of the available data as a **test set**, using the so-called train-test split procedure.

Train-test split procedure

It is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It involves taking a dataset and dividing it into two subsets, Train Dataset and Test Dataset

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate how well fitted the machine learning model is.

Train-test split procedure

- It can be used for classification or regression problems
- It can be used for any supervised learning algorithm
- Although simple to use and interpret, it should not be used when:
  - The dataset is not sufficiently large
  - In case of a classification problem, the dataset is not balanced.

What's the optimal split percentage?

The optimal split percentage does not exist, as you must choose one that meets your project's objectives with considerations that include:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Model Parameters VS Model Hyperparameters

Model Parameters are the parameters in the model that must be determined using the training data set, and are, indeed, the fitted parameters (estimated by fitting the given data to the model).
The slope and the intercept in Linear Regression are examples of parameters

Hyperparameters are adjustable parameters that must be tuned in order to obtain a model with optimal performance. Their values are set before the model start training, and they cannot be learned by fitting the model to the data. The desired depth and number of leaves in Decision Tree are examples of Hyperparameters.

Iteration after iteration…

When evaluating different settings (“hyperparameters”) for estimators, such as the C setting for an SVM, or the number of leaves in a Decision Tree, there is still a risk of overfitting on the test set. This can happen because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can “leak” into the model.

In other words, the model learns indirectly about the test set (which in ideal circumstances should remain completely unseen until the very end) and evaluation metrics fail to report on generalization performance, as they no longer refer to the model performance on completely unseen data.

Iteration after iteration…

To solve this problem, one can use the Train-Validation-Test split.

Yet another part of the dataset can be held out as a so-called “validation set”:

- training proceeds on the **training set**
- evaluation is done on the **validation set**
- when the experiment seems to be successful, final evaluation can be done on the **test set**.

Iteration after iteration…

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 for the pair of (train, validation) sets.

A solution to this problem is called cross-validation (CV).

K-fold Cross Validation

A test set is still held out for final evaluation, but the validation set is no longer needed when doing CV.

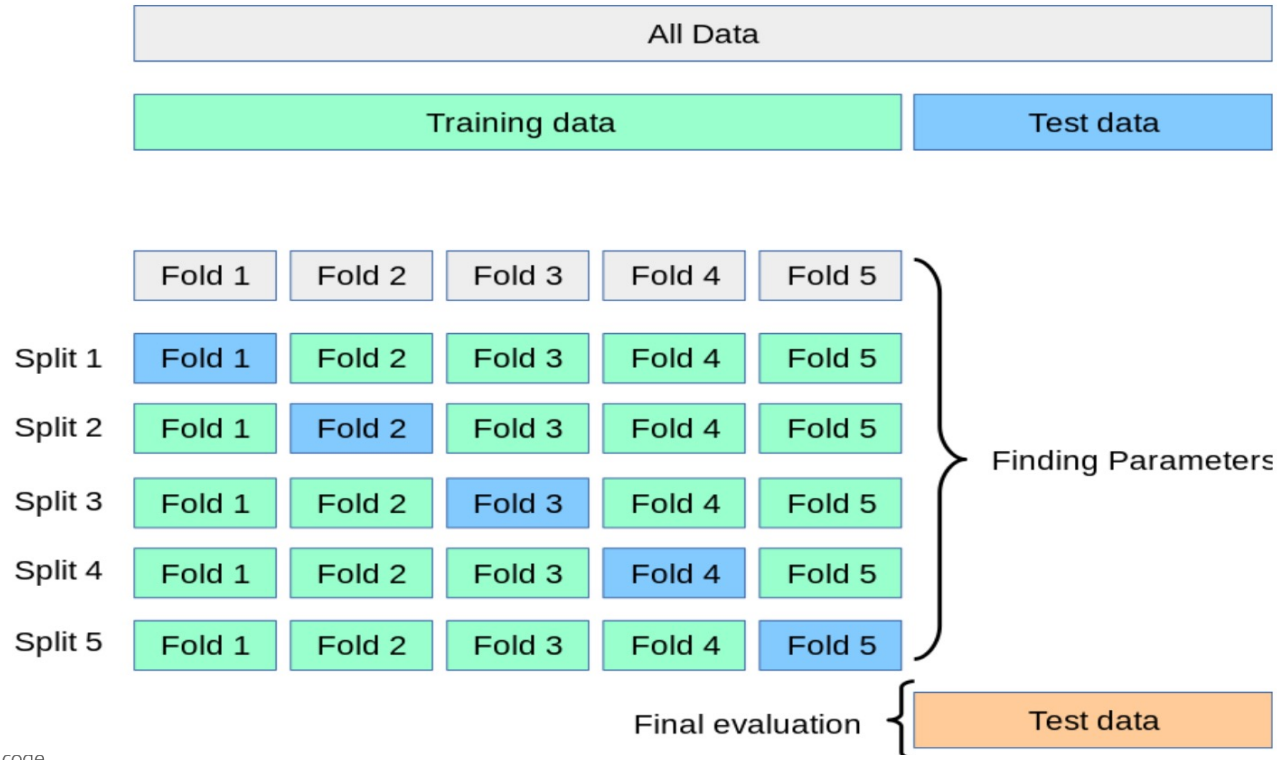
In the basic approach, called ***k*-fold CV**, the training set is split into *k* smaller sets. For each of the *k* "folds":

- A model is trained using k–1 of the folds as training data;
- The resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The performance measure reported by *k*-fold cross-validation is then the average of the values computed in the loop.

K-fold Cross Validation

This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set)



More on Cross Validation and its most common declinations coming soon…

Links

https://scikit-learn.org/stable/modules/cross_validation.html

https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

https://medium.com/analytics-vidhya/a-simple-introduction-to-validating-and-testing-a-model-part-1-2a0765deb198

https://medium.com/@ODSC/properly-setting-the-random-seed-in-ml-experiments-not-as-simple-as-you-might-imagine-219969c84752

Thanks