Machine learning 101: cross-validation

Tomasz Fruboes
Narodowe Centrum Badań Jądrowych

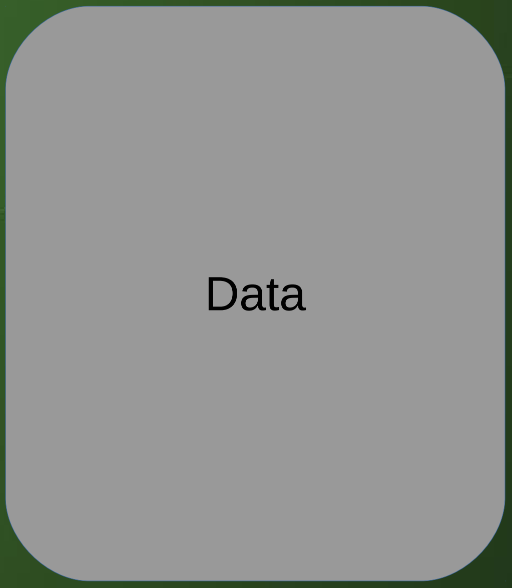
for slides visit pragmaticpython.com/xvalidation

Terminology

- This story will be told in a context of classification problem – “predict if this data point comes from class A or class B”.
  - Same methods apply to regression problems (“predict a value for this data point”)
- Machine learning (ML) is an iterative process. Usually you fit and test multiple different models
  - Hyper-parameters tuning

Wrong approach

1) Clean data
2) Fit model using all data
3) Measure model performance using all data
4) Change model parameters, e.g. depth of decision tree (hyper-parameter tuning)
5) Repeat steps 2)-4) until best possible model found

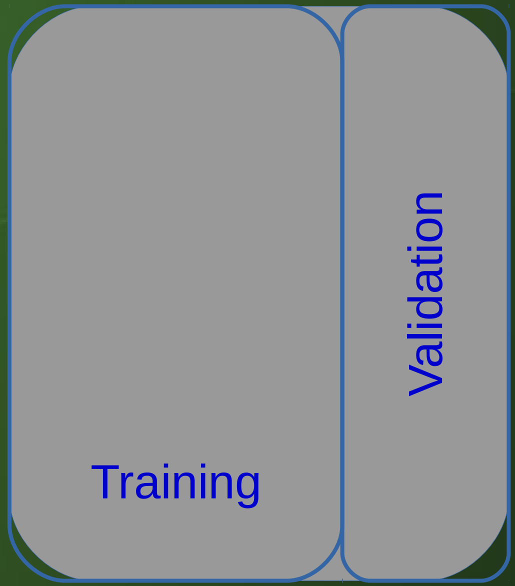


In such workflow you may reach 100% model accuracy for any ML problem*

*for ML algorithms with large enough learning capacity (e.g. decision trees)

Less wrong approach

1. Clean your data, split into **training** and **validation** sets
2. Fit model using **training** set
3. Measure model performance on the **validation** set
4. Change model parameters, e.g. depth of decision tree (hyper-parameter tuning)
5. Repeat steps 2-4 until best (=maximum accuracy over validation set) model is found



Why not use validation set for final performance evaluation?

- In “real life", multiple models (with different hyper-parameters) will be (nearly) equally good
- In such case, the main difference in performance will come from statistical fluctuations

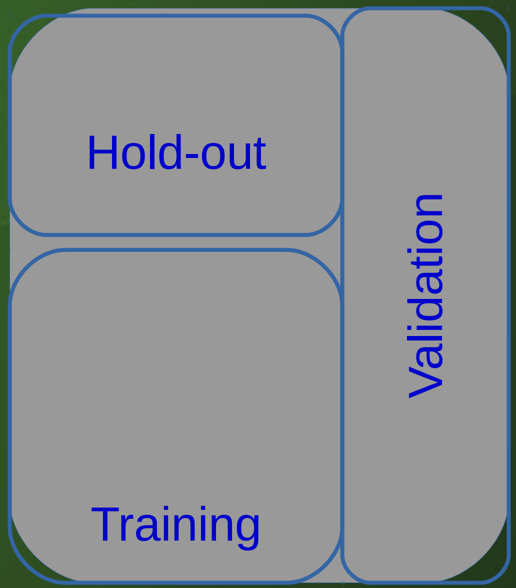
  → Performance measurement done on the validation set may be significantly higher than the true one

Extreme example

- Some ML algorithms start by randomly selecting a subset of the training set
  - Seed of the random number generator is a parameter that you can set. It may be tempting to treat it as a hyper-parameter and optimize it (note: don't do it in “real life”)
- In such scenario there will be a “best model” (with the highest accuracy) for some seed
  - Somehow we don't expect such model to outperform models with different random seed values
  - Performance difference comes only from statistical fluctuations (i.e. we are not finding any better relationship within our data)

How to evaluate final model performance?

- For reliable results, sacrifice part of your data for final model evaluation
  - Split data into three sets: training, validation and hold-out

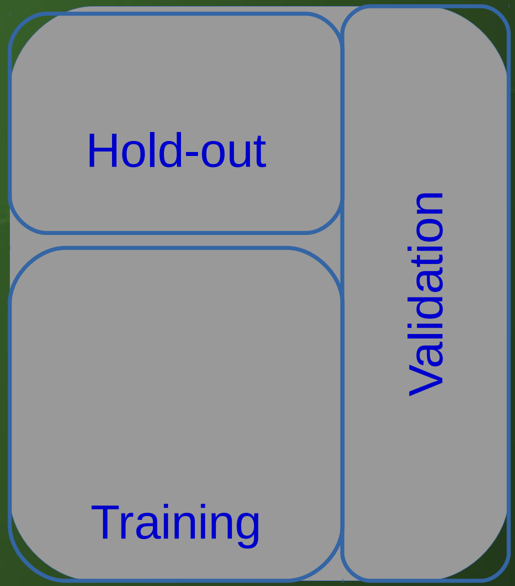


Correct approach

- As previously:

1. Fit model using **training** set
2. Measure model performance on the **validation** set
3. Change model parameters, e.g. depth of decision tree (hyper-parameter tuning)
4. Repeat steps 2-4 until best (=maximum accuracy over validation set) model is found

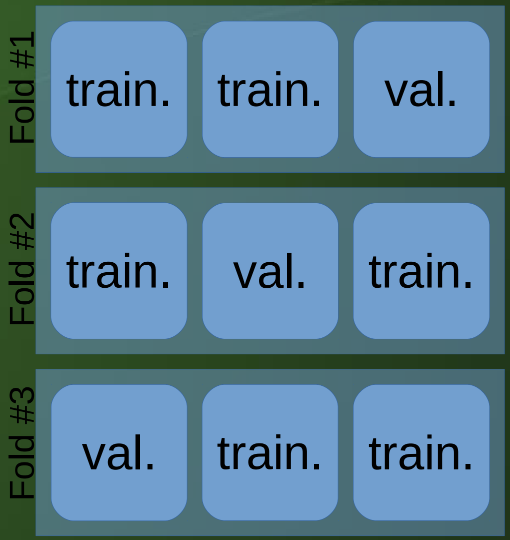
- For final evaluation measure performance of the chosen model using **hold-out** set



Some common pitfalls and their remedies

Common pitfall #1 – not having a holdout set

- Not having a hold-out set usually leads to over-optimistic performance evaluation
- Tempting - more data for training usually means better model
  - If amount of data is an issue, use k-fold cross-validation (independent hold-out set still needed)
  - Double cross validation is another technique of maximally utilizing available data
    - generalization of k-fold cross-validation - two nested k-fold cross-validation loops



Common pitfall #2 - using the hold-out set more than once

- Tempting - “can I do little better here?”
- In some cases cannot be avoided (e.g. a bug is found and fixed in data-cleaning step)
  - Not a mistake - as long you are not using the hold-out set to select the best performing model
- It may be possible to bypass the “use the hold-out set once” rule by presenting different parts of the hold-out set for each test.
  - See “The reusable holdout: Preserving validity in adaptive data analysis” (DOI: 10.1126/science.aaa9375)
  - Rather rarely used

Common pitfall #3 – sets independence

- All three (training/validation/hold-out) sets must be independent
  - In some cases this requires extra care
- Real-life example – predicting medical diagnosis for given patients' stay

Common pitfall #3 – sets independence

- Problem statement:
  - Input: set of symptoms and lab results (e.g. blood pressure or white blood cells count) observed during patient stay
  - Output: what was the patients' disease

Common pitfall #3 – sets independence

- Given patient may have multiple hospital stays
- Same diagnosis is likely for different stays
  - Data points are not independent
- All stays from given patient must be put in the same set.
  - Violation in two possible ways

Common pitfall #3 – sets independence

- Possibility #1 – patient has stays in training and validation set
  - You are likely to select a model trying to identify given patient (and not to find relation between symptoms and disease). Useless for future patients
- Possibility #2 – patient has stays in training and hold-out set (similar for validation and hold-out)
  - Final performance measure will be overoptimistic

Take-outs

#1. For reliable results, you need to sacrifice part of your data for final performance evaluation. Split data into training, validation and hold-out sets

#2. All three sets must be independent

#3. Use the hold-out set just once (for final model performance evaluation)