18.09.2024
**Colloqium / Hands - On**

TK8117 – Week 04 - Topic 01:

# Cross Model Validation

Andreas Gudahl Tufte
Anette Fagerheim Bjerke
Aria Alinejad
Azimil Gani Alam
Cameron Louis Penne
Chinmay Anirudha Patwardhan
Irene Hofmann
Nicole Quattrini
Torstein Nordgård-Hansen

Norwegian University of Science and Technology

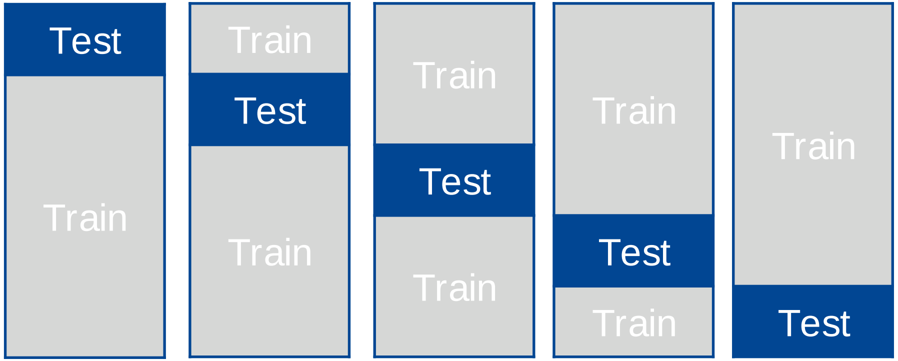
NTNU

# Overview

- Cross-Validation
- Cross *Model* Variation
- Example
- When to use?
- Pros and Cons
- Summary

# Cross-Validation

*Data Partitioning technique into some subsets*




1. Split data into k folds (e.g.: 5)
2. Train model with training data
3. Measure score with test data
4. Average scores

**Problem:**

- Risky to find best model by just trying out different variables on the same test data.
  —> high chance that model does not work well on new unseen data

**Solution**: test the model on a completely separate data set.

# Cross Model Validation

Also known as:

- ***Double cross validation***
- ***Nested cross validation***
- ***Repeated cross validation***

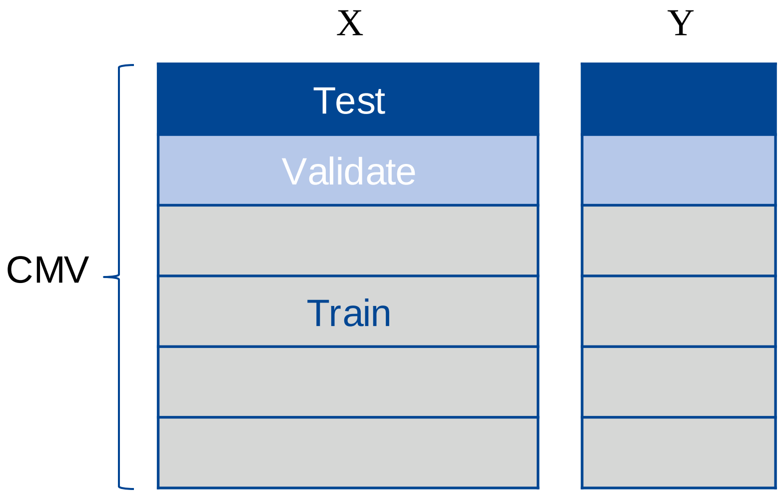
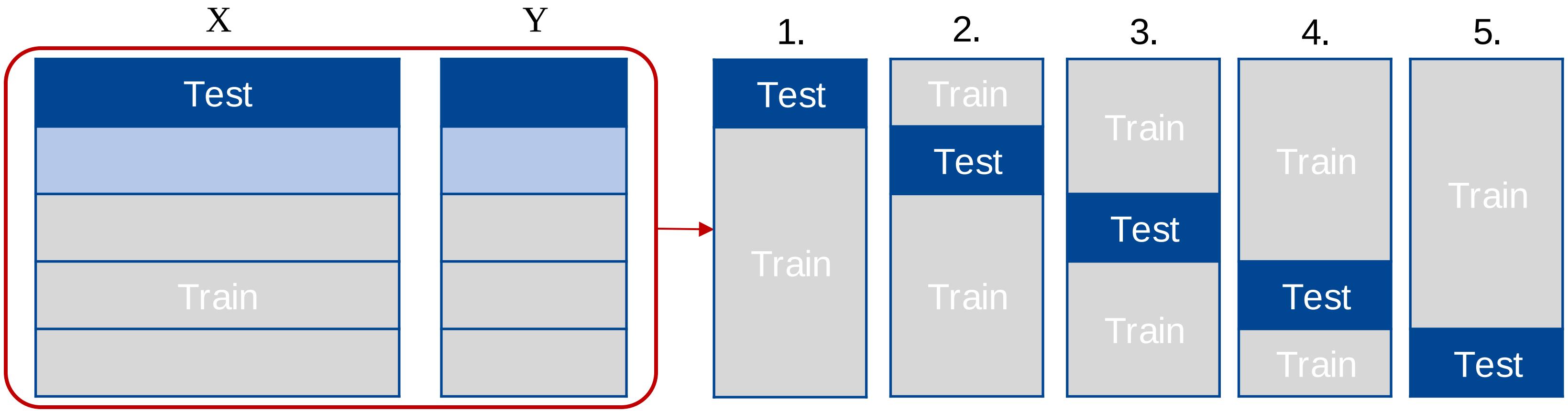


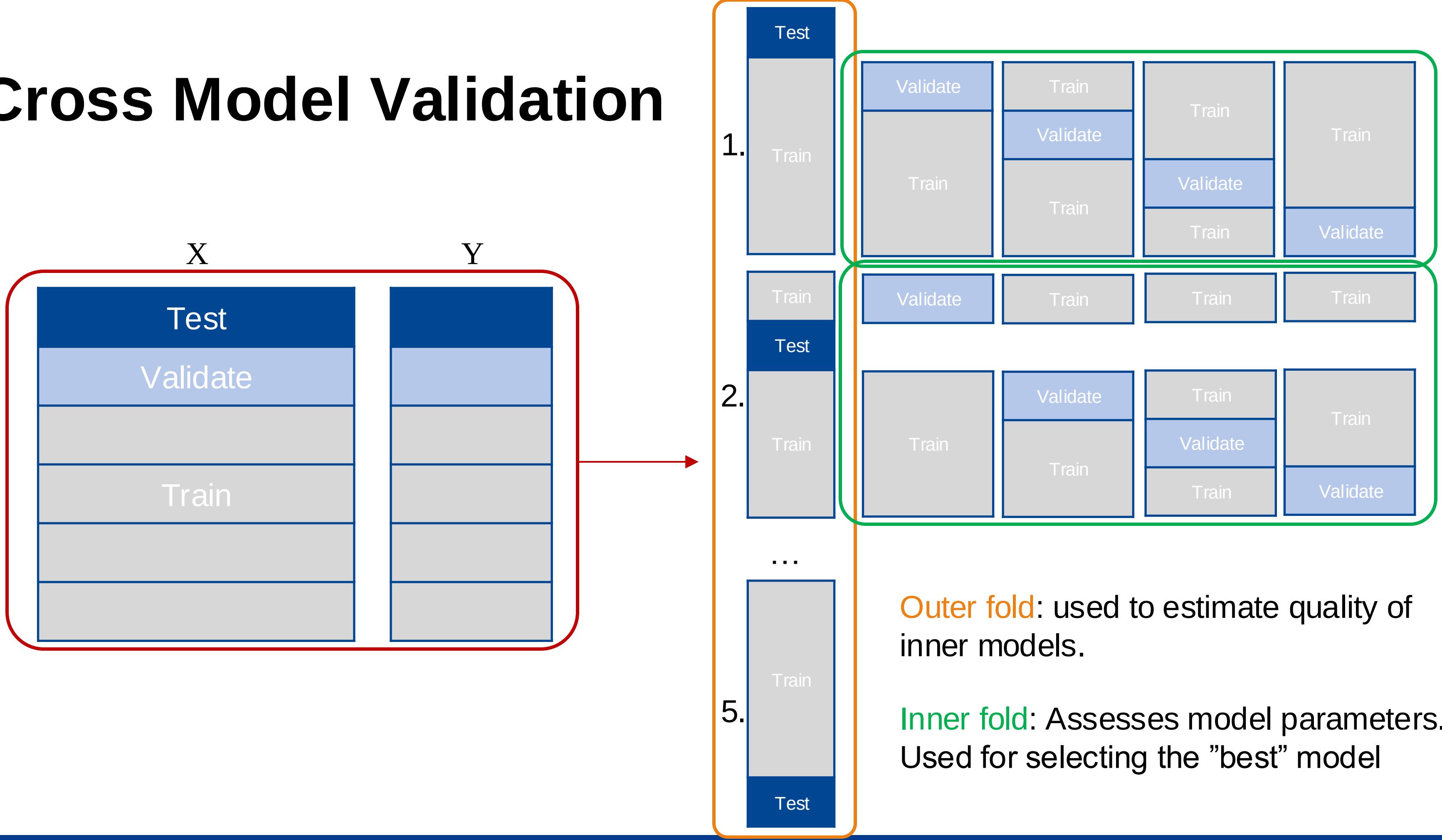



Figure inspired from: *Westad F. A retrospective look at cross model validation and its applicability in vibrational spectroscopy. Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy. 2021 Jul;255:119676*

# Cross ~~Model~~ Validation



# Cross Model Validation




Outer fold: used to estimate quality of inner models.

Inner fold: Assesses model parameters. Used for selecting the ”best” model

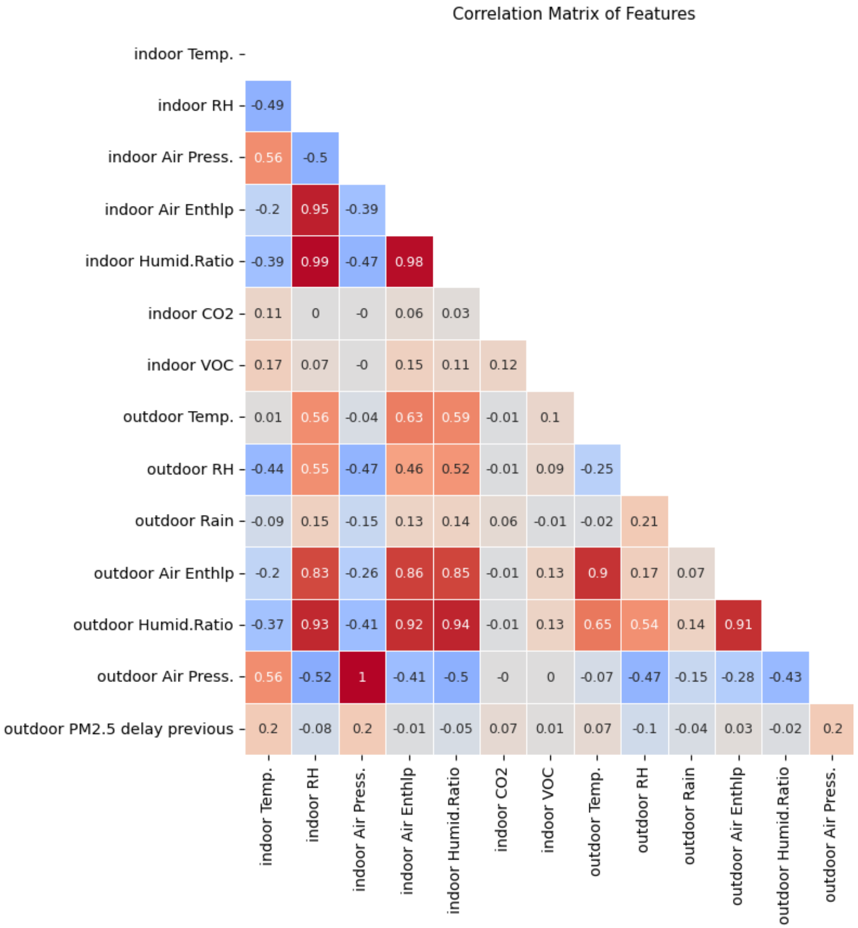
# Variable Selection

1. Run model with jackknifing
    1. Leaving one sample out
2. Find significant variables using model
    1. PLSR: variable importance in projection (VIP) using weights
3. Repeat 1) and 2) with different sample left out
4. Check frequency of variables found significant during inner fold runs

# Code example

# Data

- Air quality
- Predicting PM1
- 14 variables in dataset

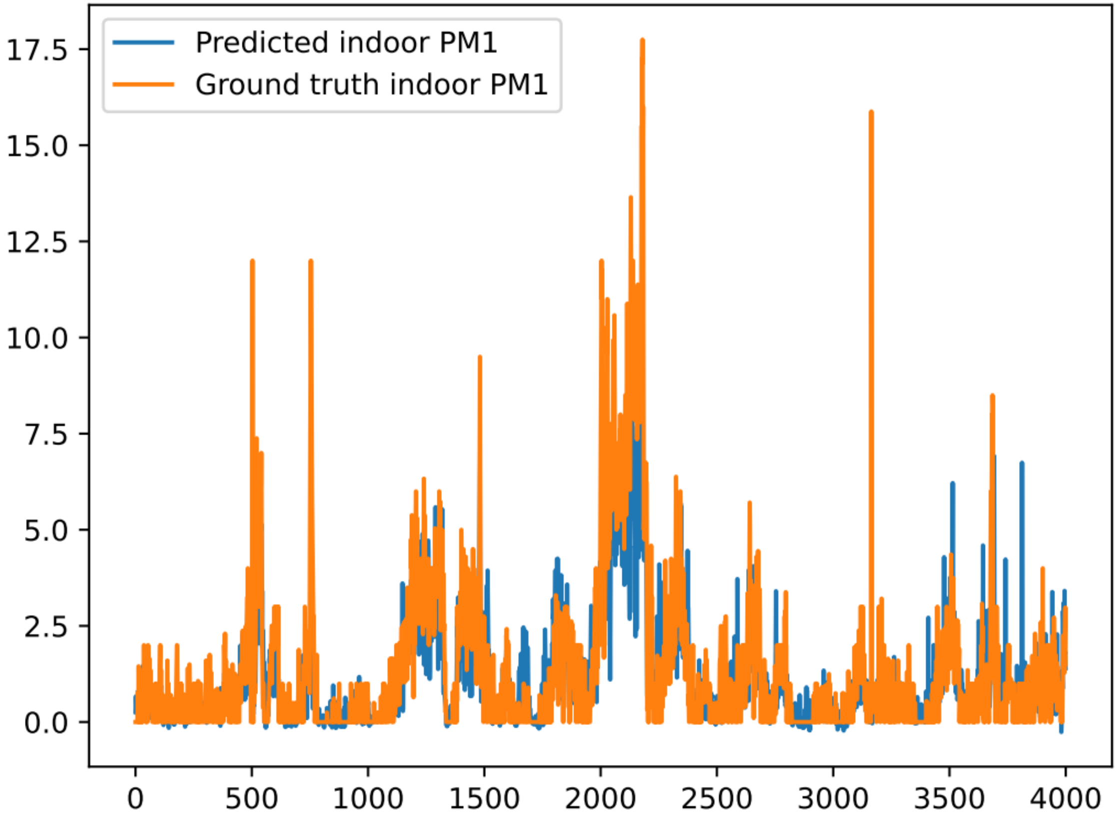



| (Unit) | Parameters | measurement location |
|---|---|---|
| (ppm) | Carbondioxide (CO2) | indoor |
| (ppb) | Volatile Organic Compounds (TVOC) | indoor |
| (ug/m3) | PM1.0 | indoor |
| (ug/m3) | PM2.5 | outdoor |
| (%) | Air Relative Humidity | indoor & outdoor |
| (deg C) | Air Temperature | indoor & outdoor |
| (hPa) | Air Pressure | indoor & outdoor |
| (kJ/kg) | Air Enthalpy | indoor & outdoor |
| (gram/kg air) | Air Humidity Ratio | indoor & outdoor |
| (mm) | Rain (Precipitation) | outdoor |

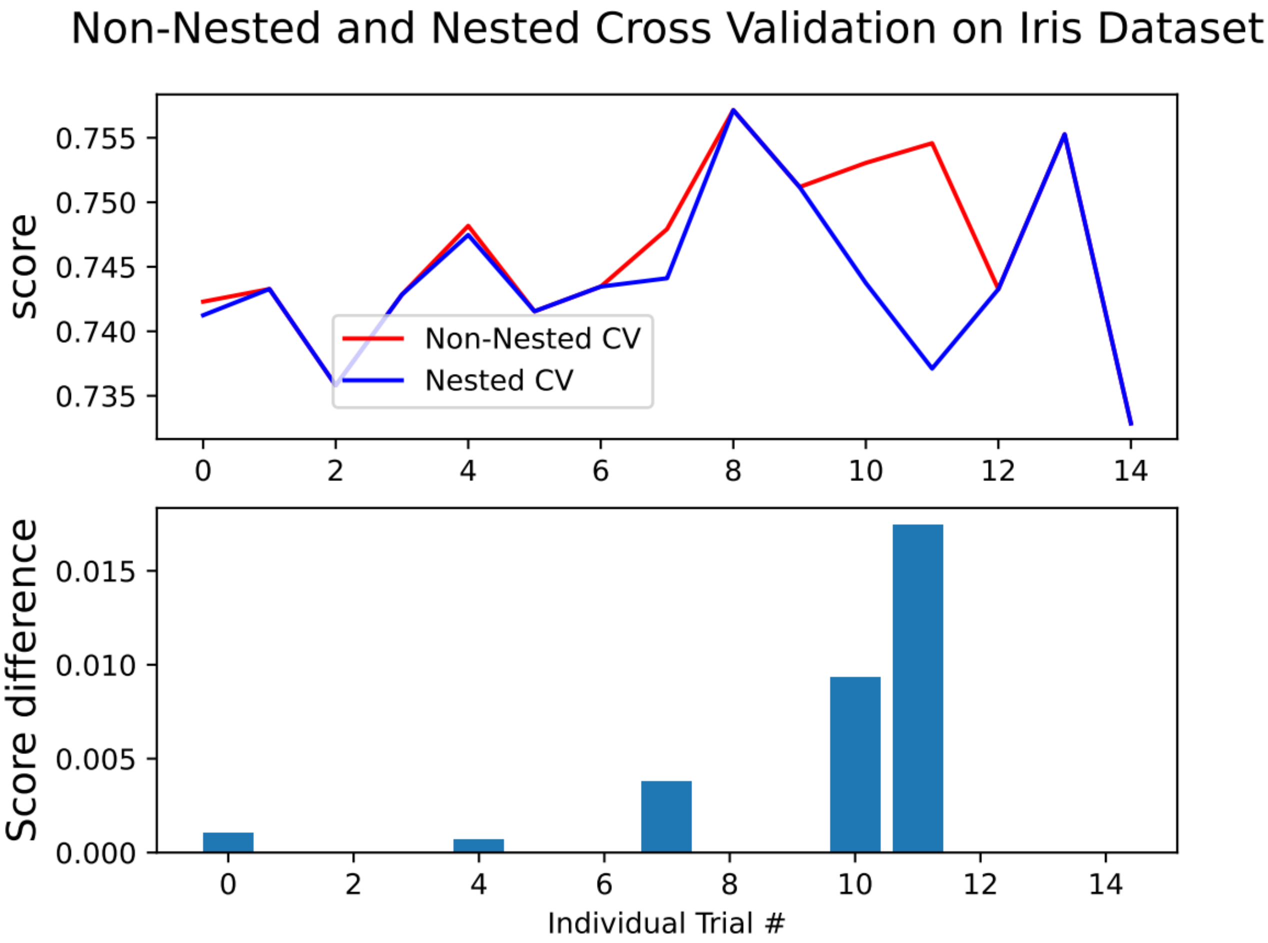
```python
# 3. Validation and nested cross-validation of the model
non_nested_scores = np.zeros(num_trials)
nested_scores = np.zeros(num_trials)

scaler = StandardScaler()
scaler.fit(x.values, y=y.values[:, 0])
scaled_x = scaler.transform(x.values)
scaled_y = y.values[:, 0]

p_grid = {"C": [1, 10, 100], "gamma": [0.01, 0.1]}

for i in range(num_trials):
    # Choose cross-validation techniques for the inner and outer loops
    inner_cv = KFold(n_splits=4, shuffle=True, random_state=i)
    outer_cv = KFold(n_splits=4, shuffle=True, random_state=i)

    # Non-nested parameter search and scoring
    regressor = GridSearchCV(estimator=svm.SVR(), param_grid=p_grid, cv=outer_cv, n_jobs=-1, scoring="r2")
    regressor.fit(scaled_x, scaled_y)
    non_nested_scores[i] = regressor.best_score_

    # Nested CV with parameter optimization
    regressor = GridSearchCV(estimator=svm.SVR(), param_grid=p_grid, cv=inner_cv, n_jobs=-1, scoring="r2")
    nested_score = cross_val_score(regressor, X=scaled_x, y=scaled_y, cv=outer_cv, n_jobs=-1)
    nested_scores[i] = nested_score.mean()

score_difference = non_nested_scores - nested_scores
```

# Predictions

- X-axis: samples
- Y-axis: PM1 value



# Scores



Non-Nested and Nested Cross Validation on Iris Dataset
score
0.755
0.750
0.745
0.740
0.735
Non-Nested CV
Nested CV
0
2
4
6
8
10
12
14
Score difference
0.015
0.010
0.005
0.000
Individual Trial #

# When to use

- Datasets where there are few samples but many variables or hyperparameters
  - Reduce overfitting and false positives
- Identifying subset of variables for model optimization
  - Variable selection

# Pros & Cons

- Prevents overfitting
- Simpler for comparing ML models
- Better hyperparameter tuning than simple cross-validation
- Works well with small data sets

- Slower process due to multiple process inside (special for large datasets),
- Complex setup and interpretation.
- Computationally very resource intensive

# Summary

- **Two-level validation –** involves an outer loop for model evaluation and an inner loop for hyperparameter tuning.
- **Prevents overfitting –** by using separate data for tuning and testing
- **Better Model Performance Estimate** – since the test data in the outer loop is not exposed during the hyperparameter tuning in the inner loop
- **Higher computational cost –** due to multiple model training runs in the inner and outer loops.