UNIVERSITÀ DEGLI STUDI DI PADOVA

Machine Learning
2024/2025

AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

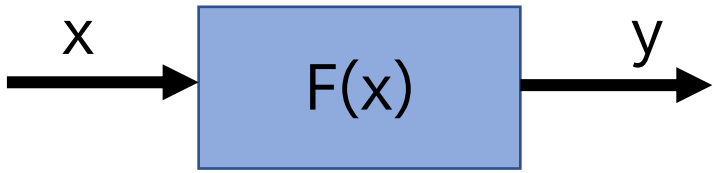
Lecture #08 Cross-Validation & Bias vs Variance & Ridge Regression

Gian Antonio Susto

Before starting

Tomorrow’s lab -> Da Room

Recap – Supervised Learning



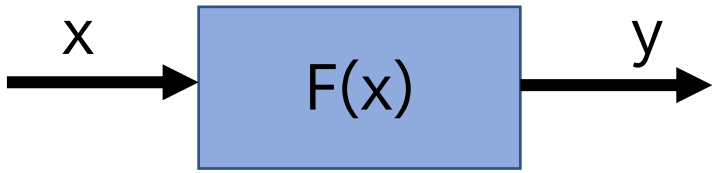
We are looking for a model *F(x)* that allows us to make predictions (estimates) of a target variable *y* based on the features $x = [x_1\ x_2\ \ldots\ x_p]$

All the model that we will consider have parameters that needs to be trained. For example, the β in OLS

$F(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$

Two elements are necessary for training:

Recap – Supervised Learning



We are looking for a model *F(x)* that allows us to make predictions (estimates) of a target variable *y* based on the features $x = [x_1\ x_2\ \dots\ x_p]$

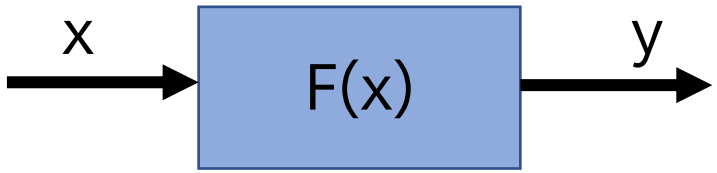
All the model that we will consider have parameters that needs to be trained. For example, the β in OLS

$F(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$

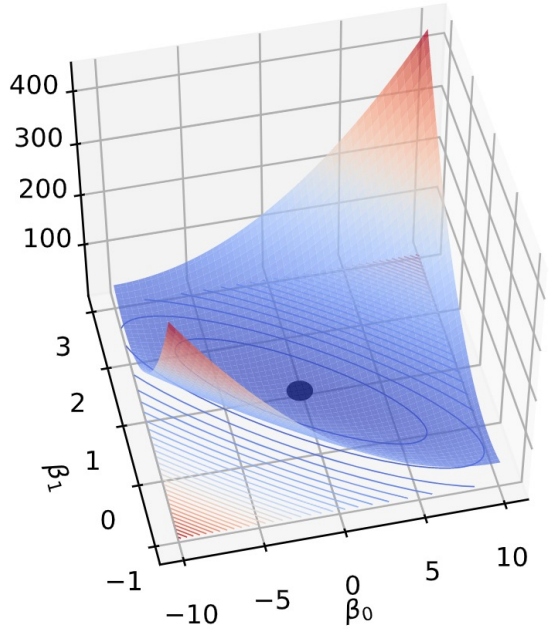
Two elements are necessary for training:

- Data
- A training procedure with an objective (cost) function

Recap – Supervised Learning



We are looking for a model *F(x)* that allows us to make predictions (estimates) of a target variable *y* based on the features $x = [x_1\ x_2\ \dots\ x_p]$



All the model that we will consider have parameters that needs to be trained. For example, the β in OLS

$F(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$

Two elements are necessary for training:

- Data
- A training procedure with an objective (cost) function

For OLS:

- Training is simple: just a closed-form solution if we consider RMSE as an objective

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Recap – Training vs Testing

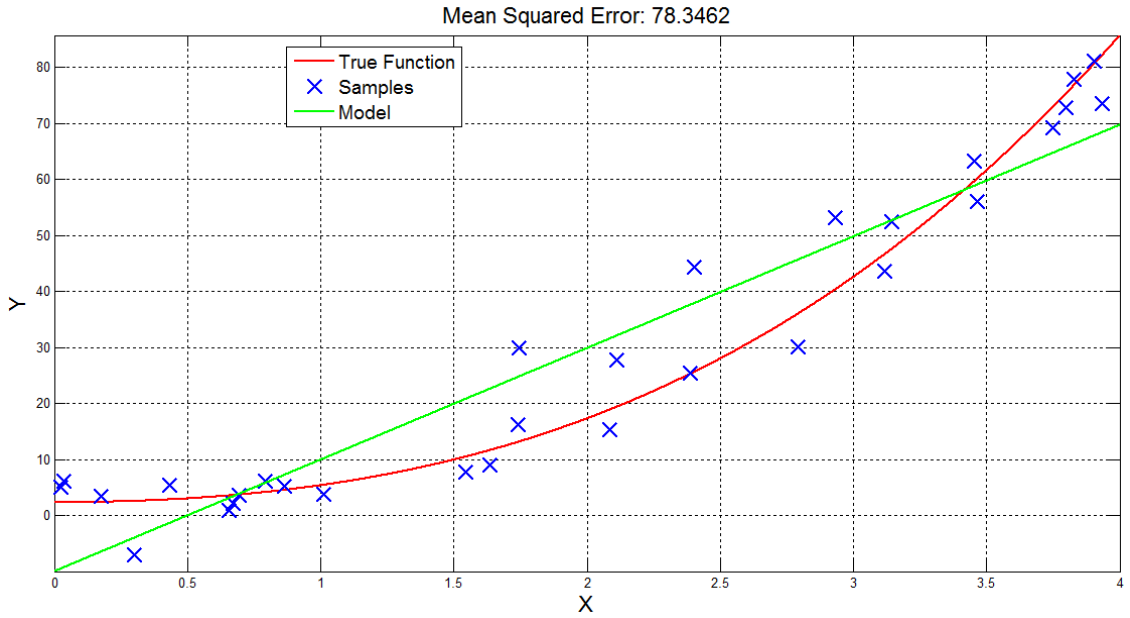
Training is not enough; we need to test a model to:

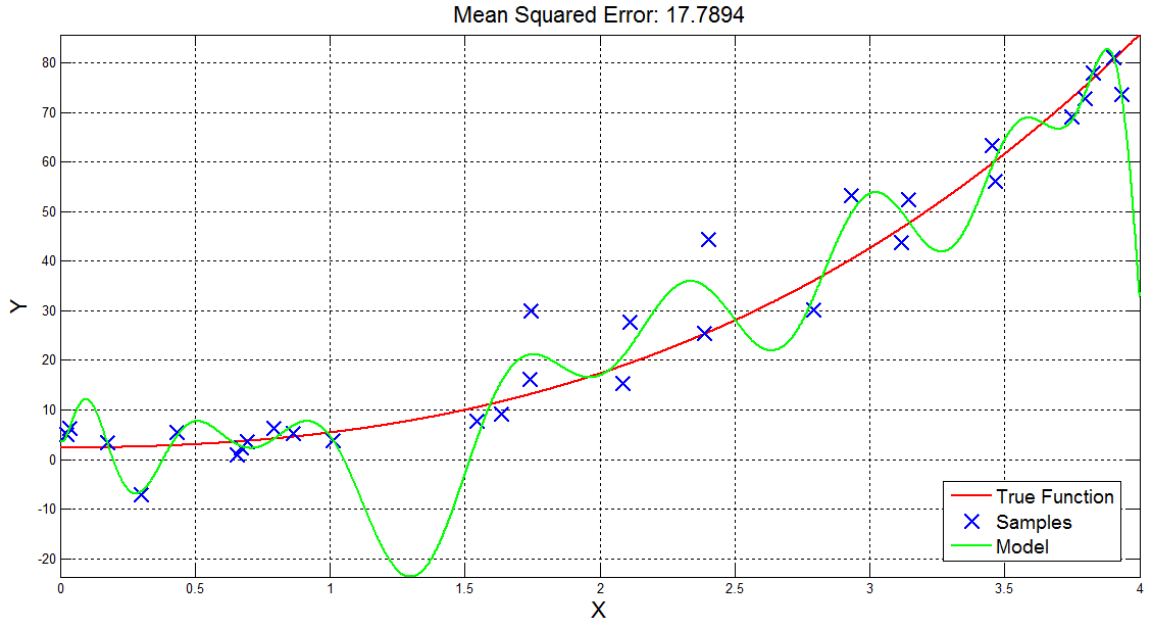
- Understand model performances (Does it generalize? What performances should I expect in the real world?)
- Among the different choices I can make in a ML pipeline, which is the best?

Recap – Training vs Testing

Training is not enough; we need to test a model to:

- Understand model performances (Does it generalize? What performances should I expect in the real world?)
- Among the different choices I can make in a ML pipeline, which is the best?

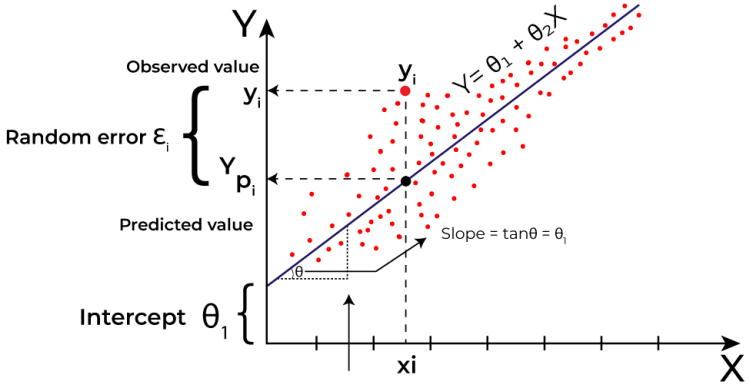


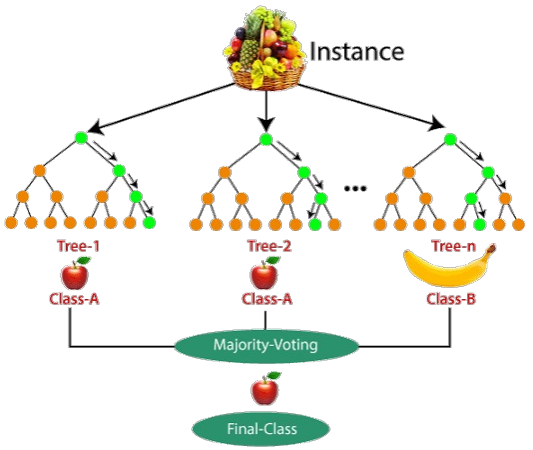


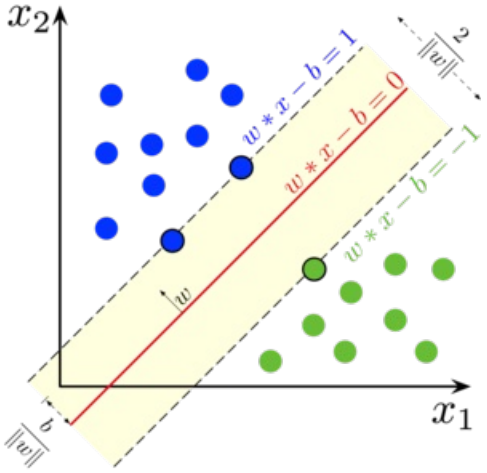
Recap – Training vs Testing

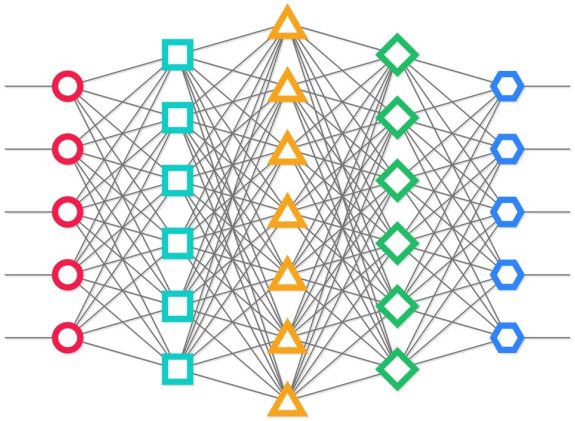
Training is not enough; we need to test a model to:

- Understand model performances (Does it generalize? What performances should I expect in the real world?)
- Among the different choices I can make in a ML pipeline, which is the best?









Recap – Training vs Testing

We put randomly 20% of the dataset in testing, while keeping the rest in training

```
Mean Squared Error (MSE): 0.657451727882265
R-squared (R2): 0.498285085954743 74

Model Coefficients:
MedInc: 0.445465596586923 64
HouseAge: 0.016904055548308032
AveRooms: -0.028380689805166 48
AveOccup: -0.004143822818663251

Intercept: 0.026697367635455382
```

Recap – Training vs Testing

Training is not enough; we need to test a model to:

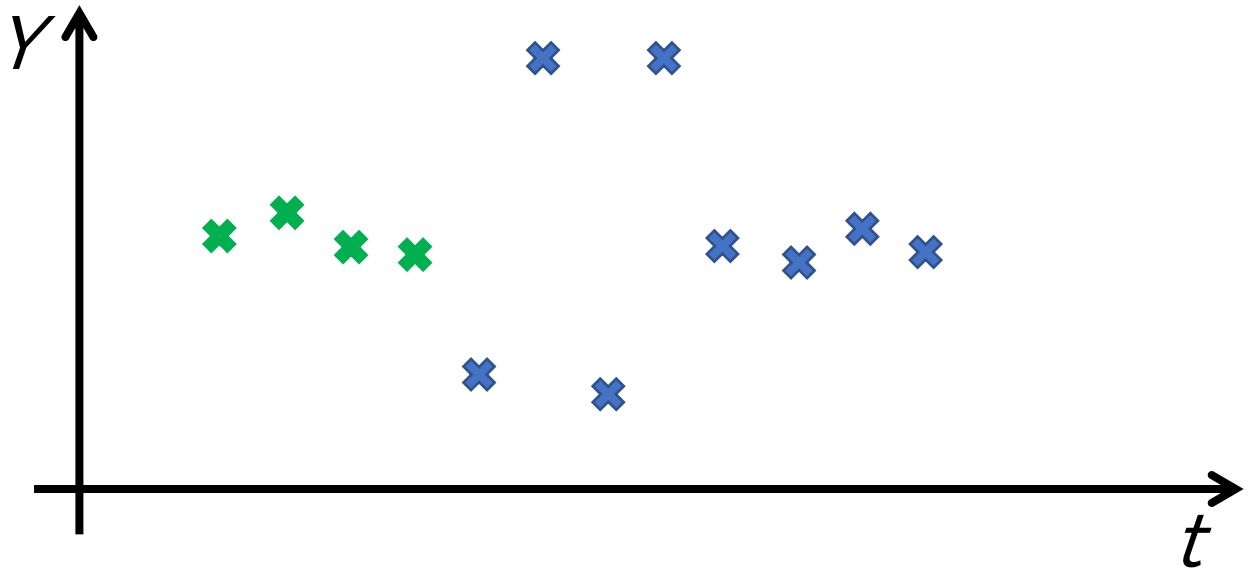
- Understand model performances (Does it generalize? What performances should I expect in the real world?)
- Among the different choices I can make in a ML pipeline, which is the best?

We then divide data into a training and a test part:

- Random choice?
- Can we do better? How to choose randomly?

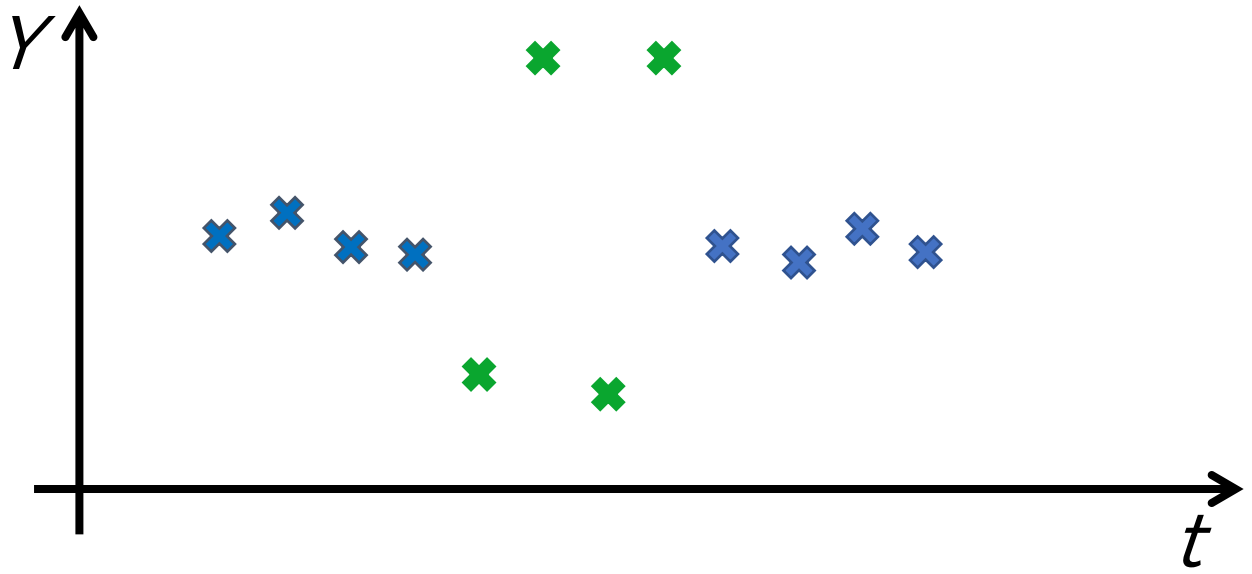
Cross-Validation

- As said, in modeling we divide data into:
  - Training ~~& Validation~~ for model building
  - Test for performance estimation
- Based on the random choice, performance can dramatically change! Especially with ‘small’ datasets



Cross-Validation

- As said, in modeling we divide data into:
  - Training ~~& Validation~~ for model building
  - Test for performance estimation
- Based on the random choice, performance can dramatically change! Especially with ‘small’ datasets



Cross-Validation

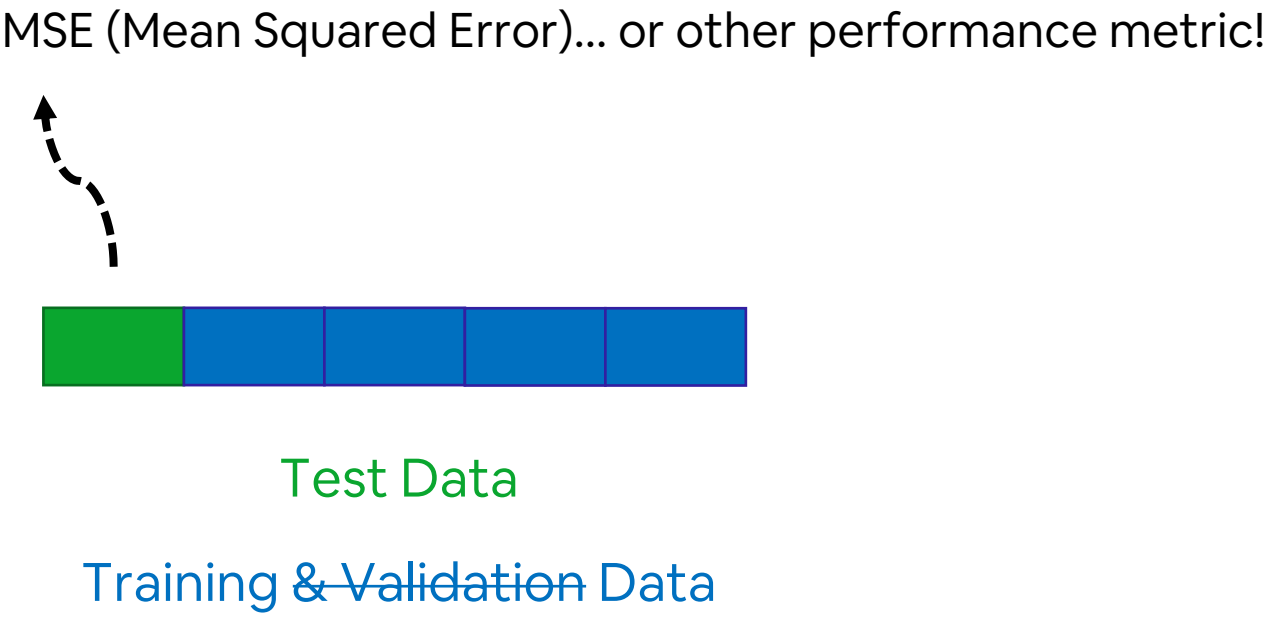
- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold

Test Data

Training ~~& Validation~~ Data

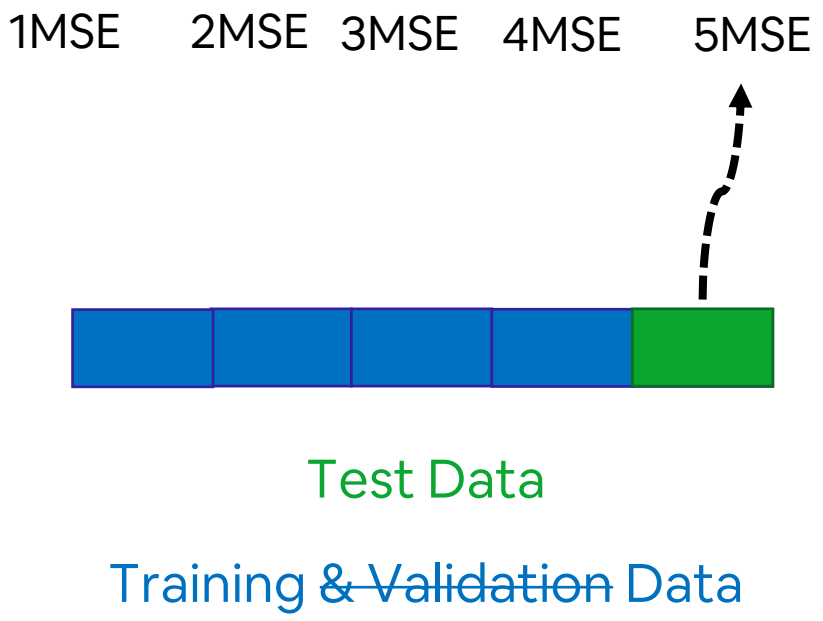
Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold



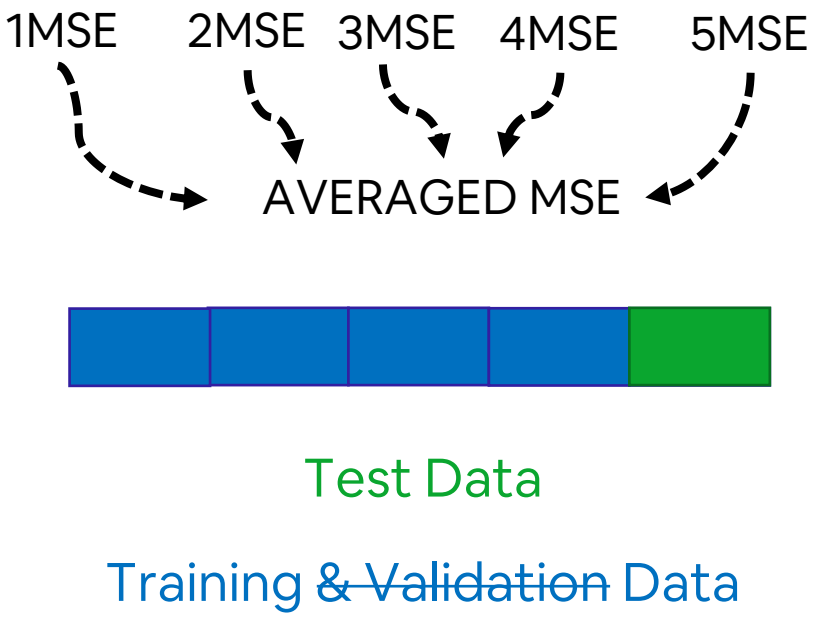
Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold



Cross-Validation

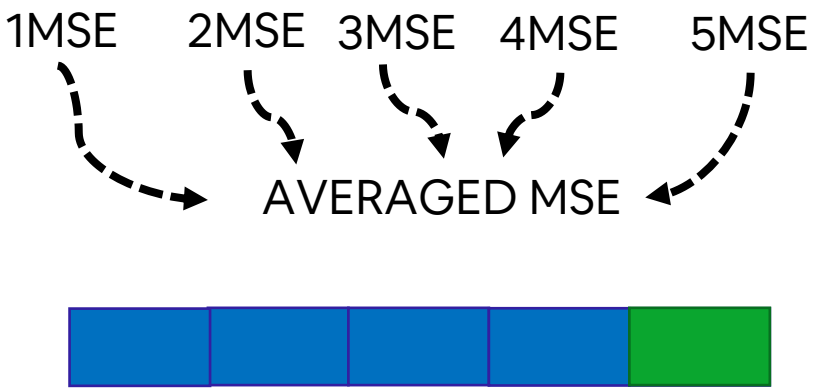
- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold



Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold

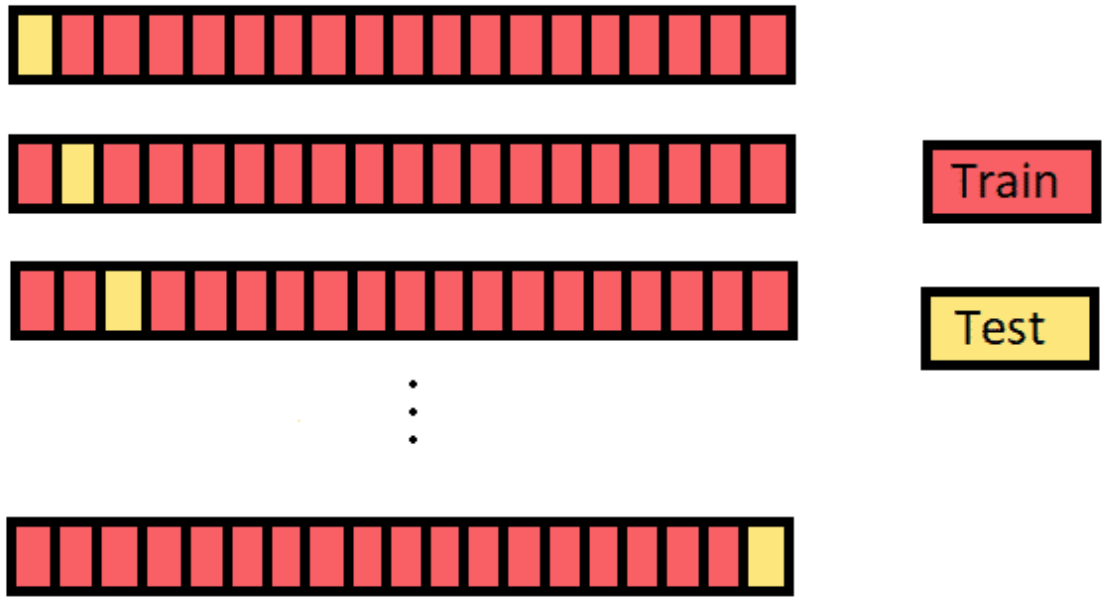
1 design choice:
- # splits (*k*)



Cross-Validation

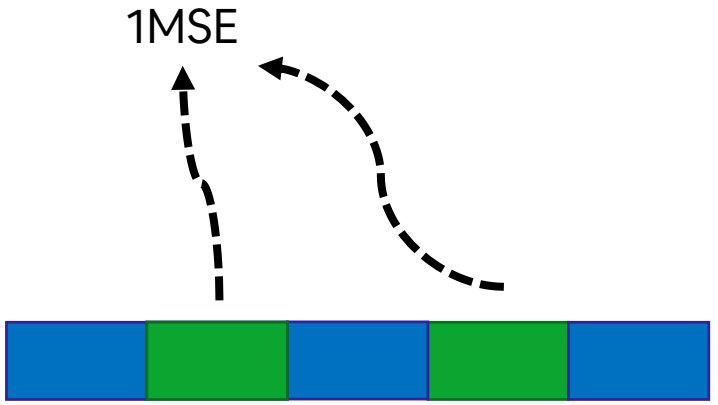
- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold

In the extreme case of k = n, we talk about ‘Leave One Out Cross Validation’



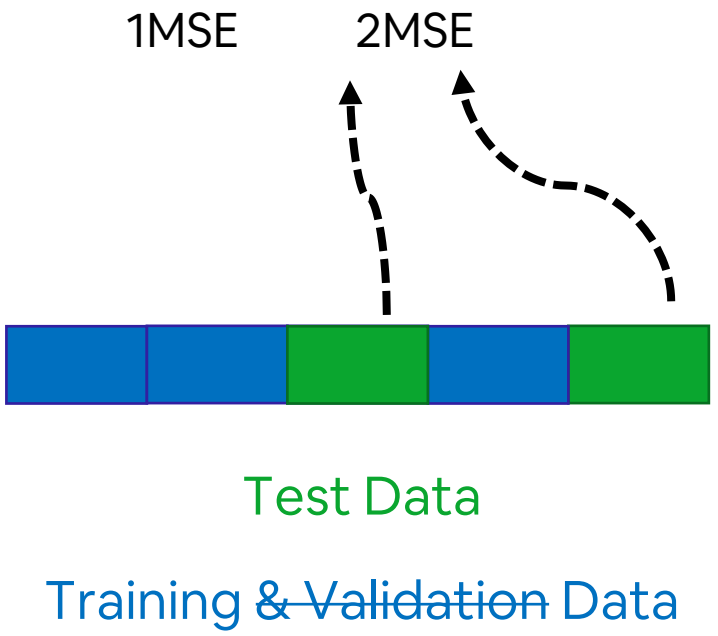
Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold
  - MonteCarlo



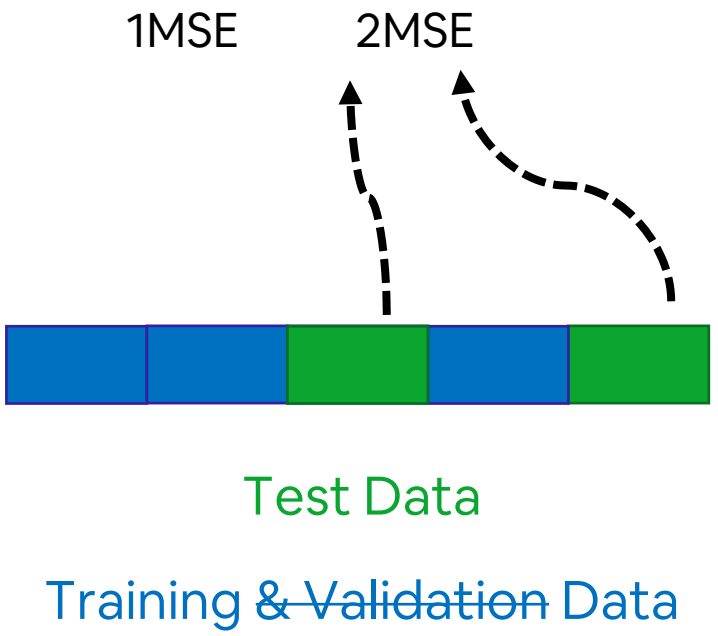
Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold
  - MonteCarlo (MCCV)



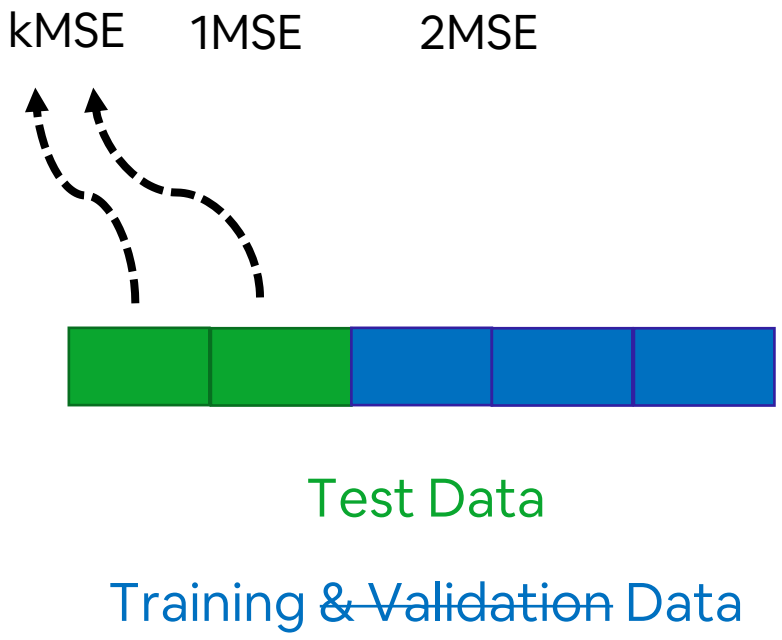
Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold
  - MonteCarlo (MCCV)



Cross-Validation

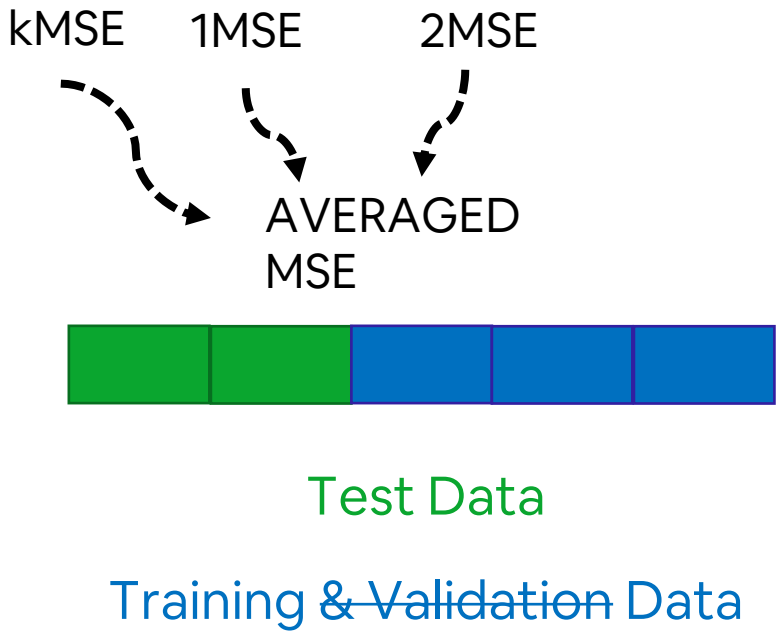
- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold
  - MonteCarlo (MCCV)



Cross-Validation

- To avoid biases in performance evaluation we use cross-validation
- Approaches
  - K-fold
  - MonteCarlo (MCCV)

2 design choices:
- # splits (k)
- ratio of test data (q)



K-fold vs MCCV

✅ Pros

- Lower variance in performance estimation: Since each data point is used exactly once in the test set, the estimated model performance is more stable.
- More efficient: Because it ensures that each data point is used for both training and testing, it typically requires fewer iterations for a reliable estimate.
- More deterministic: If the folds are fixed (~~e.g., using stratification~~), the results are more repeatable compared to MCCV.

❌ Cons

- Less flexibility in train-test splits: The size of training and test sets is fixed and depends on *k*, limiting adaptability.

✅ Pros

- More flexibility in train-test splits: MCCV allows arbitrary train-test splits, which can be useful when data distribution changes over time.
- Good for small datasets: Since the test set can be kept larger, MCCV ensures better generalization testing.
- Multiple random splits reduce bias: Since training and test sets vary across multiple runs, MCCV can better capture different aspects of the dataset.

❌ Cons

- Higher variance: Since the splits are random, some data points might appear more frequently in training sets while others might rarely be in test sets.
- Higher computational cost: Running many iterations of MCCV can be more expensive than k-Fold CV.

Example on California Housing dataset

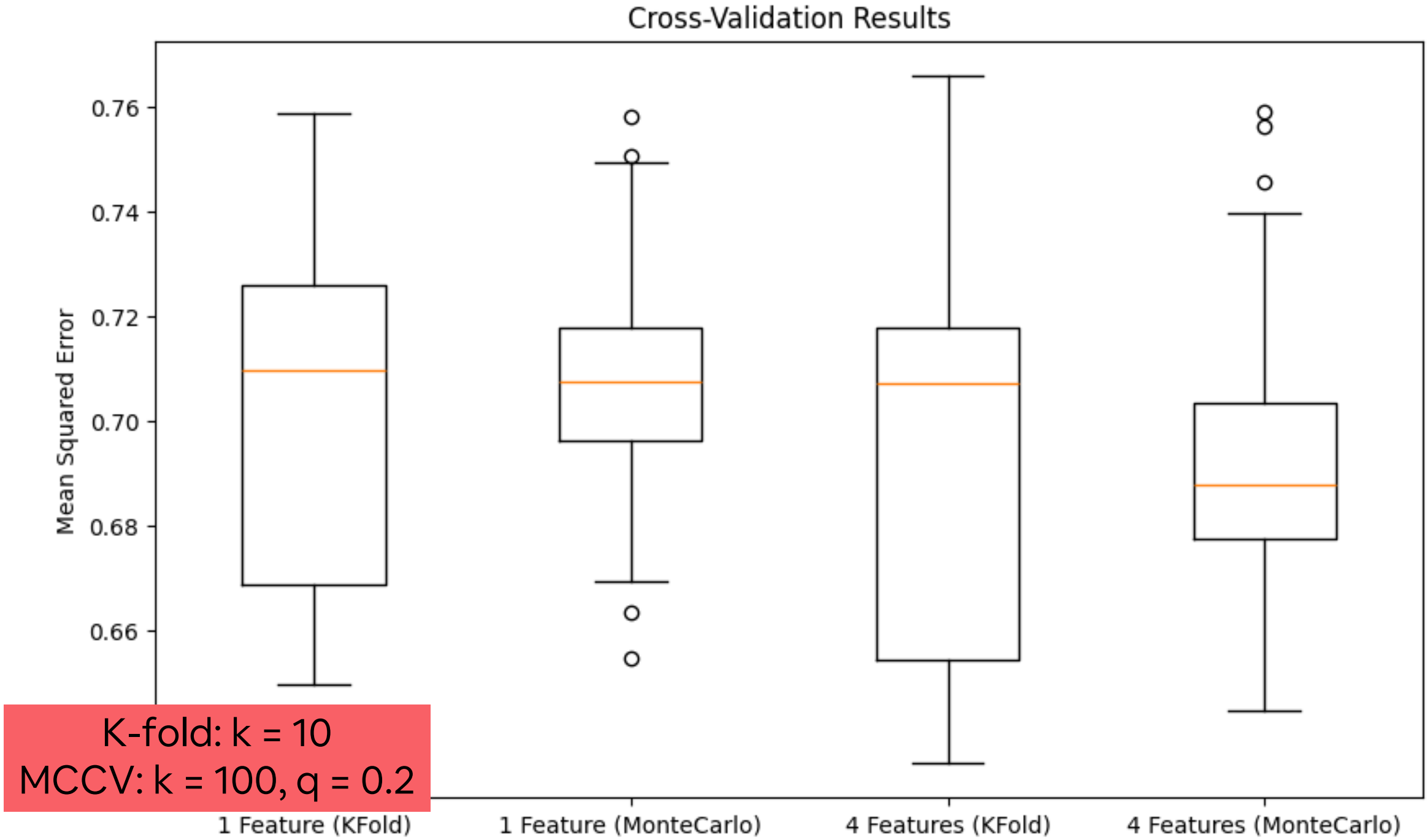
Target (Y):

- MedHouseVal (Median House Value in block group) Represents the median house price in the block group. Measured in hundreds of thousands of dollars (capped at $500,000 in the dataset).

Inputs:*

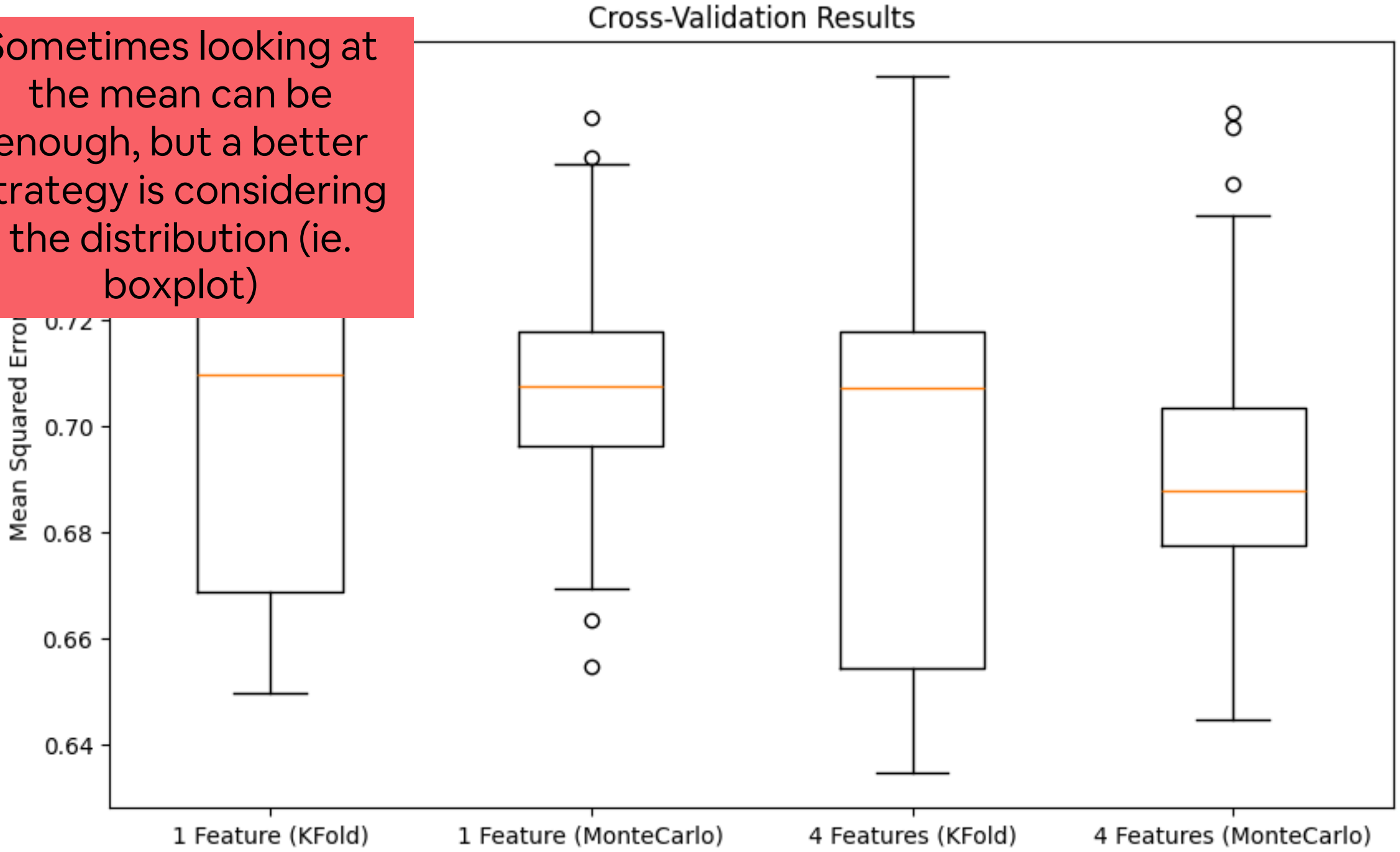
- MedInc (Median Income in block group). Represents the median income of households in the block group. Measured in tens of thousands of dollars. Used in the following in the 1 feature & 4 features case.
- HouseAge (Median House Age in block group). Represents the median age of houses in the area. Measured in years. 4 features case.
- AveRooms (Average Number of Rooms per Dwelling). Computed as the total number of rooms in the block group divided by the number of households. Helps indicate the general size of homes in an area. 4 features case.
- AveOccup (Average Number of Occupants per Household). Computed as the total population in the block group divided by the number of households. 4 features case.

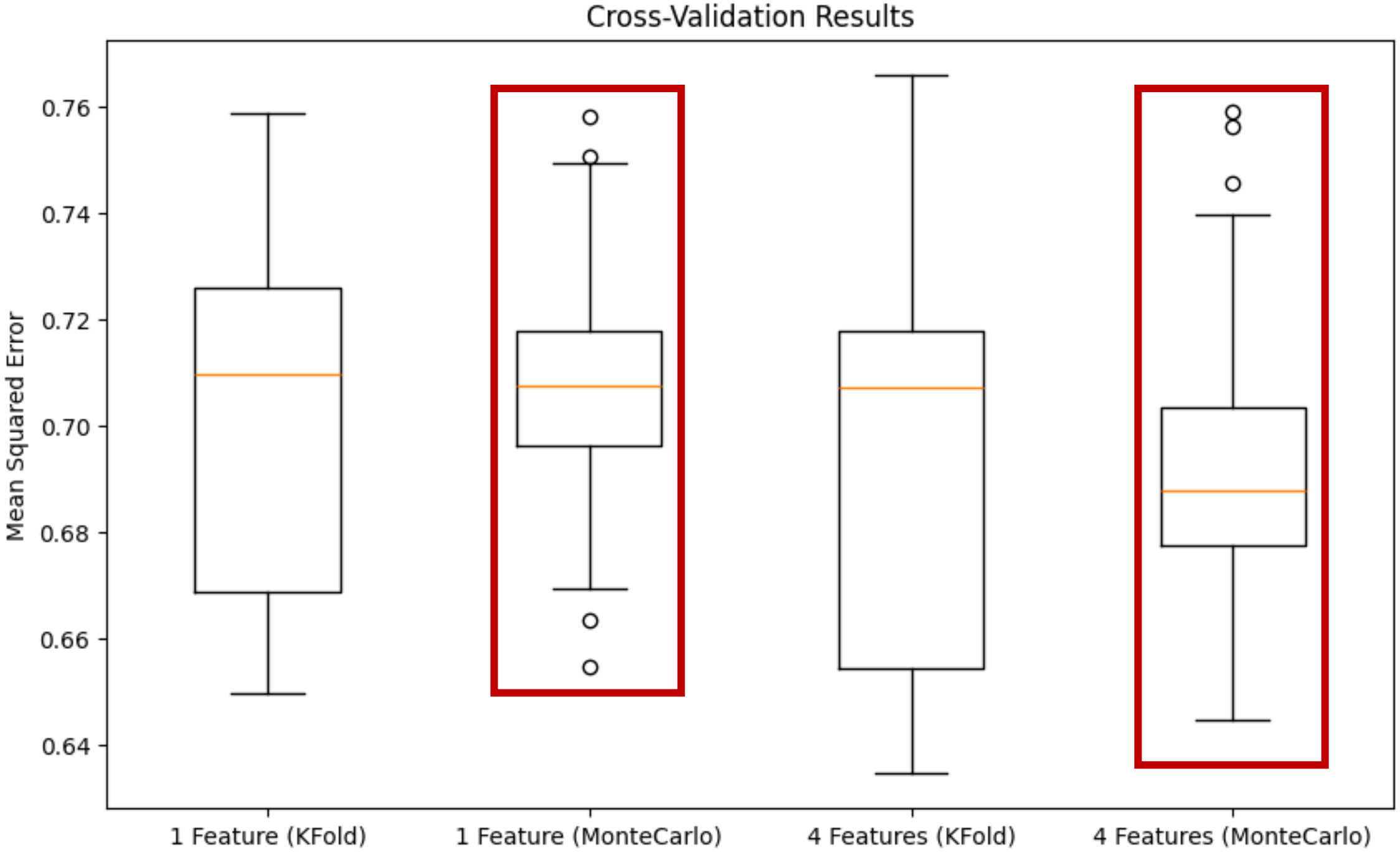
* A subset of the available inputs

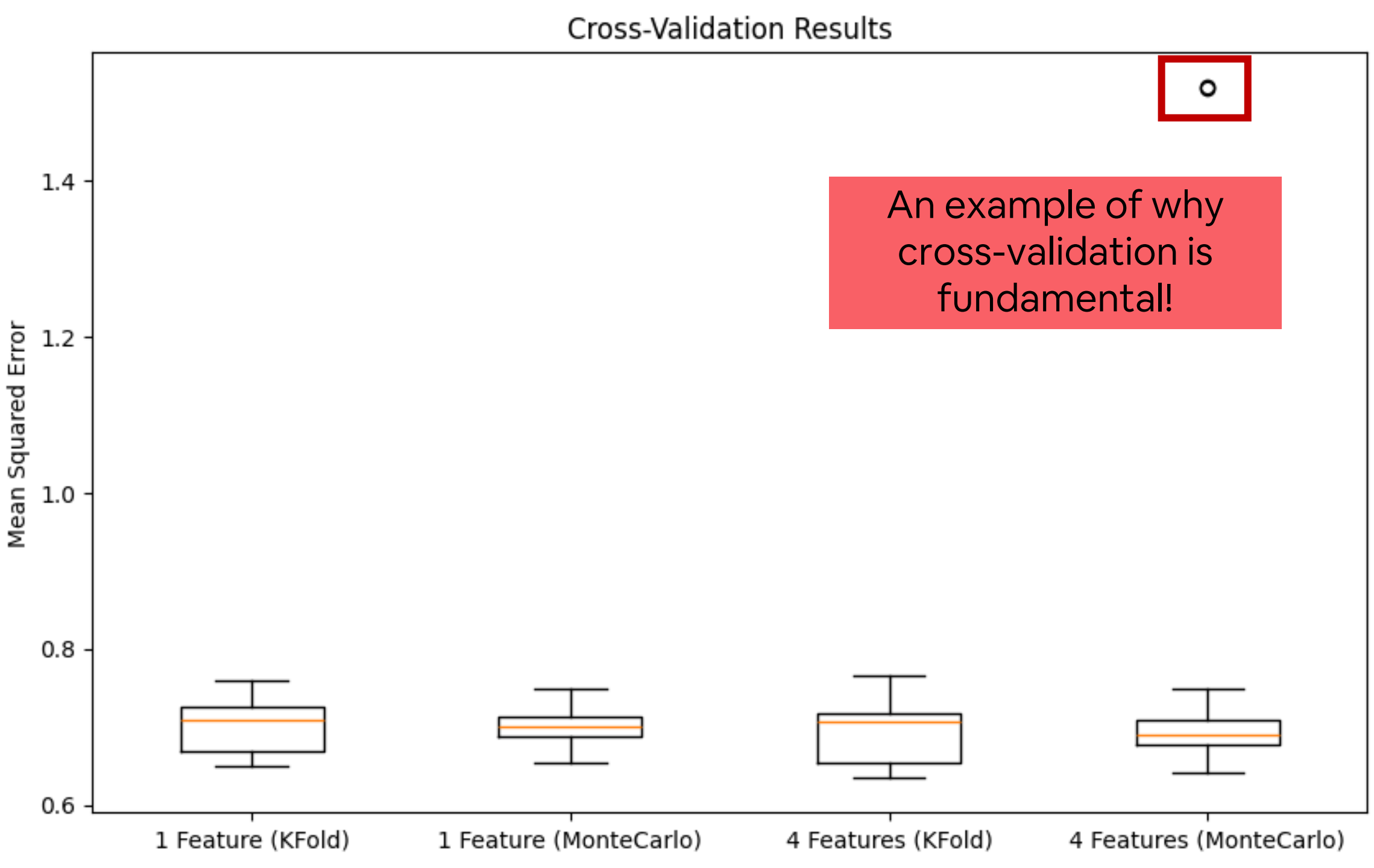


K-fold: k = 10
MCCV: k = 100, q = 0.2

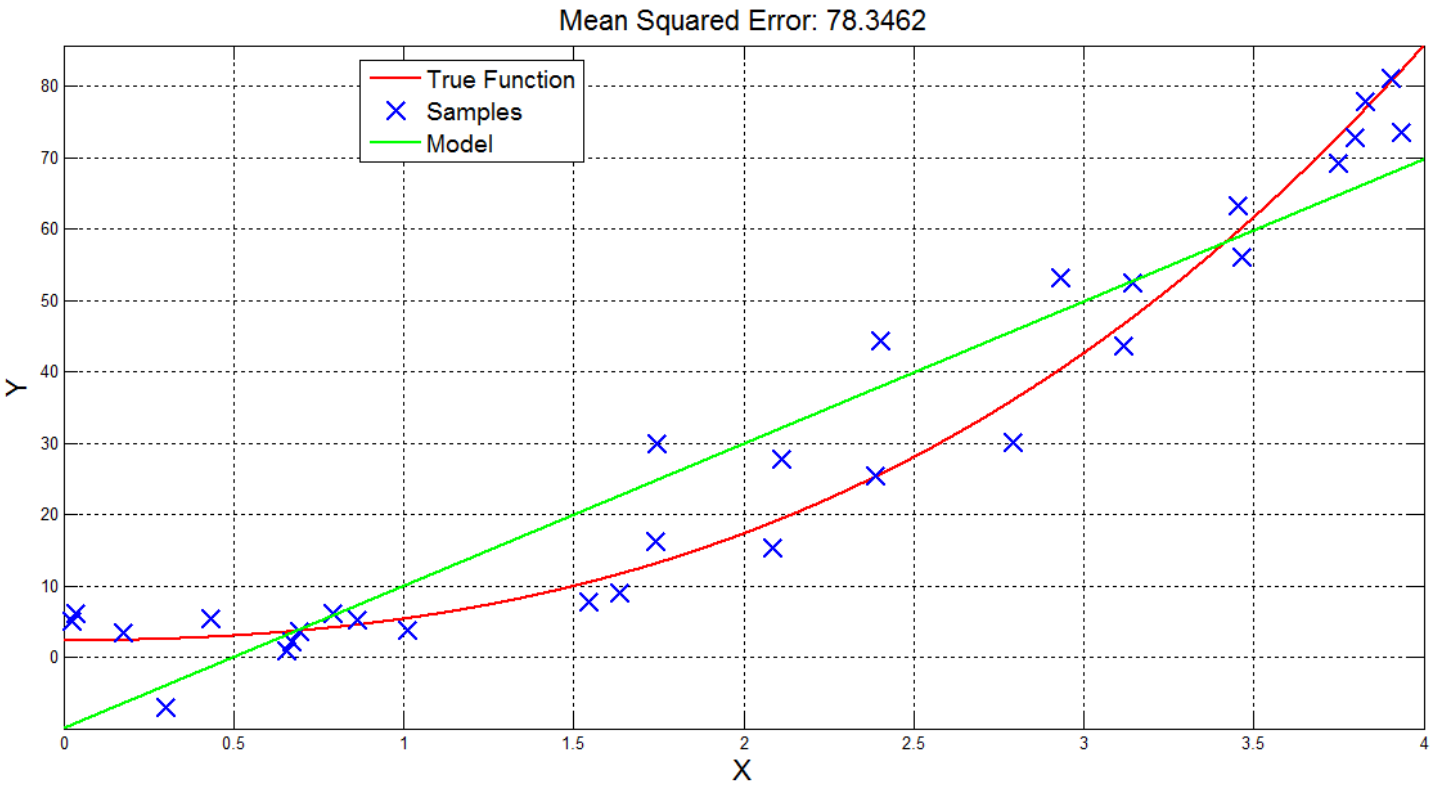
Sometimes looking at the mean can be enough, but a better strategy is considering the distribution (ie. boxplot)




Cross-Validation Results
Mean Squared Error
0.76
0.74
0.72
0.70
0.68
0.66
0.64
1 Feature (KFold)
1 Feature (MonteCarlo)
4 Features (KFold)
4 Features (MonteCarlo)


Cross-Validation Results
Mean Squared Error
1.4
1.2
1.0
0.8
0.6
1 Feature (KFold)
1 Feature (MonteCarlo)
4 Features (KFold)
4 Features (MonteCarlo)
An example of why cross-validation is fundamental!

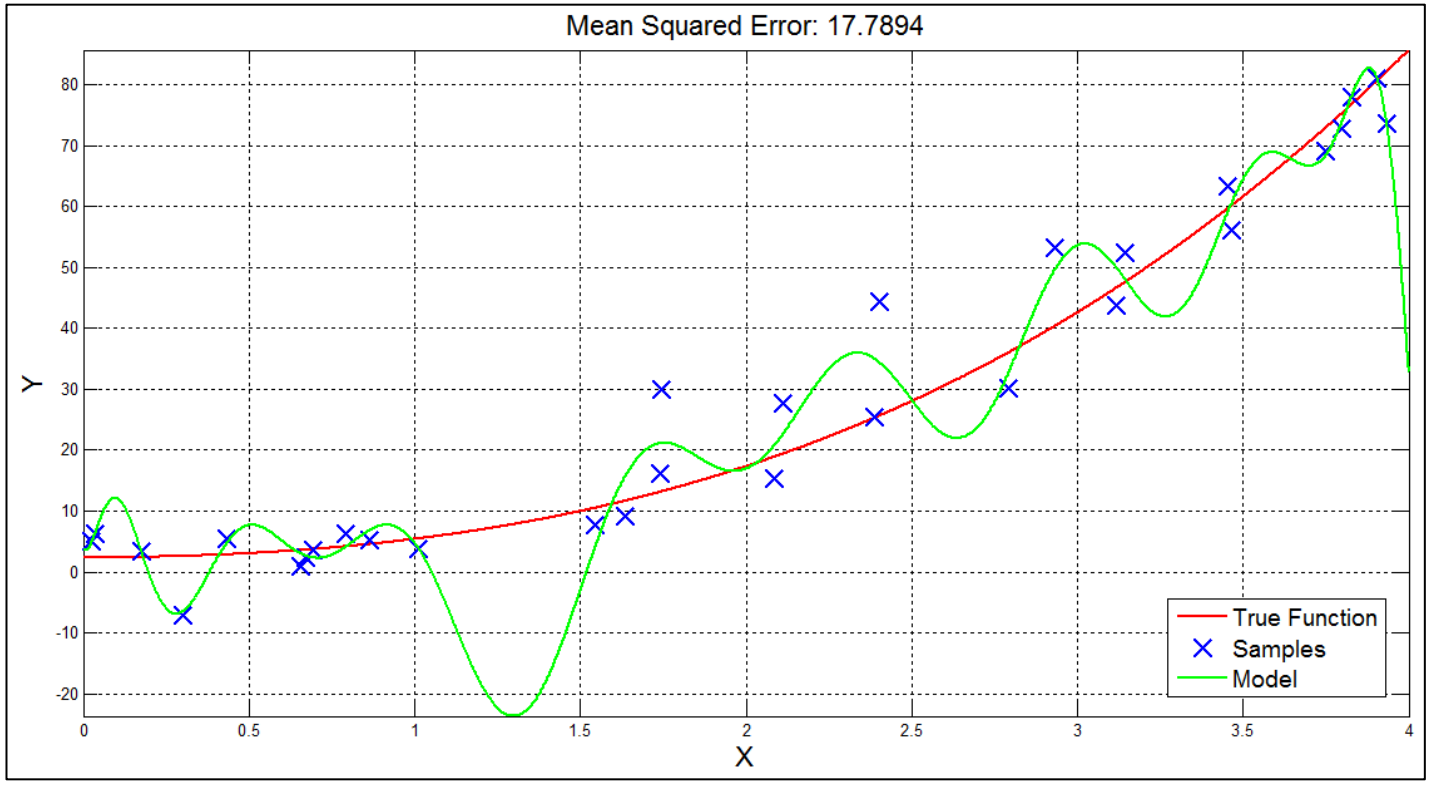
Evaluation: 'Errors' in ML can be of different types: Bias vs Variance



In this example, the linear model without basis extension (1 variable) will never capture the true nature of the underlying phenomena.

The inability for a ML method to capture the true relationship is called bias.

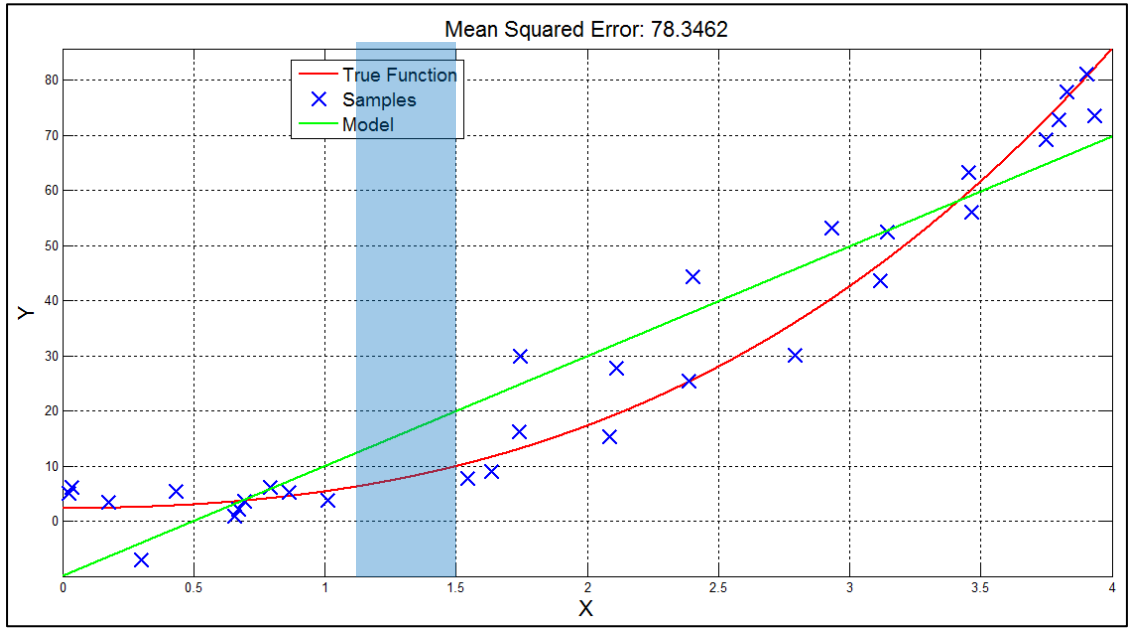
Evaluation: 'Errors' in ML can be of different types: Bias vs Variance

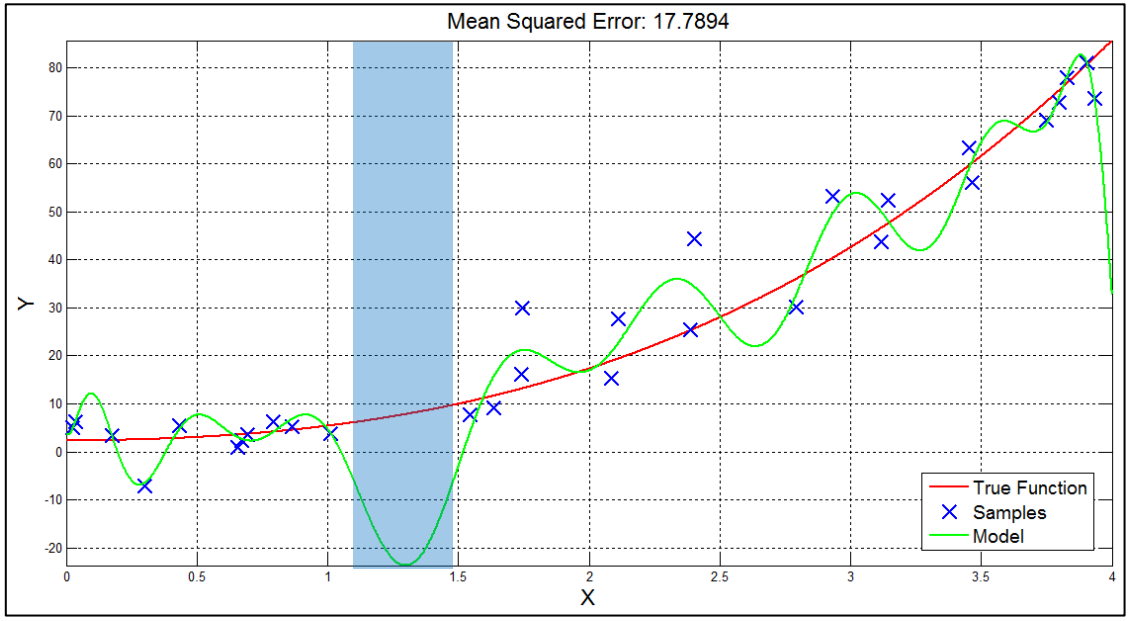


This model (20 variables!) instead has very low bias, as it is able to adapt to available data.

The inability for a ML method to capture the true relationship is called bias.

Evaluation: 'Errors' in ML can be of different types: Bias vs Variance



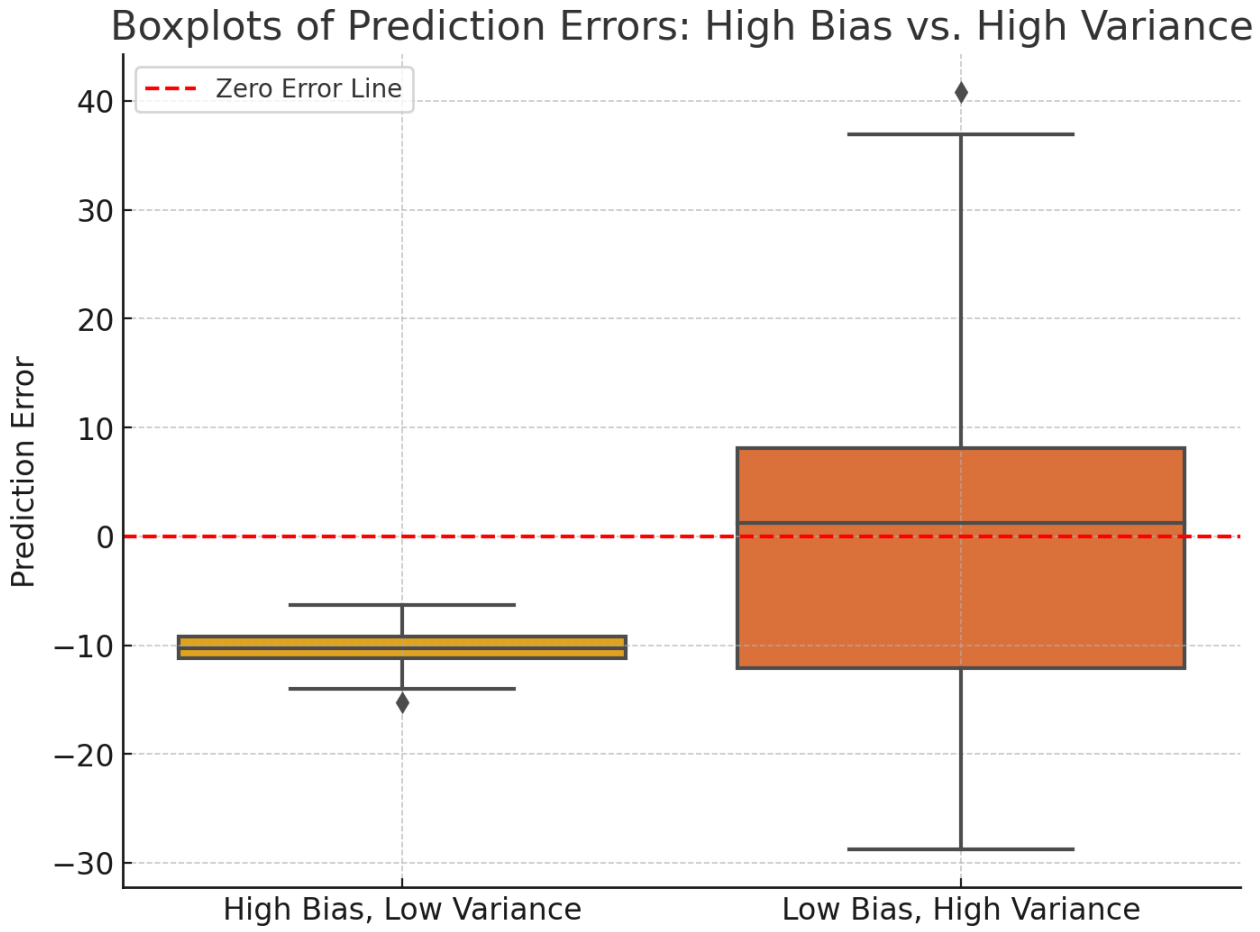


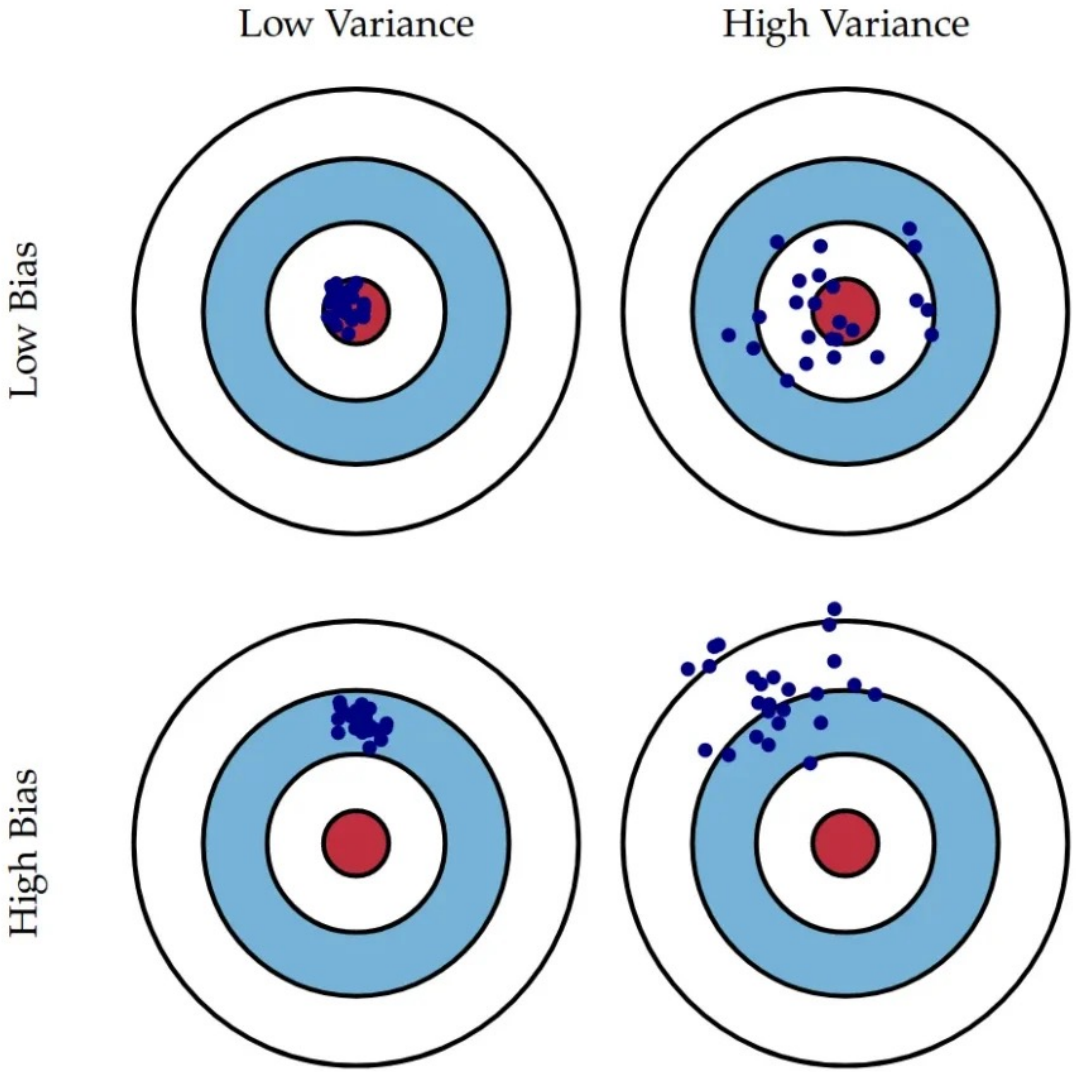
We can try to cross-validate: it is likely that both models will get 'bad' results, but of different nature.

[Right model] While the model was pretty good on training data (low bias), it will become bad on testing ones: this difference in performance on different datasets, will make the model on the right a high-variance model (with a lot of variability).

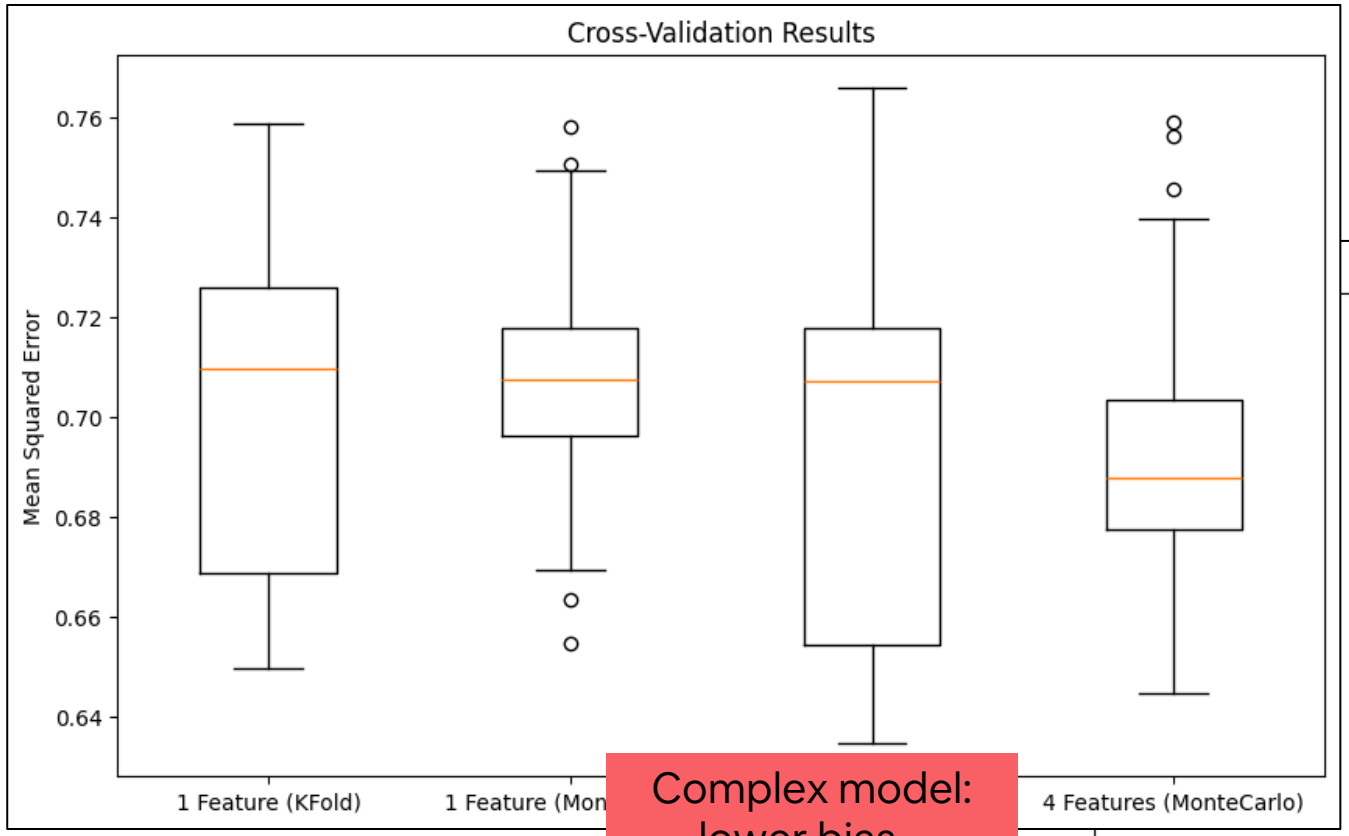
[Left model] As said, this is a high bias model, but it is consistent across folds: this is a low-variance model.

Evaluation: ’Errors’ in ML can be of different types: Bias vs Variance

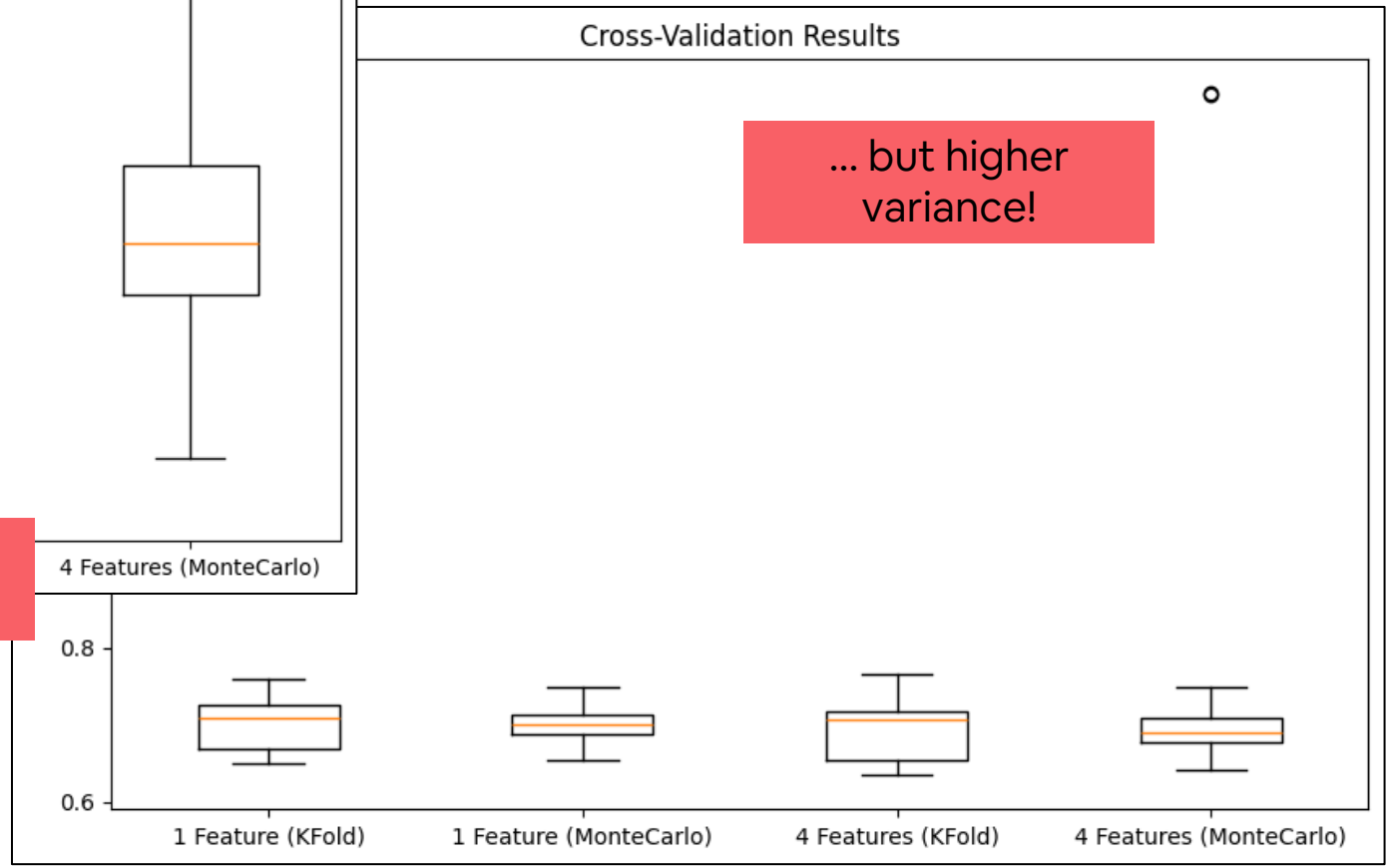




Evaluation: 'Errors' in ML can be of different types: Bias vs Variance



Complex model: lower bias...

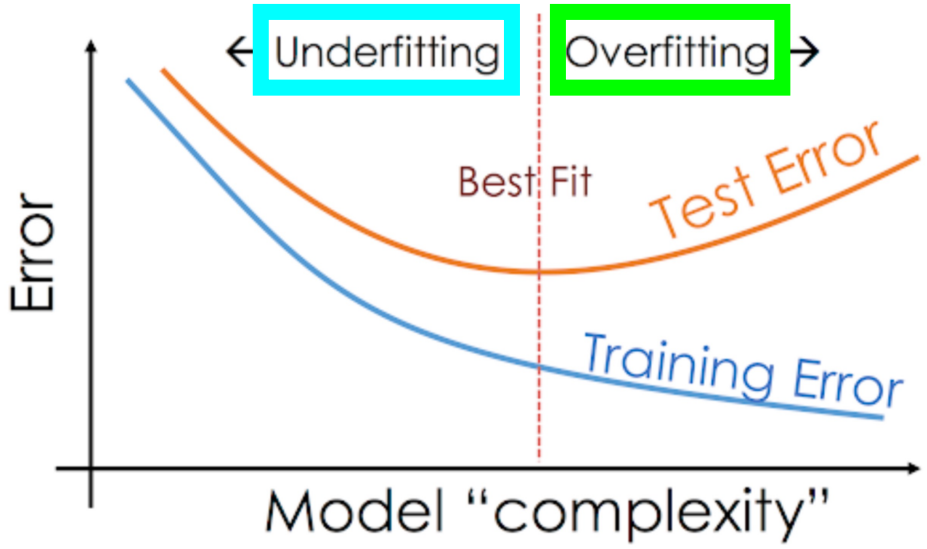


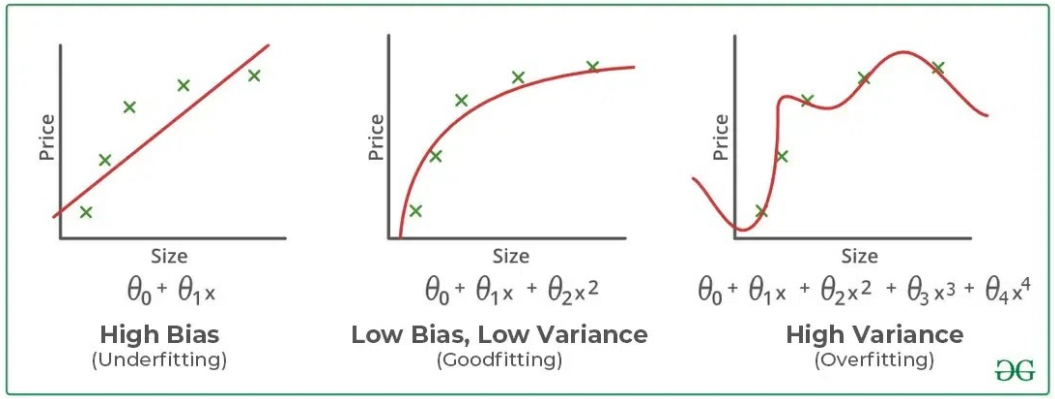
Bias & Variance – Underfitting & Overfitting

A complicated model is not always 'optimal':

- Overly complex models tend to 'overfit', meaning they fail to 'generalize.' This results in high variance, where the model captures noise rather than true patterns in the data.
- On the other hand, very simple models lead to 'underfitting', where the available data is not fully leveraged ('the model has learned too little'). This is associated with high bias, as the model is too simplistic to capture the underlying structure of the data.

The key is to find a balance between bias and variance to achieve good generalization.



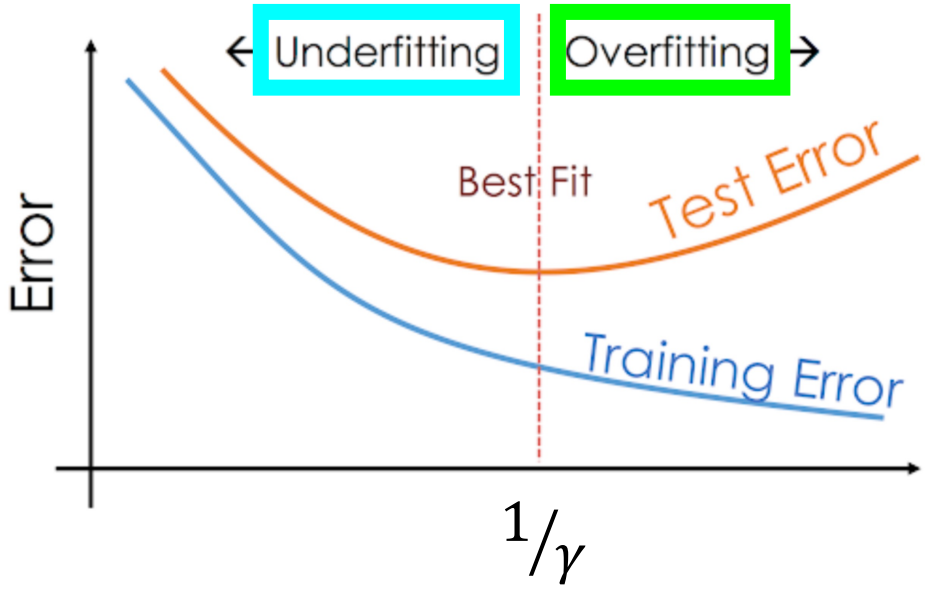


Regularization finds this balance in the modelling!

- There are more ‘formal’ approaches that allow for optimized management of the trade-off between underfitting and overfitting.
- Regularization is a technique used in ML to prevent overfitting by adding a penalty term to the loss function of a model.
- In linear regression, this is achieved by ‘simply’ changing the cost function: β s.t.

$$J = \sum_{i=1}^{n} [y^{(i)} - \widehat{y^{(i)}}]^2 + \gamma R,$$

R is a penalty on model complexity.



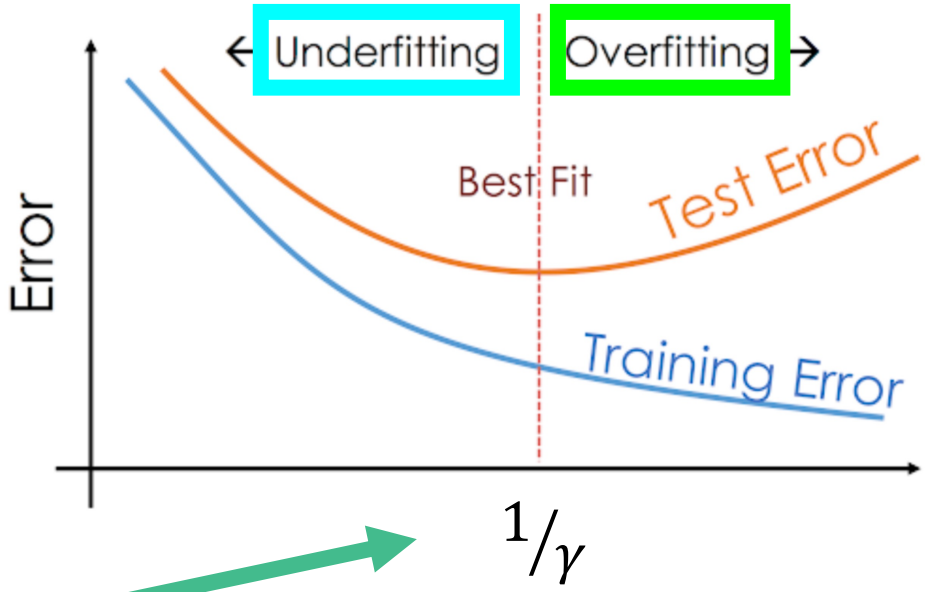
Regularization finds this balance in the modelling!

- There are more 'formal' approaches that allow for optimized management of the trade-off between underfitting and overfitting.
- Regularization is a technique used in ML to prevent overfitting by adding a penalty term to the loss function of a model.
- In linear regression, this is achieved by 'simply' changing the cost function: β s.t.

$$J = \sum_{i=1}^{n} [y^{(i)} - \widehat{y^{(i)}}]^2 + \gamma R,$$

R is a penalty on model complexity.

Regularization parameter
(this is an hyperparameter)



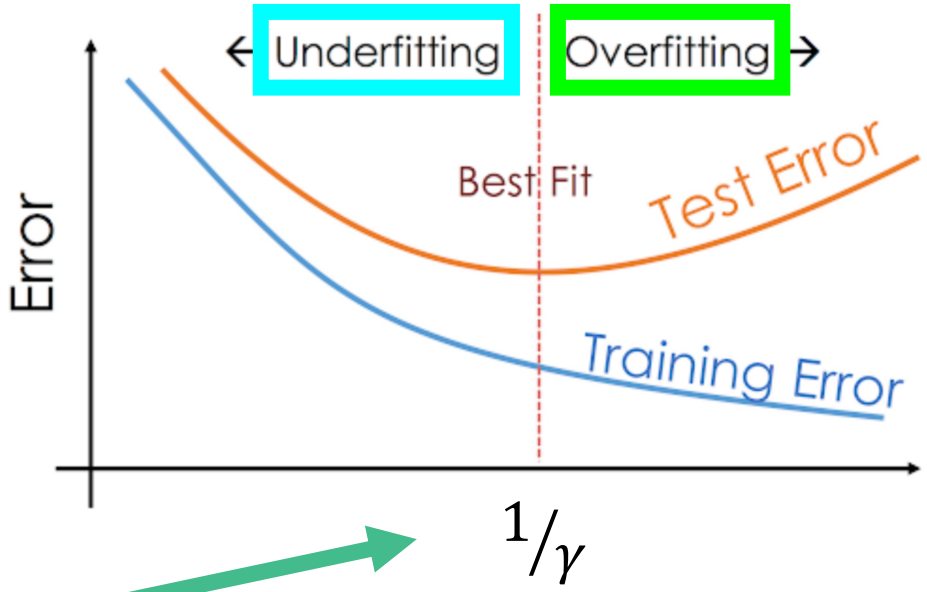
Regularization finds this balance in the modelling!

- There are more 'formal' approaches that allow for optimized management of the trade-off between underfitting and overfitting.
- Regularization is a technique used in ML to prevent overfitting by adding a penalty term to the loss function of a model.
- In linear regression, this is achieved by 'simply' changing the cost function: β s.t.

$$J = \sum_{i=1}^{n} [y^{(i)} - \widehat{y^{(i)}}]^2 + \gamma R,$$

R is a penalty on model complexity.

Regularization parameter
(this is an hyperparameter)



If $R = \sum_{j=0}^{p} \beta_j^{\,2}$ (L2 penalization) we are dealing with Ridge Regression

Regularization finds this balance in the modelling!

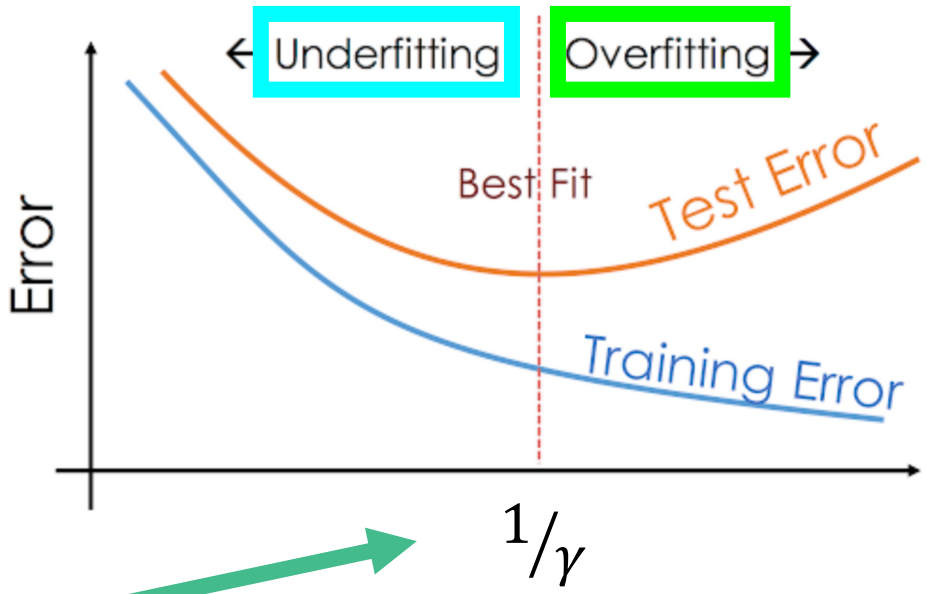
Based on a test dataset, it is possible to choose the hyperparameter value to achieve the best trade-off between model complexity and accuracy

- Regularization is a technique used in ML to prevent overfitting by adding a penalty term to the loss function of a model.
- In linear regression, this is achieved by 'simply' changing the cost function: β s.t.

$$J = \sum_{i=1}^{n} [y^{(i)} - \widehat{y^{(i)}}]^2 + \gamma R,$$

R is a penalty on model complexity.

Regularization parameter (this is an hyperparameter)



If $R = \sum_{j=0}^{p} \beta_j^{\,2}$ (L2 penalization) we are dealing with Ridge Regression

Ridge Regression (RR): closed-form solution

OLS:

$$\min_\beta \sum_{i=1}^{n} (y_i - x_i\beta)^2 = (Y - X\beta)^T(Y - X\beta)$$

$$\Rightarrow \beta = (X^T X)^{-1} X^T Y$$

Ridge regression (trivial derivation*):

$$\min_\beta (Y - X\beta)^T(Y - X\beta) + \lambda\beta^2$$

$$\Rightarrow \beta = (X^T X + \lambda I)^{-1} X^T Y$$

* If you understood OLS...

Ridge Regression (RR): closed-form solution

OLS:

$$\min_\beta \sum_{i=1}^{n}(y_i - x_i\beta)^2 = (Y - X\beta)^T(Y - X\beta)$$

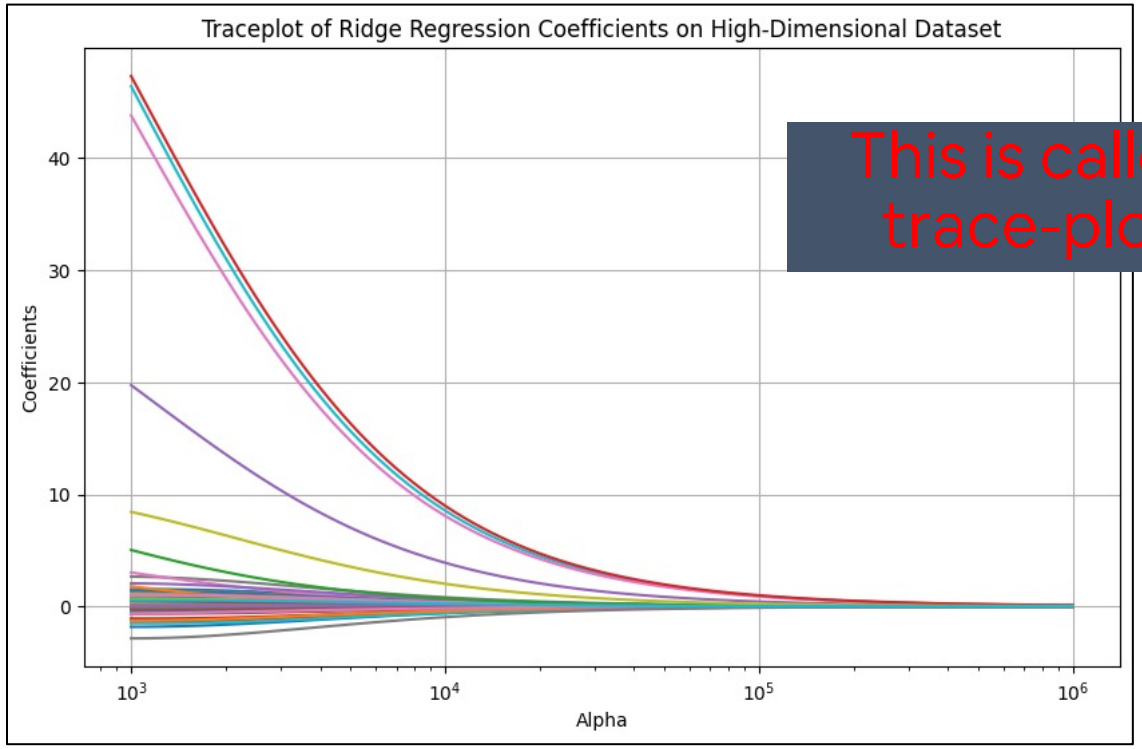
$$\Rightarrow \beta = (X^T X)^{-1} X^T Y$$

Ridge regression (trivial derivation*):

$$\min_\beta (Y - X\beta)^T(Y - X\beta) + \lambda\beta^2$$

$$\Rightarrow \beta = (X^T X + \lambda I)^{-1} X^T Y$$

* If you understood OLS...



This is called trace-plot

- For lambda = 0, OLS = RR
- At the increase of lambda (reported also as alpha), the coefficients shrink
- With high collinearity in the dataset the robustness improve

Ridge Regression (RR): closed-form solution

- If datasets have high-collinearity, you could end up with OLS solutions that are highly sensitive

  y = 100x1 – 99x2 + …

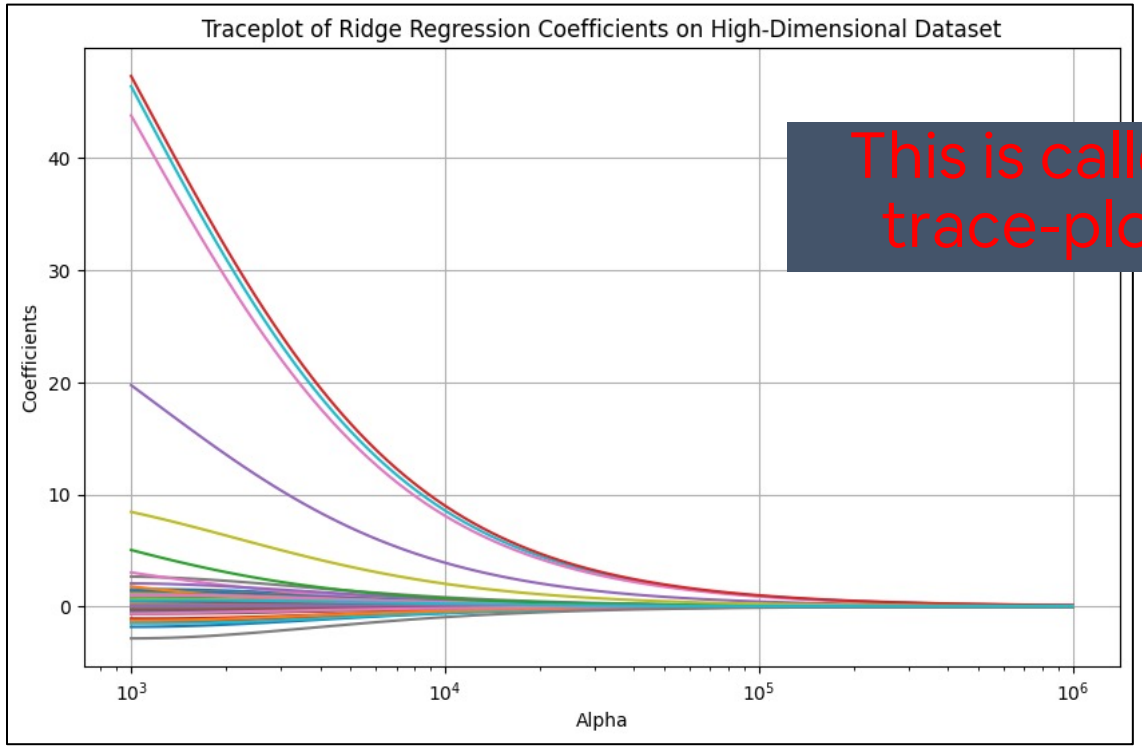
- RR avoids that! Let's see it in an algebra perspective…

Ridge regression (trivial derivation*):

$$\min_\beta (Y - X\beta)^T (Y - X\beta) + \lambda\beta^2$$

$$\Rightarrow \beta = (X^T X + \lambda I)^{-1} X^T Y$$

* If you understood OLS…



This is called trace-plot

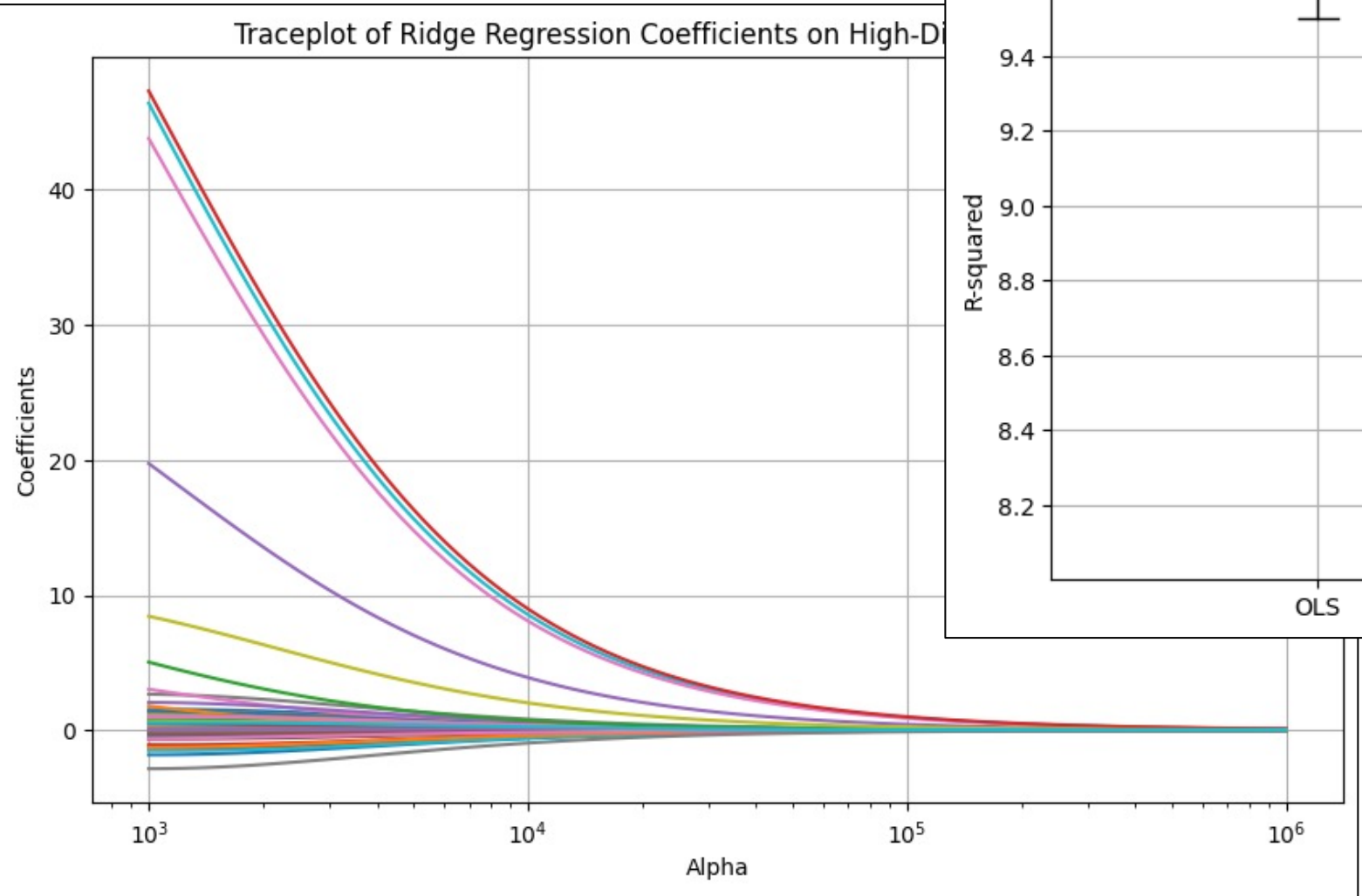
- For lambda = 0, OLS = RR
- At the increase of lambda (reported also as alpha), the coefficients shrink
- With high collinearity in the dataset the robustness improve

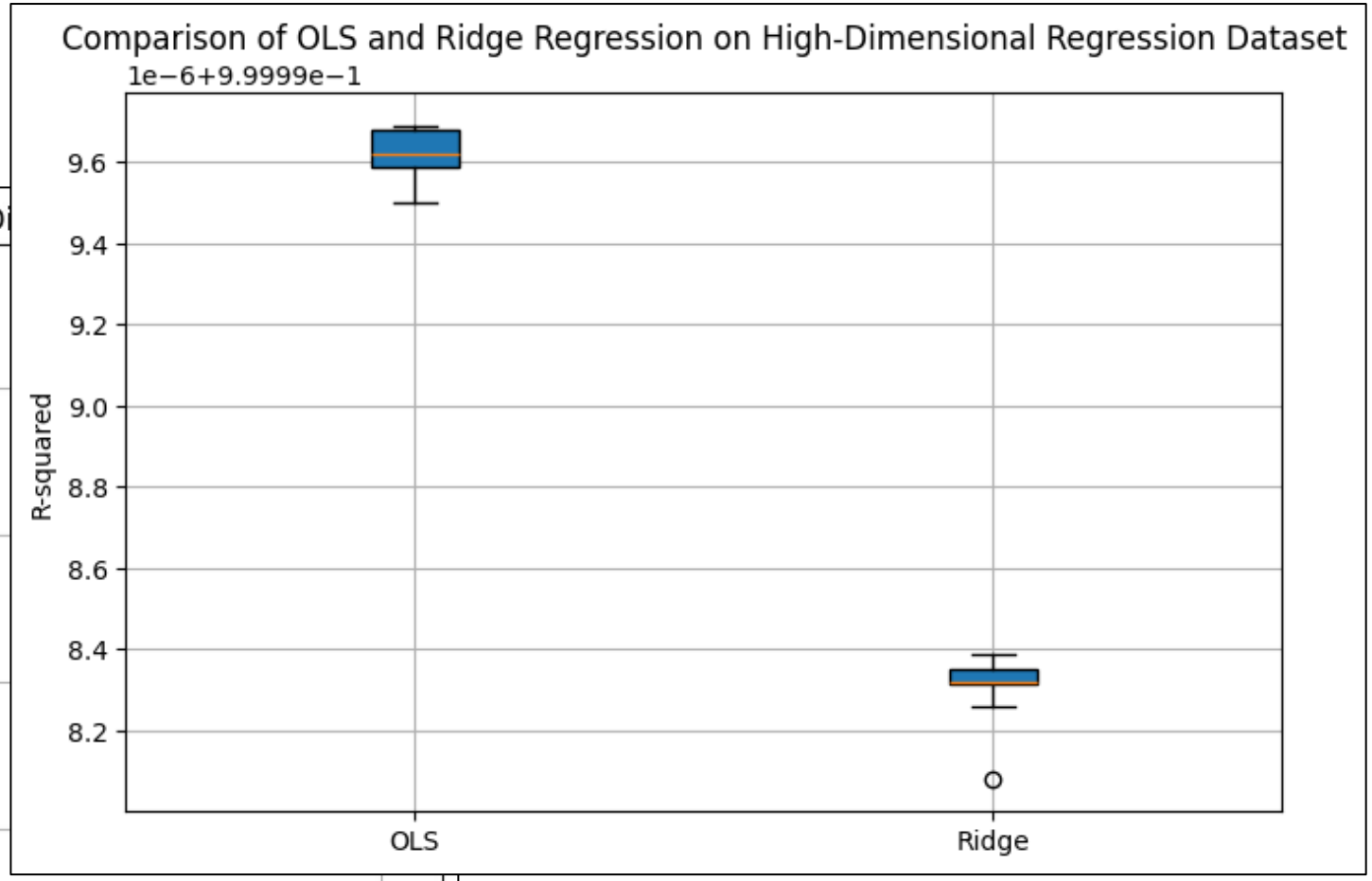
Ridge Regression (RR): closed-form solution

- Ridge Regression helps when X′X (the Gram matrix in least squares) is ill-conditioned or nearly singular.
- This condition occurs when the predictors are highly correlated, leading to unstable estimates in Ordinary Least Squares (OLS), this means that X'X has some eigenvalues close to zero
- The inverse $(X'X)^{-1}$ then became unstable or nearly singular, leading to large variance in coefficient estimates: small changes in data can cause large swings in estimated coefficients, making predictions unstable.
- RR, by adding the term λI adds a small value to the diagonal of the matrix we need to invert, shift all eigenvalues away from zero, improving numerical stability

$$\beta_{ridge} = (X'X + \lambda I)^{-1} X'Y$$

Example: a high dimensional data with p = 50





Nested CV for Hyperparameter Tuning

The Problem with Standard Cross-Validation for Hyperparameter Tuning

- If we perform hyperparameter tuning using cross-validation (e.g., grid search, random search), we typically select the best-performing model based on its performance on a validation set.

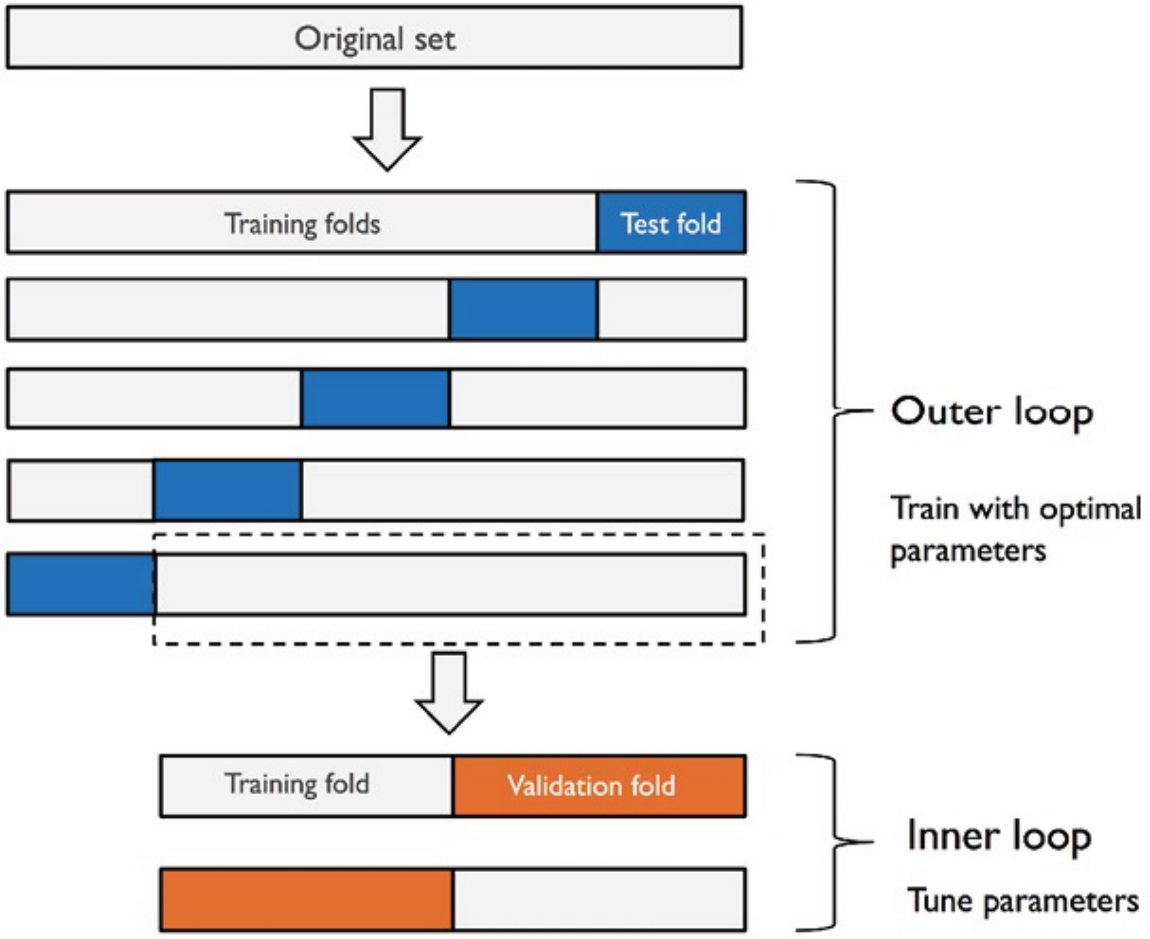
- Then, we evaluate this best model on a separate test set.

- The issue: The test set has indirectly influenced model selection, leading to optimistic performance estimates.

Nested CV for Hyperparameter Tuning

Nested cycle of CV

1. Inner
   - Training data for model construction
   - Validation data for choosing the hyperparameter(s)
2. Outer
   - Training data (training+validation) for model building
   - Test data for performance evaluation



Bonus: Occam’s Razor

‘We consider it a good principle to explain the phenomena by the simplest hypothesis possible’

Regularization embodies the principle of Occam’s Razor, which suggests that simpler solutions are generally better than complex ones.

William of Occam (Occam 1288 – Munich 1347)

Thank you!

Gian Antonio Susto