# PSY9511: Seminar 4

## Model selection, validation and testing

Esten H. Leonardsen

23.09.24

UNIVERSITETET
I OSLO

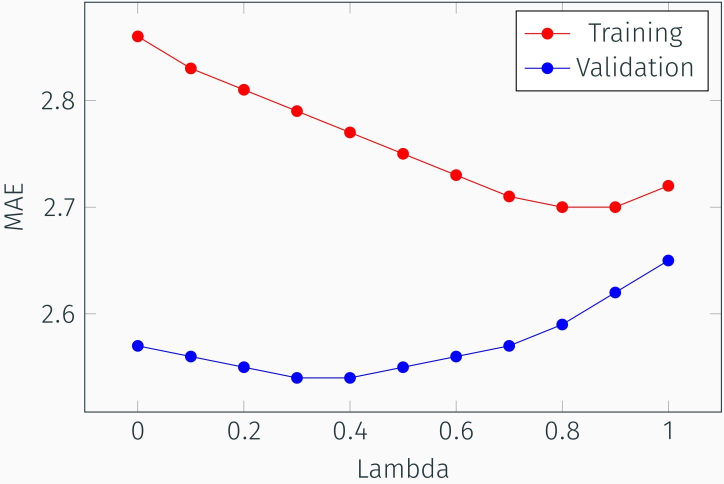
# Outline

1. Assignment 3
2. Loss functions and performance metrics
3. Strategies for model evaluation
    - Training and validation split
    - (Stratification)
    - (Leave-one-out cross-validation)
    - Cross-validation
    - Bootstrap
    - Model comparison
4. Strategies for model selection **and** evaluation
    - Train/validation/test split
    - Nested cross-validation

# Assigment 3

UNIVERSITETET
I OSLO

# Assignment 3: Performance



# Assignment 3: Preprocessing

Training

| x |
|---|
| 0.0 |
| 0.2 |
| 0.4 |
| 0.6 |
| 0.8 |
| 1.0 |

Validation

| x |
|---|
| 1.0 |
| 1.2 |
| 1.4 |
| 1.6 |
| 1.8 |
| 2.0 |

# Assignment 3: Preprocessing

Training

| x |
|---|
| 0.0 |
| 0.2 |
| 0.4 |
| 0.6 |
| 0.8 |
| 1.0 |

$\bar{x} = 0.5$

Validation

| x |
|---|
| 1.0 |
| 1.2 |
| 1.4 |
| 1.6 |
| 1.8 |
| 2.0 |

$\bar{x} = 1.5$

# Assignment 3: Preprocessing

Training

| x |
|---|
| 0.0 |
| 0.2 |
| 0.4 |
| 0.6 |
| 0.8 |
| 1.0 |

$\bar{x} = 0.5$

Validation

| x |
|---|
| 1.0 |
| 1.2 |
| 1.4 |
| 1.6 |
| 1.8 |
| 2.0 |

$\bar{x} = 1.5$

In a real-life scenario we don't have a validation set, only individual observations

# Assignment 3: Preprocessing

Training

| x |
|---|
| 0.0 |
| 0.2 |
| 0.4 |
| 0.6 |
| 0.8 |
| 1.0 |

$\bar{x} = 0.5$

Validation

| x |
|---|
| 1.0 |
| 1.2 |
| 1.4 |
| 1.6 |
| 1.8 |
| 2.0 |

$\bar{x} = 1.5$

# Assignment 3: Preprocessing

Training

| x |
|---|
| 0.0 |
| 0.2 |
| 0.4 |
| 0.6 |
| 0.8 |
| 1.0 |

↓

| x |
|---|
| -0.5 |
| -0.3 |
| -0.1 |
| 0.1 |
| 0.3 |
| 0.5 |

Validation

| x |
|---|
| 1.0 |
| 1.2 |
| 1.4 |
| 1.6 |
| 1.8 |
| 2.0 |

↓

| x |
|---|
| -0.5 |
| -0.3 |
| -0.1 |
| 0.1 |
| 0.3 |
| 0.5 |

# The mortal sins of machine learning in neuroscience

1. Reporting inflated modelling performance due to insufficient testing procedures
2. Using accuracy as a model performance measure when it is not appropriate

# The mortal sins of machine learning in neuroscience

1. Reporting inflated modelling performance due to insufficient testing procedures
2. Using accuracy as a model performance measure when it is not appropriate

↓

We believe that we know things that we actually don’t (e.g. what brain regions are involved in a mental disorder)

# Loss functions and performance metrics

UNIVERSITETET
I OSLO

# Loss functions and performance metrics

### Commonalities

- Allows us to evaluate the performance of a model
- Typically on the form $f(y, \hat{y})$

### Loss functions

- Tailored specifically for mathematical optimization of models

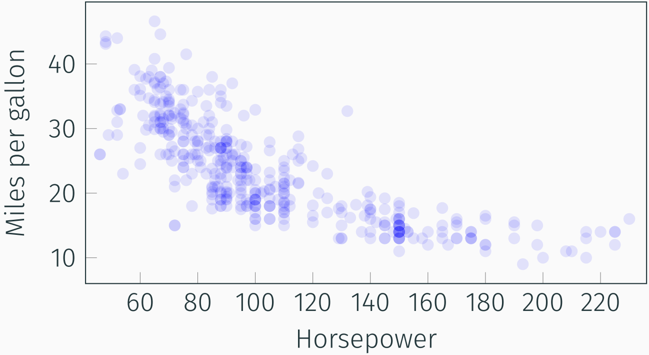
### Performance metrics

- Tailored specifically for interpretation of model performance by humans

# Loss functions and performance metrics

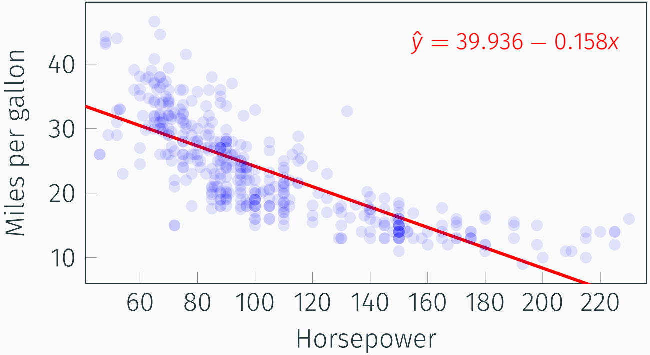
$$\mathrm{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

# Loss functions and performance metrics



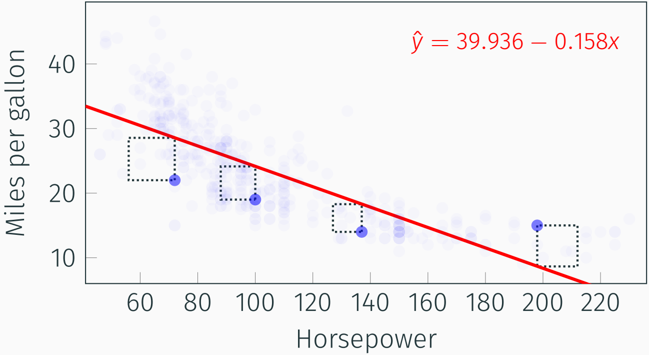
$$\mathrm{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

# Loss functions and performance metrics



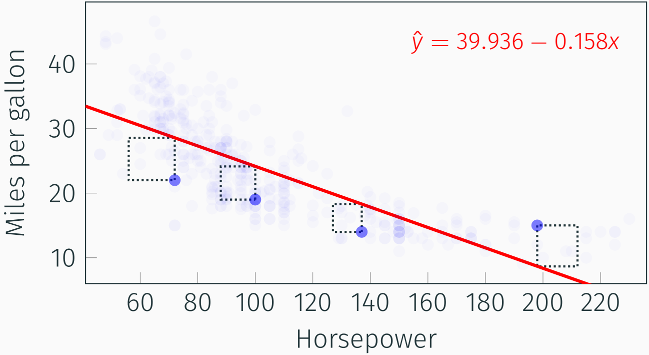
$$\mathrm{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

# Loss functions and performance metrics



$$\text{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

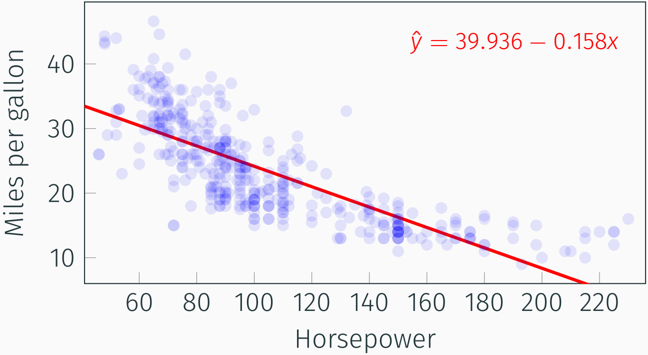
# Loss functions and performance metrics



$$\text{mse}(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

$$= 23.94$$

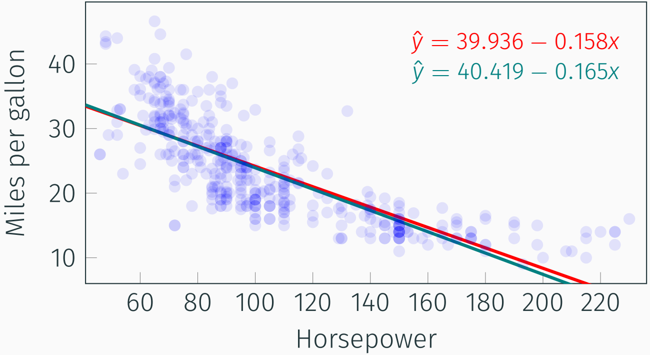
# Loss functions and performance metrics



$$\mathrm{mse}(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

$$\mathrm{mae}(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$$

# Loss functions and performance metrics



$$\text{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

$$\text{mae}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} |y_i - \hat{y}_i|$$

# Loss functions and performance metrics

## Loss functions

- Different loss functions measures different properties of the model fit
- Optimizing for them gives different parameter estimates

# Loss functions and performance metrics

**Tolerance-based accuracy:**

A prediction is considered correct if it is within a predefined margin of error from the true value

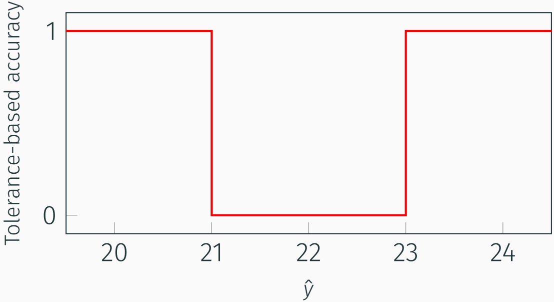
$$\text{accuracy*}(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| < \text{tol} \\ 1 & else \end{cases}$$

# Loss functions and performance metrics

| mpg | horsepower |
|---|---|
| 22 | 72 |

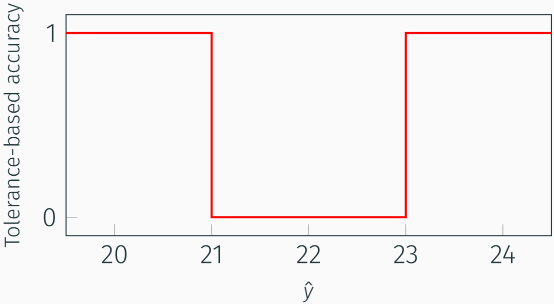
# Loss functions and performance metrics

| mpg | horsepower |
|---|---|
| 22 | 72 |



# Loss functions and performance metrics

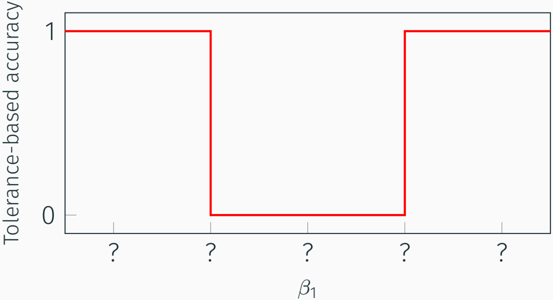
| mpg | horsepower |
|---|---|
| 22 | 72 |



$$\hat{y} = \beta_0 + \beta_1 \times \text{horsepower}$$

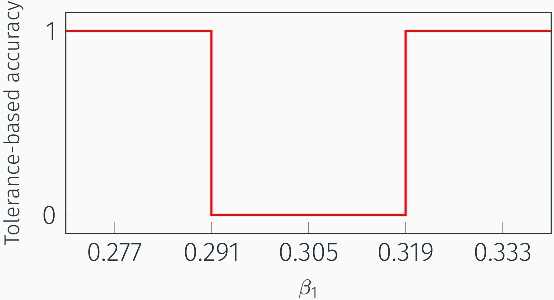
# Loss functions and performance metrics

| mpg | horsepower |
|---|---|
| 22 | 72 |



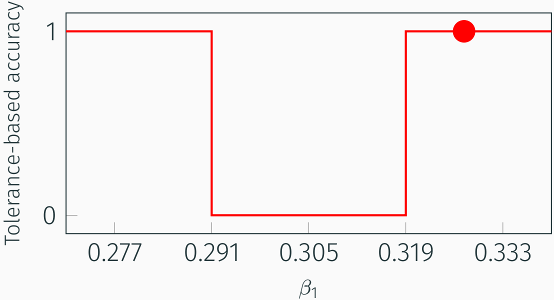
$$\hat{y} = \beta_0 + \beta_1 \times \text{horsepower}$$

# Loss functions and performance metrics



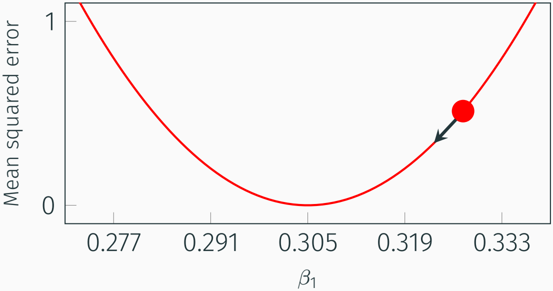
$$\hat{y} = 0 + \beta_1 \times \text{horsepower}$$

# Loss functions and performance metrics



$$\hat{y} = 0 + 0.33 \times \text{horsepower}$$

# Loss functions and performance metrics



$$\hat{y} = 0 + 0.33 \times \text{horsepower}$$

# Loss functions and performance metrics

## Loss functions

- Different loss functions measures different properties of the model fit
- Optimizing for them gives different parameter estimates
- Must be differentiable to allow for mathematical optimization

# Loss functions and performance metrics

$$\text{mse}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

OR

$$\text{mae}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n} |y_i - \hat{y}_i|$$

# Performance metrics: Regression

$$\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

Mean squared error (MSE)

- + Can be used as a loss function
- + Widely used
- + Intuitive
- + Penalizes large errors
- ? Interpretation
- - Depends on scale

# Performance metrics: Regression

$$\sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}$$

Root mean squared error (RMSE)

- + Can be used as a loss function
- + Intuitive
- + Penalizes large errors
- + More interpretable than MSE, total loss ≈ individual loss
- - Depends on scale

# Performance metrics: Regression

$$\frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$$

Mean absolute error (MAE)

- + Can be used as a loss function
- + More interpretable than MSE/RMSE, total loss = average error
- - Feels a bit off
- - Depends on scale

# Performance metrics: Regression

$$\frac{\sum_{i=1}^{n}(y_i-\bar{y})(\hat{y}_i-\bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2\sum_{i=1}^{n}(\hat{y}_i-\bar{\hat{y}})^2}}$$

Pearson correlation coefficient (r)

+ Scale independent

? Captures linear correlation

- Should not be used as a loss function

- Does not care about whether the predictions are close to the true values

# Performance metrics: Regression

$$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

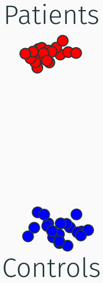
Proportion of variance explained ($r^2$)

+ Scale independent
+ Interpretable

? Captures linear correlation

- Should not be used as a loss function
- Does not care about whether the predictions are close to the true values

# Performance metrics: Binary classification

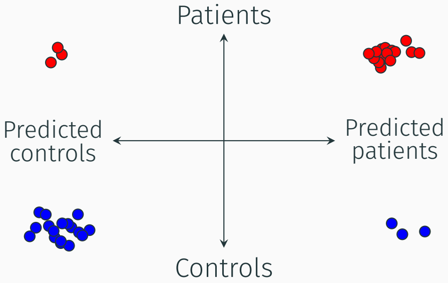


Patients

Controls

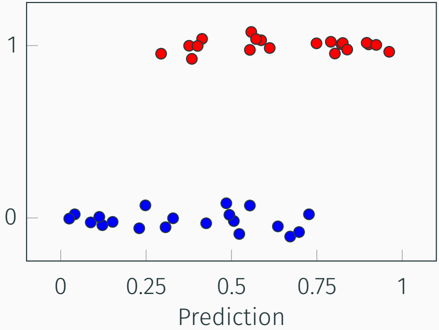
$$y \in \{Patients, Controls\}$$

# Performance metrics: Binary classification



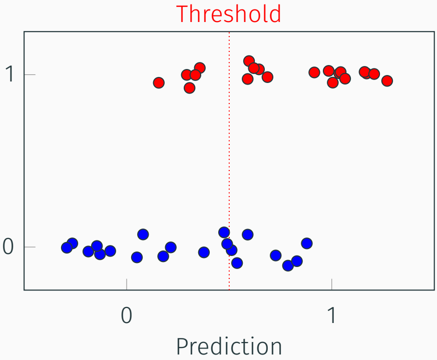
$$y \in \{Patients, Controls\}$$
$$\hat{y} \in \{Patients, Controls\}$$

# Performance metrics: Binary classification



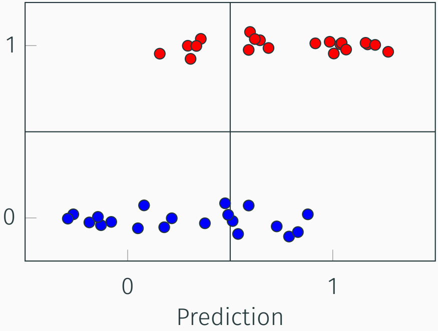
$$y \in \{0, 1\}$$
$$\hat{y} \in [0, 1]$$

# Performance metrics: Binary classification



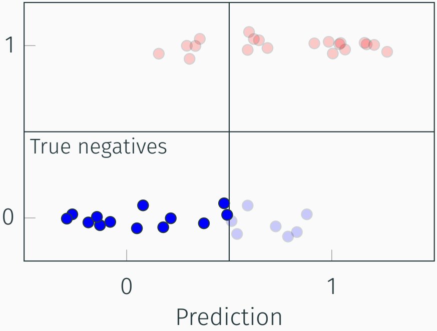
$$y \in \{0,1\}$$
$$\hat{y} \in \{0,1\}$$

# Performance metrics: Binary classification



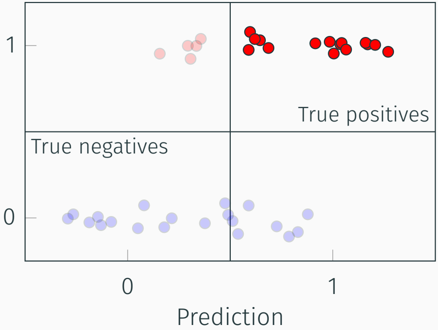
| | |
|---|---|
| | |
| | |

# Performance metrics: Binary classification



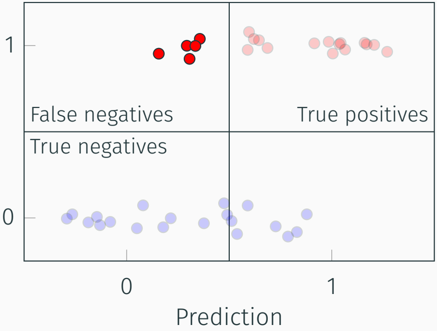
| | |
|---|---|
| TN | |
| | |

# Performance metrics: Binary classification



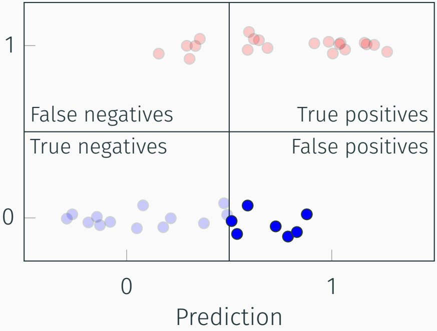
|  |  |
|---|---|
| TN |  |
|  | TP |

# Performance metrics: Binary classification



| | |
|---|---|
| TN | |
| FN | TP |

# Performance metrics: Binary classification



|  |  |
|---|---|
| TN | FP |
| FN | TP |

# Performance metrics: Binary classification

**Confusion matrix:**

| | | Predicted | |
|---|---|---|---|
| | | 0 | 1 |
| True | 0 | TN | FP |
| | 1 | FN | TP |

# Performance metrics: Binary classification

Binary classification metrics:

- Many metrics rely on thresholding the predictions to obtain binary predictions.
- Although not a metric per se, the confusion matrix is a very useful tool to understand model behaviour, and should **always** be looked at (and preferably reported).

# Performance metrics: Binary classification

$$-(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

Logloss

- + Does not rely on thresholding
- + Can be used as a loss function (and very often is)
- - Not very interpretable

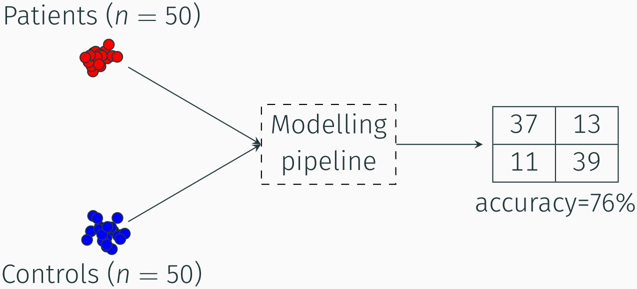
# Performance metrics: Binary classification

$$\frac{TP+TN}{TP+TN+FP+FN}$$

Accuracy

- + Interpretable
- – Does not account for imbalanced classes
- – Does not account for different costs of misclassification

# Performance metrics: Binary classification



Patients ($n = 50$)

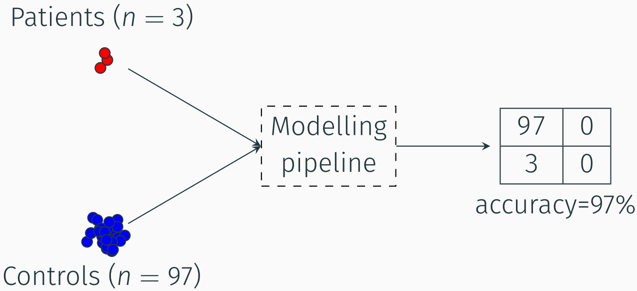
Controls ($n = 50$)

Modelling pipeline

|  |  |
|---|---|
| 37 | 13 |
| 11 | 39 |

accuracy=76%

# Performance metrics: Binary classification



Patients ($n = 3$)

Controls ($n = 97$)

Modelling pipeline

| | |
|---|---|
| 97 | 0 |
| 3 | 0 |

accuracy=97%

# Performance metrics: Binary classification

$$\frac{TP}{TP+FN}$$

True positive rate (sensitivity)

- + Interpretable, calculates the proportion of cases that are detected
- + Useful when the cost of false negatives is high (Population-wide screening for severe disease)

# Performance metrics: Binary classification

$$\frac{TN}{TN+FP}$$

## True negative rate (specificity)

- + Interpretable, calculates the proportion of controls that are detected
- + Useful when the cost of false positives is high (Intrusive treatment of rare and mild conditions)

# Performance metrics: Binary classification

$$\frac{TP}{TP+FP}$$

Positive predictive value (PPV, precision)

- + Interpretable, calculates the proportion of predicted cases that are actually cases
- + Useful when the cost of false positives is high (Selection of participants for expensive clinical trials)

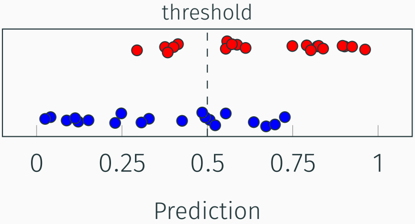
# Performance metrics: Binary classification

$$\frac{\frac{TP}{TP+FN}+\frac{TN}{TN+FP}}{2}$$

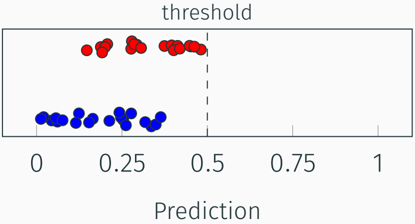
Balanced accuracy

+ Interpretable, behaves similarly to regular accuracy.
+ Takes into account imbalanced classes

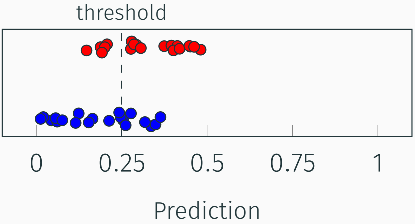
# Classification metrics: Area under the curve



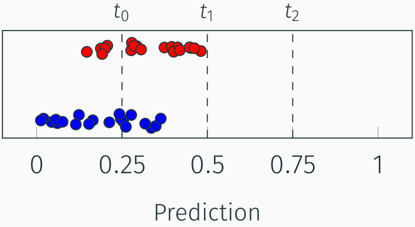
# Classification metrics: Area under the curve



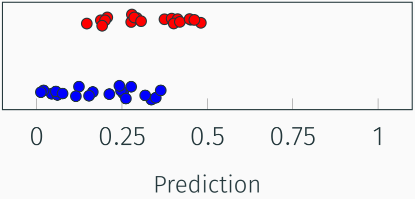
# Classification metrics: Area under the curve



# Classification metrics: Area under the curve

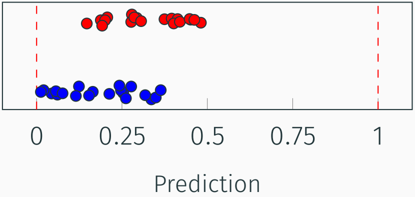


# Classification metrics: Area under the curve



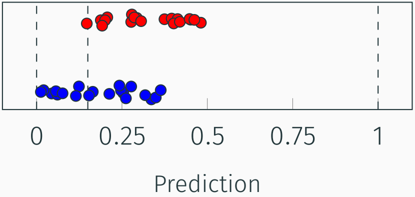
| threshold | TPR | FPR |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

# Classification metrics: Area under the curve



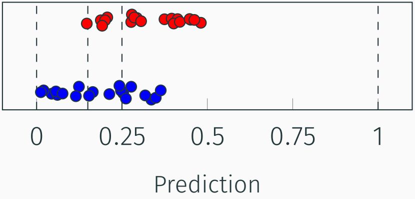
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| | | |
| | | |
| | | |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



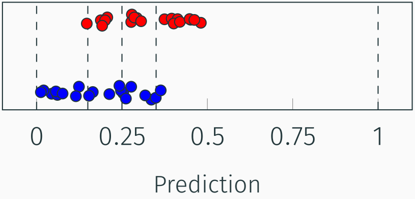
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| | | |
| | | |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



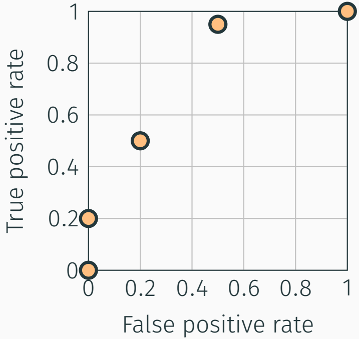
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| | | |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



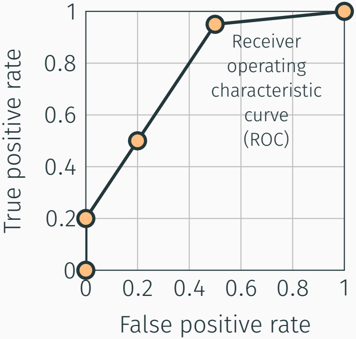
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



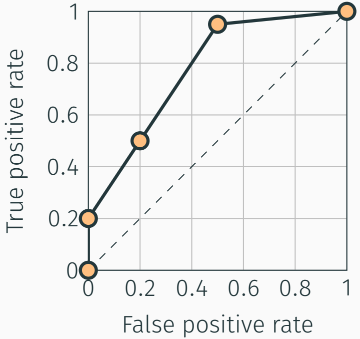
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



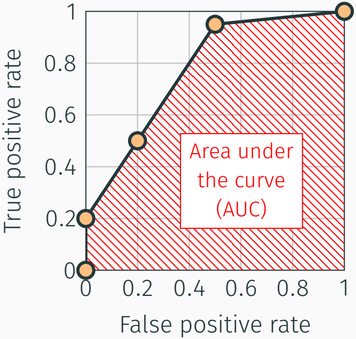
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



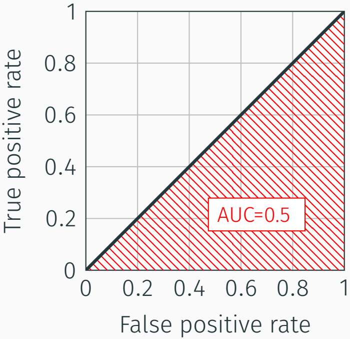
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



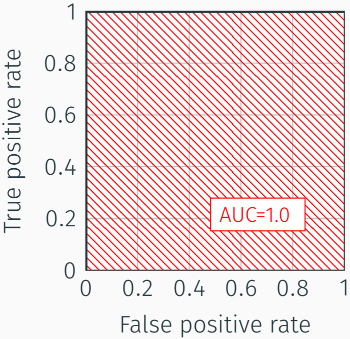
| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve



| threshold | TPR | FPR |
|---|---|---|
| 0 | 1 | 1 |
| 0.15 | 0.95 | 0.5 |
| 0.25 | 0.5 | 0.2 |
| 0.35 | 0.2 | 0.0 |
| 1 | 0 | 0 |

# Classification metrics: Area under the curve

Area under the receiver operating characteristic curve (AUC/AUROC)

- A performance metric that does not rely on a correct classification threshold
- Measures whether the predictions are ranked correctly (e.g. patients have a higher prediction than controls)
- **Handles class imbalance (relatively well)** and is commonly reported in the literature

# Loss functions and performance metrics: Summary

Performance metrics and loss functions measure the performance of a predictive model

- There is a range of alternatives that can be used, each capturing a different aspect of a model’s performance
- It is good practice to report more than one metric
- For regression:
    - MSE is a common loss function with nice mathematical properties.
    - MAE is an intuitive performance metric
- For classification:
    - Log-loss is the most common loss function for probabilistic classifiers
    - AUC is a widely used metric that is easy to interpret, handles class imbalance (to some degree), and is not reliant on the choice of classification threshold

# Loss functions and performance metrics: Summary

http://localhost:8889/notebooks/notebooks%2FClassification%20metrics.ipynb

# Strategies for model evaluation

UNIVERSITETET
I OSLO

# Model evaluation: Rationale

Statistical inference:

Goal: In-sample quantification

Predictive modelling:

Goal: Out-of-sample generalization

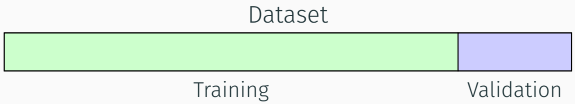
# Model evaluation: Rationale

How can we test how good our model is on **unseen data** and **be certain that performance holds if we present even more new data**

# Model evaluation: Validation set

Dataset

# Model evaluation: Validation set

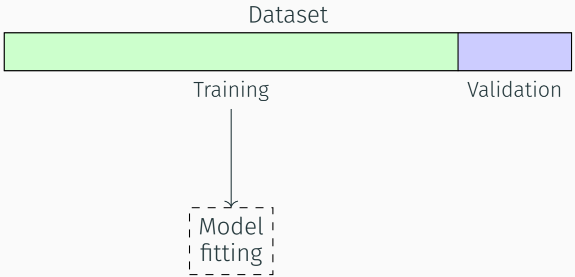


Dataset

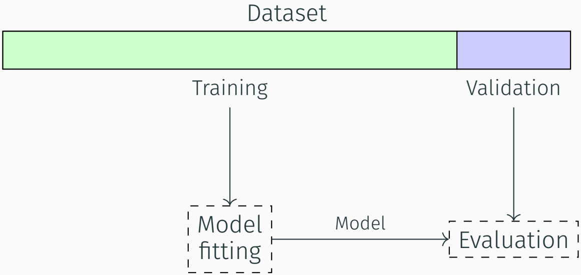
Training

Validation

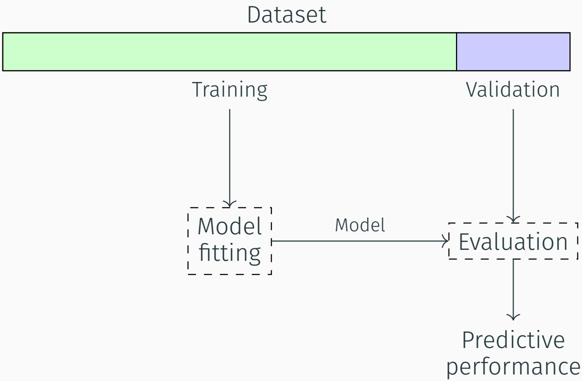
# Model evaluation: Validation set



# Model evaluation: Validation set



# Model evaluation: Validation set

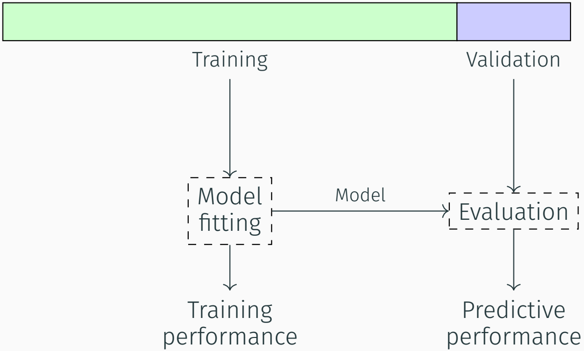


# Model evaluation: Validation set

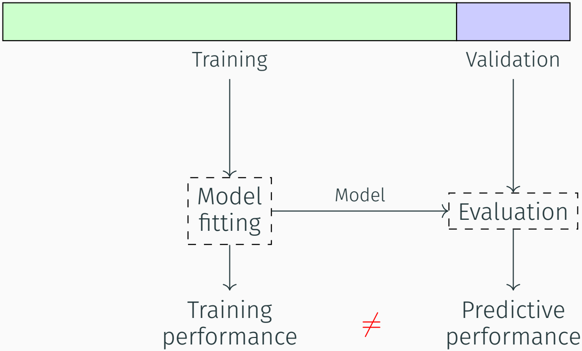
In the validation set approach we split the dataset into two subsets (commonly ∼80%/20%), use the first for training the model and the second to test its performance.

- + Accurate estimate of out-of-sample error
- + Simple
- - Variable results depending on the exact split
- - Only uses a subset of data for training models
- - Gives a point estimate of the error, without confidence intervals

# Model evaluation: Validation set



# Model evaluation: Validation set



Training

Validation

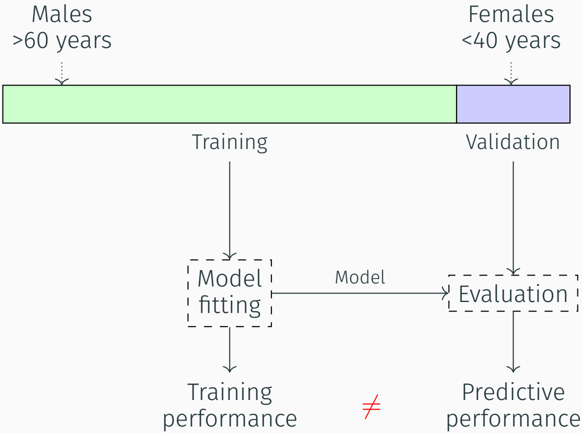
Model fitting

Model

Evaluation

Training performance ≠ Predictive performance

# Model evaluation: Validation set



Males
>60 years

Females
<40 years

Training

Validation

Model
fitting

Model

Evaluation

Training
performance

≠

Predictive
performance

# Model evaluation: Stratification

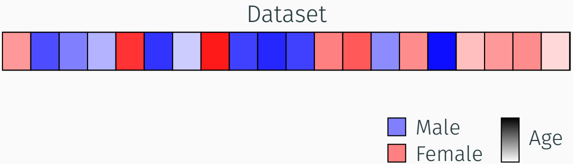
Stratification:

Ensuring all folds of the dataset are similar with respect to some given characteristics.

# Model evaluation: Stratification

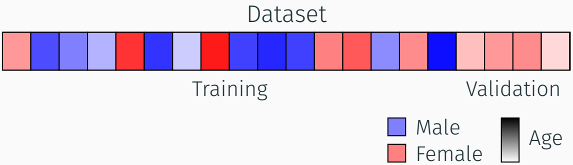
```
In[1]: df = ...
```

# Model evaluation: Stratification



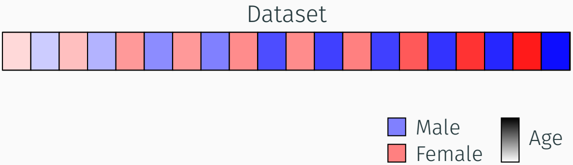
```
In[1]:  df = ...
```

# Model evaluation: Stratification



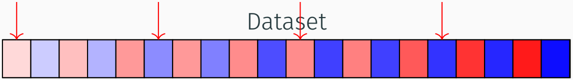
In[1]:
```
df = ...

train = df.iloc[:int(len(df) * 0.8)]
validation = df.iloc[int(len(df) * 0.8):]
```

# Model evaluation: Stratification



```
In[1]:  df = ...
        df = df.sort_values(['sex', 'age'])
```

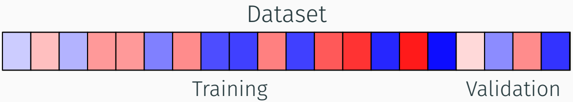
# Model evaluation: Stratification



In[1]:

```
df = ...
df = df.sort_values(['sex', 'age'])

df['fold'] = np.arange(len(df)) % (1 / 0.2)
train = df[df['fold'] != 0]
val = df[df['fold'] == 0]
```

# Model evaluation: Stratification



Dataset

Training

Validation

In[1]:

```
df = ...
df = df.sort_values(['sex', 'age'])

df['fold'] = np.arange(len(df)) % (1 / 0.2)
train = df[df['fold'] != 0]
val = df[df['fold'] == 0]
```

# Model evaluation: Stratification

Stratification:

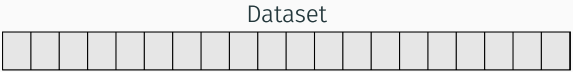
Ensuring all folds of the dataset are similar with respect to some given characteristics.

- Helps alleviate the risk of training performance ≫ validation performance
- **Always** stratify on target variable first
- Also good idea to stratify on other core characteristics, e.g. sex and age

In[1]:
```
from sklearn.model_selection import  train_test_split
```

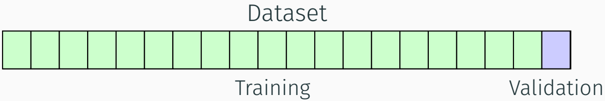
```
library(splitstackshape)
stratified(data, columns, split)
```

# Model evaluation: Leave-one-out cross-validation

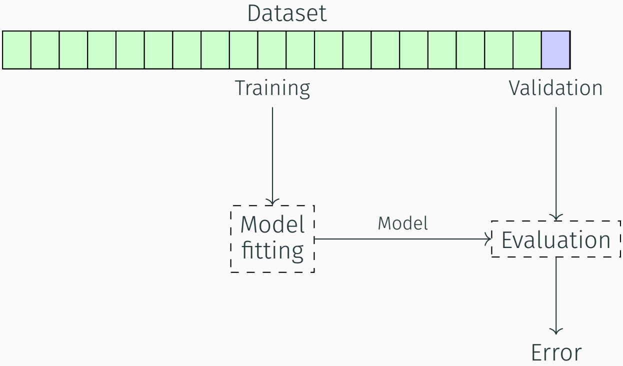


Dataset

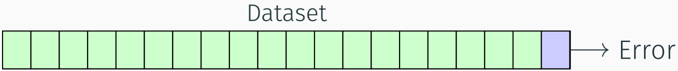
# Model evaluation: Leave-one-out cross-validation



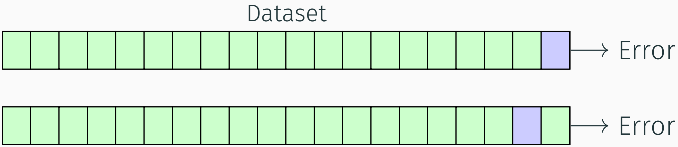
# Model evaluation: Leave-one-out cross-validation



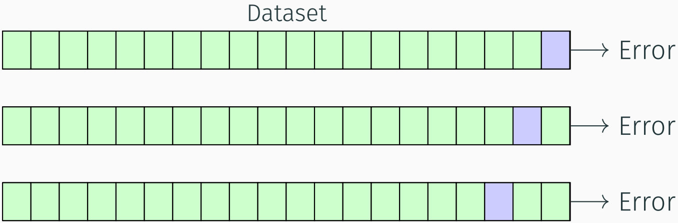
# Model evaluation: Leave-one-out cross-validation



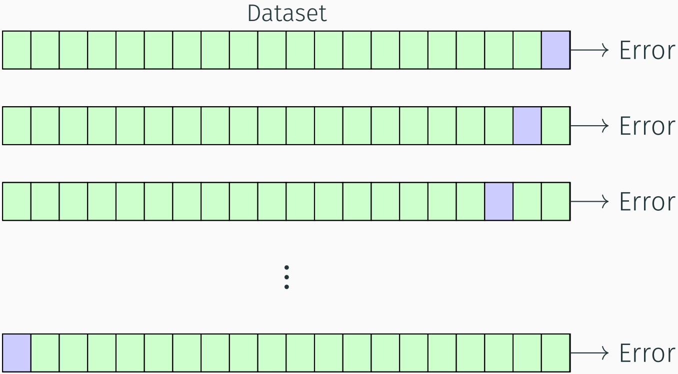
# Model evaluation: Leave-one-out cross-validation



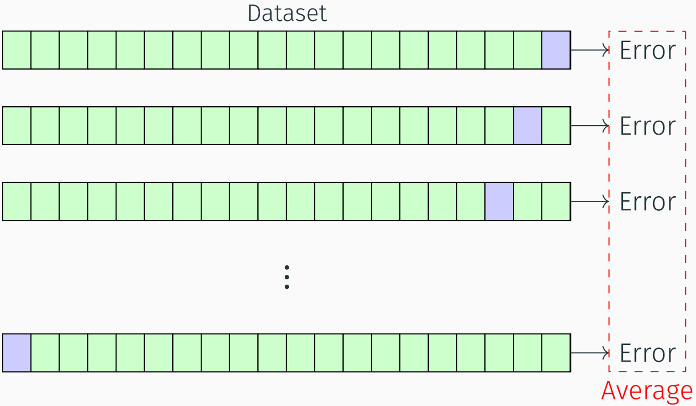
# Model evaluation: Leave-one-out cross-validation



# Model evaluation: Leave-one-out cross-validation



# Model evaluation: Leave-one-out cross-validation



# Model evaluation: Leave-one-out cross-validation

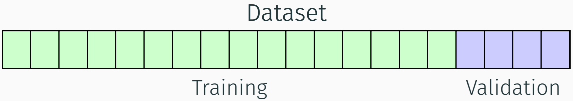
Fits $n$ models for $n$ datapoints, each time leaving a single datapoint out for testing.

+ Uses all data to train models
+ Not dependent on arbitrary data splits
+ Unbiased (with regards to the full dataset)

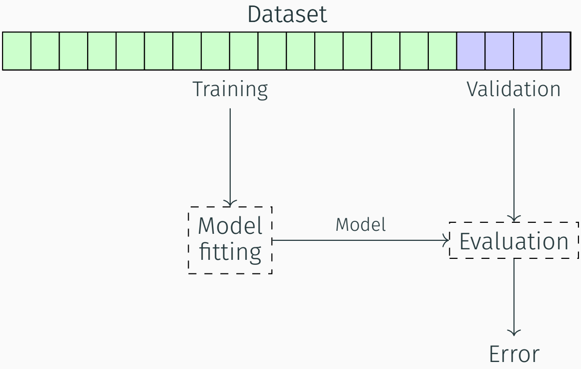
- Computationally expensive
- Effectively gives a point estimate of the error
- All models are going to be trained on $> 99\%$ overlapping data $\rightarrow$ highly correlated

# Model evaluation: Cross-validation

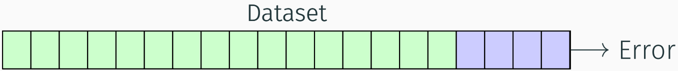
# Model evaluation: Cross-validation



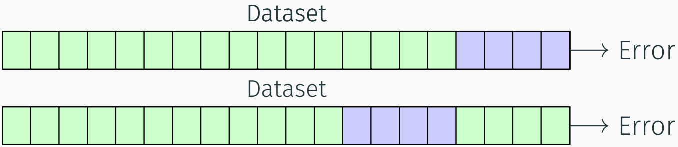
# Model evaluation: Cross-validation



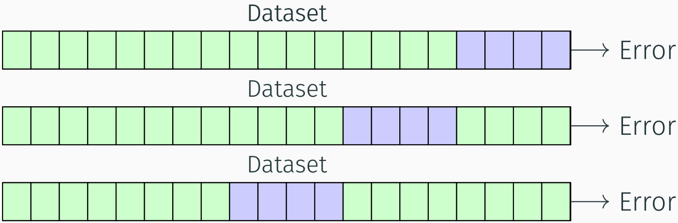
# Model evaluation: Cross-validation



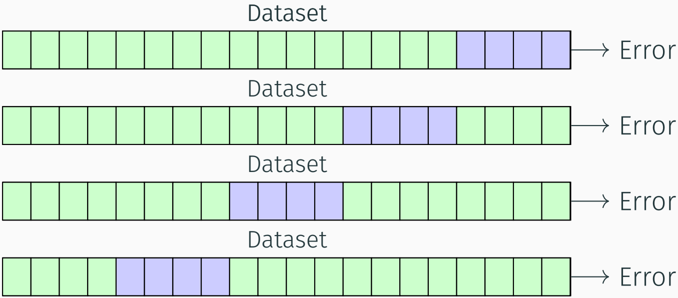
# Model evaluation: Cross-validation



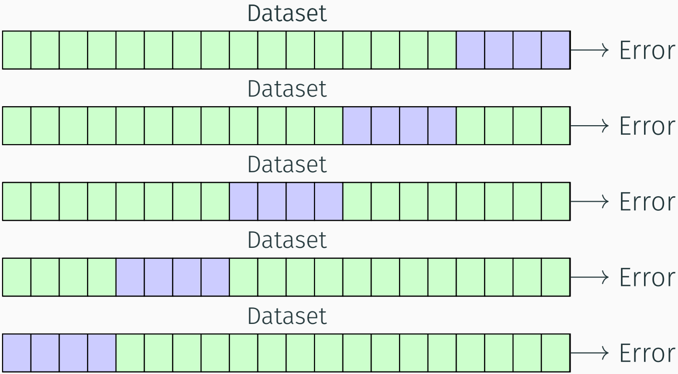
# Model evaluation: Cross-validation



# Model evaluation: Cross-validation



# Model evaluation: Cross-validation



Dataset → Error

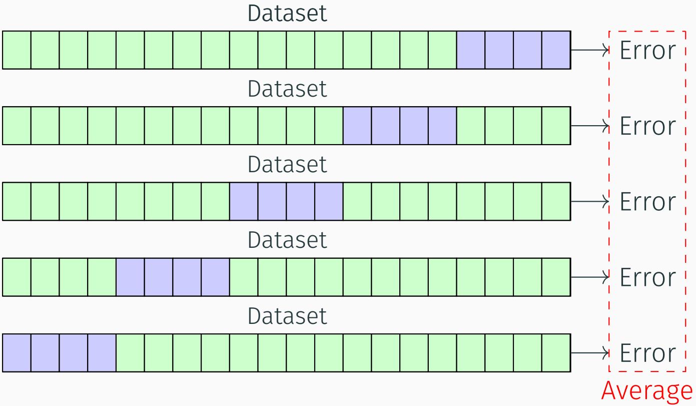
Dataset → Error

Dataset → Error

Dataset → Error

Dataset → Error

# Model evaluation: Cross-validation



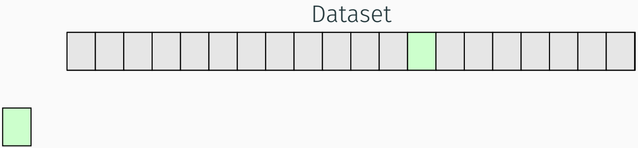
# Model evaluation: Cross-validation

Fits $k$ (usually $k \in \{5, 10\}$) models for $n > k$ datapoints, each leaving $n/k$ datapoints for out-of-sample testing.

+ Uses all data to train models
+ Yields multiple estimates of out-of-sample error
- Different choices of $k$ (and exact splits) yields different results
- No longer a single model from which information (e.g. parameter estimates and p-values) can be derived

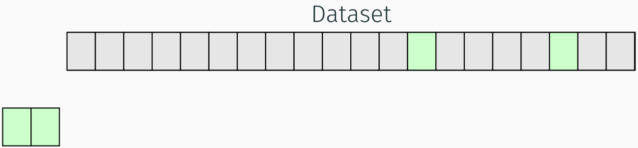
# Model evaluation: Bootstrapping

# Model evaluation: Bootstrapping

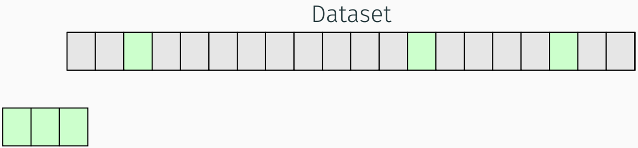


Dataset

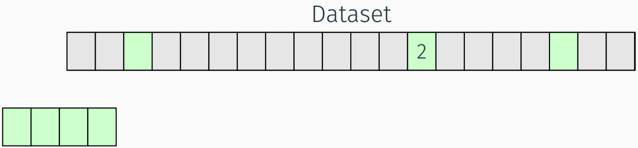
# Model evaluation: Bootstrapping



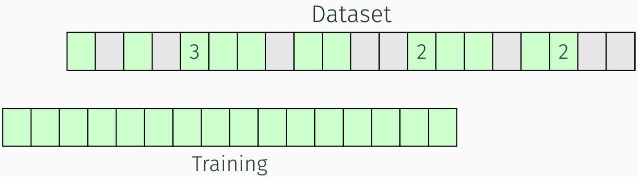
# Model evaluation: Bootstrapping



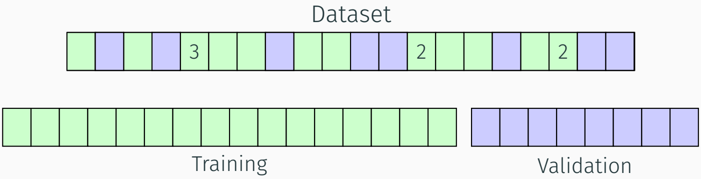
# Model evaluation: Bootstrapping



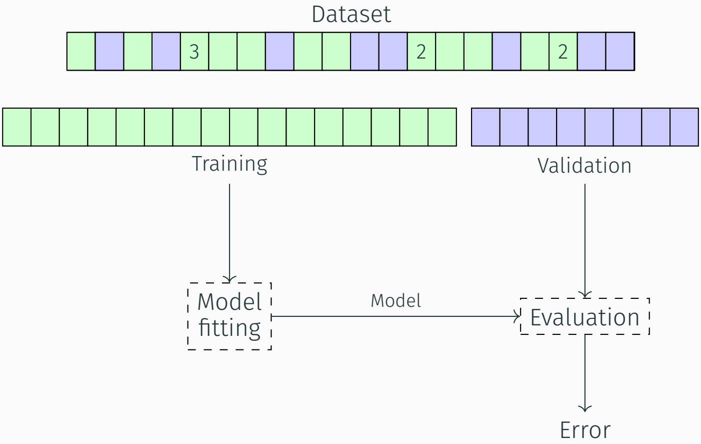
# Model evaluation: Bootstrapping



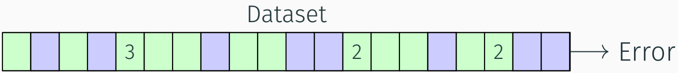
# Model evaluation: Bootstrapping



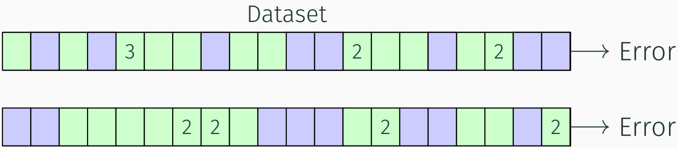
# Model evaluation: Bootstrapping



# Model evaluation: Bootstrapping



# Model evaluation: Bootstrapping

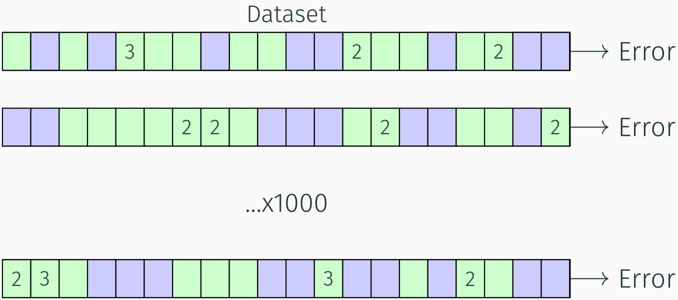


Dataset

3 2 2 → Error

2 2 2 2 → Error

# Model evaluation: Bootstrapping



Dataset

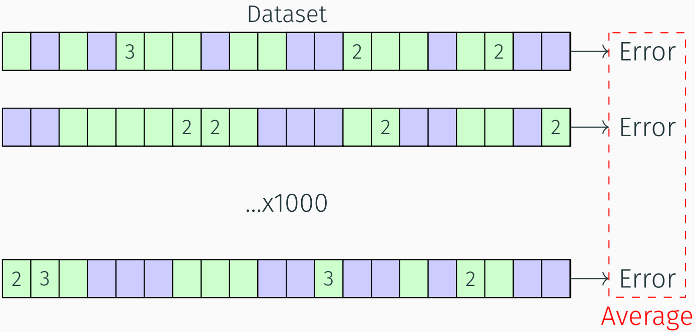
→ Error

→ Error

...x1000

→ Error

# Model evaluation: Bootstrapping

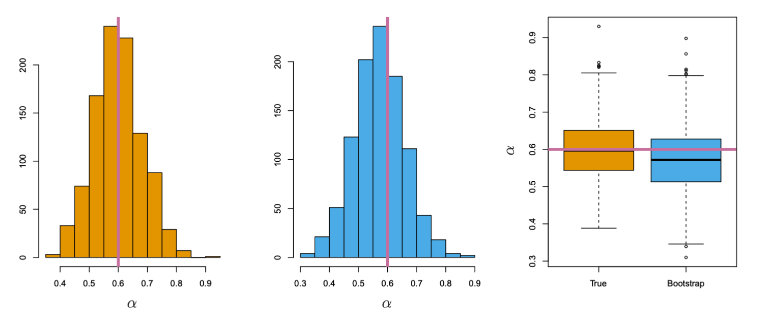


# Model evaluation: Bootstrapping

Fits $b$ models with $m$ datapoints (typically $m < n$), sampled from the original dataset **with replacement**.

- + Uses all data to train models
- + Provides a dense distribution of model performances
- + **Versatile: Can be used for other things, e.g. getting a confidence interval for model parameters**
- - Different choices of $b$ (and exact splits) yields different results

# Model evaluation: Bootstrapping

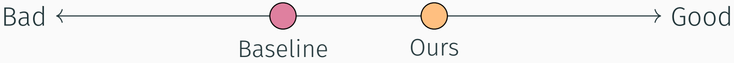


# Model evaluation: Comparison

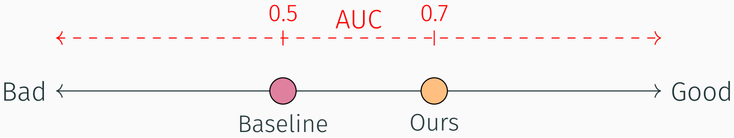
Why do we want to evaluate our model?

1. We want to show that our model is better than random guessing
2. We want to show that our model is better than another model

# Model evaluation: Comparison



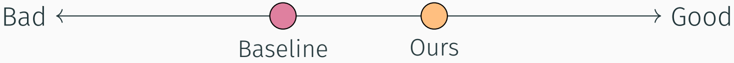
# Model evaluation: Comparison



# Model evaluation: Comparison

http://localhost:8890/notebooks/notebooks%2FModel%20variability.ipynb

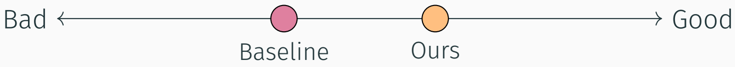
# Model evaluation: Comparison



There is going to be variability in our model's performance (and possibly the baseline).

**Is our model significantly better?**

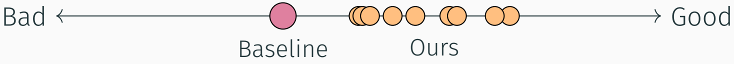
# Model evaluation: Comparison



Approach 1:

Is the mean of the distribution of performances from our model (with regards to variability that is **unrelated** to efficacy) significantly higher than the point-estimate baseline?

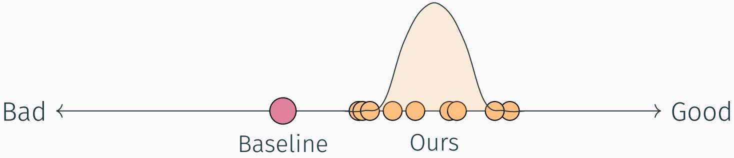
# Model evaluation: Comparison



Approach 1:

Is the mean of the distribution of performances from our model (with regards to variability that is **unrelated** to efficacy) significantly higher than the point-estimate baseline?

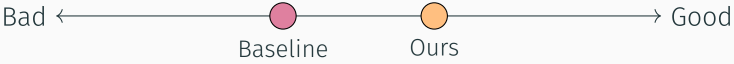
# Model evaluation: Comparison



Approach 1:

Is the mean of the distribution of performances from our model (with regards to variability that is **unrelated** to efficacy) significantly higher than the point-estimate baseline?

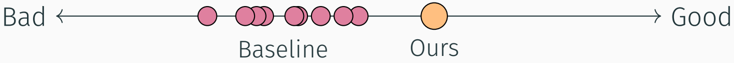
# Model evaluation: Comparison



Approach 2:

Is the point-estimate performance of our model significantly higher than the mean of the baseline distribution?

# Model evaluation: Comparison

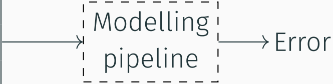


Approach 2:

Is the point-estimate performance of our model significantly higher than the mean of the baseline distribution?

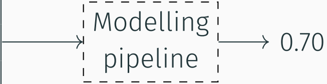
# Model evaluation: Comparison

| Age | Sex | Feature | Outcome |
|---|---|---|---|
| 25 | Male | 0.53 | 1 |
| 38 | Female | -0.76 | 1 |
| 45 | Male | 0.89 | 1 |
| 33 | Female | -0.21 | 1 |
| 29 | Male | 0.12 | 1 |
| 41 | Female | -0.68 | 0 |
| 56 | Male | 0.45 | 0 |
| 52 | Female | -0.32 | 0 |
| 31 | Male | 0.91 | 0 |
| 48 | Female | -0.15 | 0 |



# Model evaluation: Comparison

| Age | Sex | Feature | Outcome |
|---|---|---|---|
| 25 | Male | 0.53 | 1 |
| 38 | Female | -0.76 | 1 |
| 45 | Male | 0.89 | 1 |
| 33 | Female | -0.21 | 1 |
| 29 | Male | 0.12 | 1 |
| 41 | Female | -0.68 | 0 |
| 56 | Male | 0.45 | 0 |
| 52 | Female | -0.32 | 0 |
| 31 | Male | 0.91 | 0 |
| 48 | Female | -0.15 | 0 |

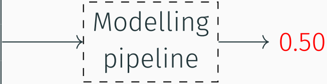


Modelling pipeline

0.70

# Model evaluation: Comparison

| Age | Sex | Feature | Outcome |
|---|---|---|---|
| 25 | Male | 0.53 | 1 |
| 38 | Female | -0.76 | 0 |
| 45 | Male | 0.89 | 1 |
| 33 | Female | -0.21 | 0 |
| 29 | Male | 0.12 | 1 |
| 41 | Female | -0.68 | 0 |
| 56 | Male | 0.45 | 1 |
| 52 | Female | -0.32 | 0 |
| 31 | Male | 0.91 | 1 |
| 48 | Female | -0.15 | 0 |

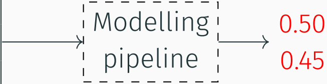


Modelling pipeline

0.50

# Model evaluation: Comparison

| Age | Sex | Feature | Outcome |
|---|---|---|---|
| 25 | Male | 0.53 | 0 |
| 38 | Female | -0.76 | 0 |
| 45 | Male | 0.89 | 0 |
| 33 | Female | -0.21 | 0 |
| 29 | Male | 0.12 | 0 |
| 41 | Female | -0.68 | 1 |
| 56 | Male | 0.45 | 1 |
| 52 | Female | -0.32 | 1 |
| 31 | Male | 0.91 | 1 |
| 48 | Female | -0.15 | 1 |



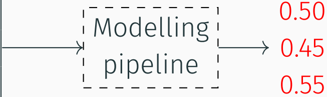
Modelling pipeline

0.50

0.45

# Model evaluation: Comparison

| Age | Sex | Feature | Outcome |
|---|---|---|---|
| 25 | Male | 0.53 | 0 |
| 38 | Female | -0.76 | 1 |
| 45 | Male | 0.89 | 0 |
| 33 | Female | -0.21 | 0 |
| 29 | Male | 0.12 | 1 |
| 41 | Female | -0.68 | 1 |
| 56 | Male | 0.45 | 0 |
| 52 | Female | -0.32 | 0 |
| 31 | Male | 0.91 | 1 |
| 48 | Female | -0.15 | 1 |



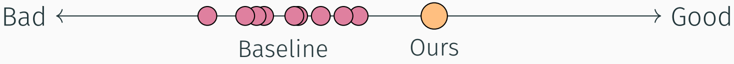
Modelling pipeline

0.50

0.45

0.55

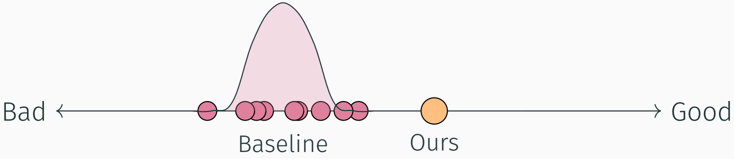
# Model evaluation: Comparison



Approach 2:

Is the point-estimate performance of our model significantly higher than the mean of the baseline distribution?

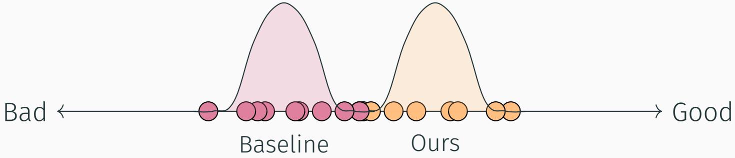
# Model evaluation: Comparison



Approach 2:

Is the point-estimate performance of our model significantly higher than the mean of the baseline distribution?

# Model evaluation: Comparison



Approach 3:

Is the mean of the distribution of performances from our model significantly higher than the mean of the distribution of baseline performances?

# Model evaluation: Comparison

| Fold | Ours | Baseline |
|---|---|---|
| 1 | 0.75 | 0.71 |
| 2 | 0.62 | 0.55 |
| 3 | 0.58 | 0.57 |
| 4 | 0.87 | 0.81 |
| 5 | 0.65 | 0.63 |
| 6 | 0.98 | 0.97 |
| 7 | 0.55 | 0.52 |
| 8 | 0.69 | 0.52 |
| 9 | 0.91 | 0.85 |
| 10 | 0.88 | 0.81 |

The small gain of our model will disappear in the noise between the folds using a non-paired statistical test. Use a paired test, e.g. Wilcoxon signed-rank test

# Model evaluation: Summary

- Model evaluation should **always** happen out-of-sample
- If n is big ($\geq 10000$), a single train/validation split is often sufficient
- For smaller samples, k-fold cross-validation with $5 \leq k \leq 10$ is a good trade-off between bias and variance
- The bootstrap is an effective way of getting confidence intervals for model performance **and parameters**
- Cross-validation (or bootstrapping) will produce a distribution of model performances (although caution the correlation)
- Permutation testing will produce a distribution of baseline performances
- Compare models across folds using Wilcoxon signed-rank test **(ensure the folds are the same!)**

# Model selection and evaluation

UNIVERSITETET
I OSLO

# Model selection and evaluation: Rationale

- Model evaluation via cross-validation is sufficient if we want to estimate the out-of-sample error of a **known model**.
- Very often we want to know whether a set of predictors are informative for an outcome **given the best possible model**.
- In that case, we have to both **choose the best model**, and **estimate its performance**.
- If we choose the model based on regular cross-validation, the performance estimate will (likely) be inflated

# Model selection and evaluation: Rationale

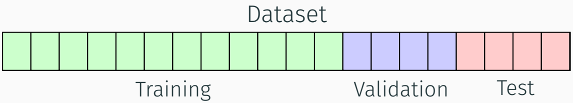
- Model evaluation via cross-validation is sufficient if we want to estimate the out-of-sample error of a **known model**.
- Very often we want to know whether a set of predictors are informative for an outcome **given the best possible model**.
- In that case, we have to both **choose the best model**, and **estimate its performance**.
- If we choose the model based on regular cross-validation, the performance estimate will (likely) be inflated

→ We need a more advanced strategy

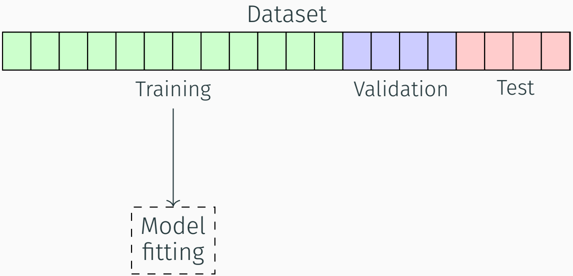
# Model selection and evaluation: Train/validation/test

Dataset

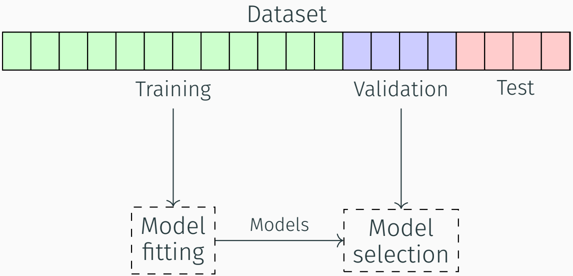
# Model selection and evaluation: Train/validation/test



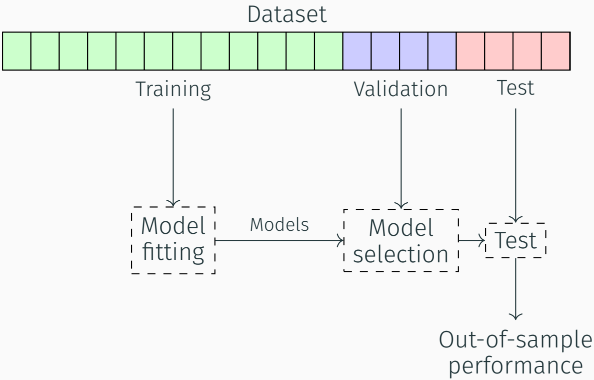
# Model selection and evaluation: Train/validation/test



# Model selection and evaluation: Train/validation/test



# Model selection and evaluation: Train/validation/test



Dataset

Training

Validation

Test

Model fitting

Models

Model selection

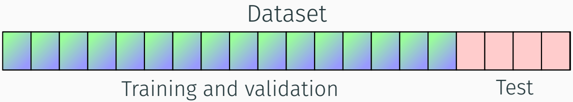
Test

Out-of-sample performance

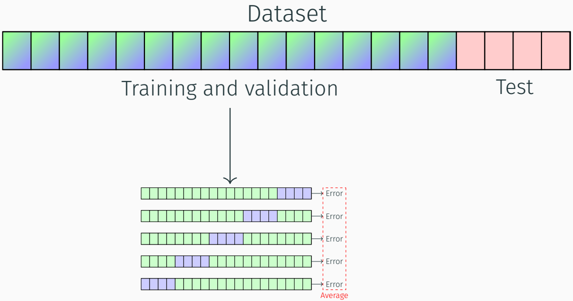
# Model selection and evaluation: Nested cross-validation

Dataset

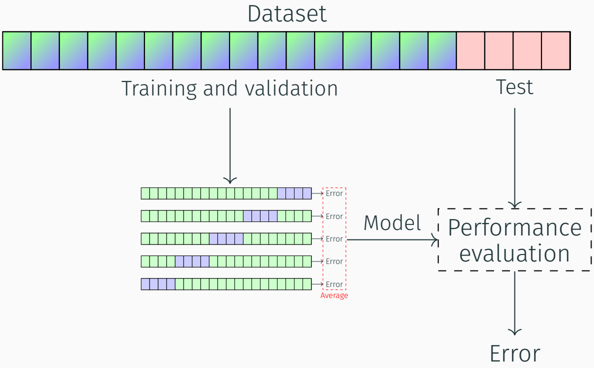
# Model selection and evaluation: Nested cross-validation



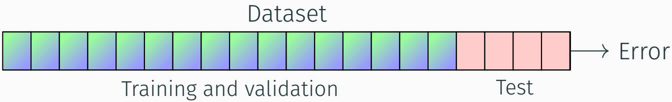
# Model selection and evaluation: Nested cross-validation



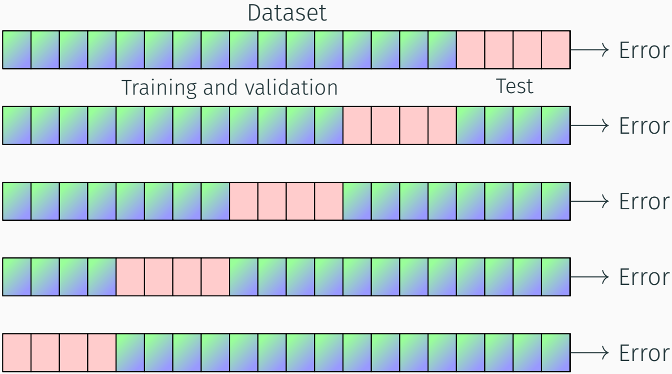
# Model selection and evaluation: Nested cross-validation



# Model selection and evaluation: Nested cross-validation



# Model selection and evaluation: Nested cross-validation



Dataset

Training and validation

Test

→ Error

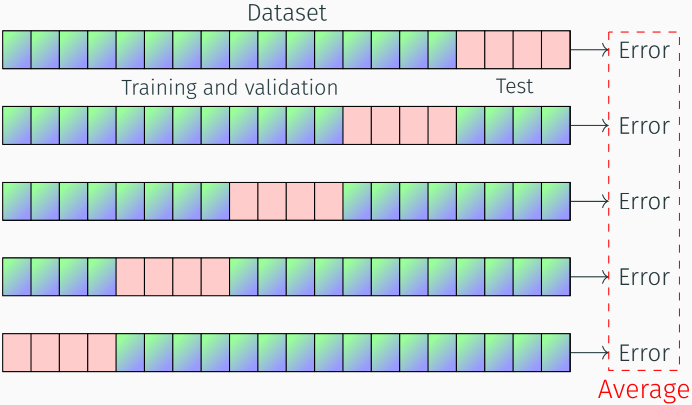
→ Error

→ Error

→ Error

→ Error

# Model selection and evaluation: Nested cross-validation



# Model selection and evaluation: Nested cross-validation

Performs $k$ outer cross-validations, each performing $k/k-1$ inner cross validations, and uses the best models from the inner loop to predict in the outer loop.

+ Uses all data to train models
+ **Unbiased estimate of out-of-sample error**
- **Very** computationally expensive
- We now have either $k$, or $k^2$ models that might behave in different ways

# Model selection and evaluation: Summary

- Whenever a modelling choice is made on the basis of performance in a dataset, **we have to assume the performance achieved by the chosen model is inflated**
- If n is big ($\geq 10000$), a single train/validation/test split is often sufficient
- When possible, use nested cross-validation to select the best model and estimate the out-of-sample error

# Model selection and evaluation: Summary

!!!!!

- Whenever a modelling choice is made on the basis of performance in a dataset, **we have to assume the performance achieved by the chosen model is inflated**
- If n is big ($\geq 10000$), a single train/validation/test split is often sufficient
- When possible, use nested cross-validation to select the best model and estimate the out-of-sample error