

PSY9511: Seminar 4

Testing, resampling, and splitting

Esten H. Leonardsen

26.10.23



**UNIVERSITETET
I OSLO**

1. Coding tips
 - Loops
 - Functions
2. Performance metrics
3. Strategies for model assessment
 - Training and validation split
 - (Stratification)
 - (Leave-one-out cross-validation)
 - Cross-validation
 - Bootstrap
4. Strategies for model selection **and** assessment
 - Train/validation/test split
 - Nested cross-validation

Coding tips



UNIVERSITETET
I OSLO

Coding tips

```
In[1]: import numpy as np
import pandas as pd

df = pd.read_csv('Auto.csv')
df = df.replace('?', np.nan)
train = df.iloc[:200].copy()
test = df.iloc[200:].copy()

test['cylinders'] = (test['cylinders'] - train['cylinders'].mean()) / train['cylinders'].std()
train['cylinders'] = (train['cylinders'] - train['cylinders'].mean()) / train['cylinders'].std()
test['weight'] = (test['weight'] - train['weight'].mean()) / train['weight'].std()
train['weight'] = (train['weight'] - train['weight'].mean()) / train['weight'].std()
test['year'] = (test['year'] - train['year'].mean()) / train['year'].std()
train['year'] = (train['year'] - train['year'].mean()) / train['year'].std()
```



[Live coding](#)

Coding tips: Python

```
In[1]: import numpy as np
import pandas as pd

df = pd.read_csv('Auto.csv')
df = df.replace('?', np.nan)
train = df.iloc[:200].copy()
test = df.iloc[200:].copy()

test['cylinders'] = (test['cylinders'] - train['cylinders'].mean()) / train['cylinders'].std()
train['cylinders'] = (train['cylinders'] - train['cylinders'].mean()) / train['cylinders'].std()
test['weight'] = (test['weight'] - train['weight'].mean()) / train['weight'].std()
train['weight'] = (train['weight'] - train['weight'].mean()) / train['weight'].std()
test['year'] = (test['year'] - train['year'].mean()) / train['year'].std()
train['year'] = (train['year'] - train['year'].mean()) / train['year'].std()
```

```
In[2]: import numpy as np
import pandas as pd

df = pd.read_csv('Auto.csv')
df = df.replace('?', np.nan)
train = df.iloc[:200].copy()
test = df.iloc[200:].copy()

def standardize(train: pd.DataFrame, test: pd.DataFrame, column: str):
    train = train.copy()
    test = test.copy()

    test[column] = (test[column] - train[column].mean()) / train[column].std()

    return train, test

for column in ['cylinders', 'displacement', 'weight']:
    train, test = standardize(train, test, column=column)
```



Coding tips: R

```
data <- read.csv('Auto.csv')
data[] <- lapply(data, function(x) replace(x, x == '?', NA))

train <- data[1:200,]
test <- data[200:nrow(data),]

test$cylinders <- (test$cylinders - mean(train$cylinders)) / sd(train$cylinders)
train$cylinders <- (train$cylinders - mean(train$cylinders)) / sd(train$cylinders)
test$weight <- (test$weight - mean(train$weight)) / sd(train$weight)
train$weight <- (train$weight - mean(train$weight)) / sd(train$weight)
test$year <- (test$year - mean(train$year)) / sd(train$year)
train$year <- (train$year - mean(train$year)) / sd(train$year)
```

```
data <- read.csv('Auto.csv')
data[] <- lapply(data, function(x) replace(x, x == '?', NA))

train <- data[1:200,]
test <- data[200:nrow(data),]

test$cylinders <- (test$cylinders - mean(train$cylinders)) / sd(train$cylinders)
train$cylinders <- (train$cylinders - mean(train$cylinders)) / sd(train$cylinders)
test$weight <- (test$weight - mean(train$weight)) / sd(train$weight)
train$weight <- (train$weight - mean(train$weight)) / sd(train$weight)
test$year <- (test$year - mean(train$year)) / sd(train$year)
train$year <- (train$year - mean(train$year)) / sd(train$year)
```



Coding tips: Minimal, complete scripts

Ctrl+Shift+Enter



Performance metrics



UNIVERSITETET
I OSLO

Performance metrics: Regression

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

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

Mean squared error (MSE)

- + Widely used
- + Intuitive
- + Penalizes large errors
- ? Interpretation
- Depends on scale

Performance metrics: Regression

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

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

Root mean squared error (RMSE)

- + Intuitive
- + Penalizes large errors
- + More interpretable than MSE,
total loss \approx individual loss
- Depends on scale

Performance metrics: Regression

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

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

Mean absolute error (MAE)

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

Performance metrics: Regression

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

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

Pearson correlation coefficient (r)

- + Scale independent
- Captures linear correlation
- Does not care about whether the predictions are close to the true values

Performance metrics: Regression

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

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

Proportion of variance explained (r^2)

- + Scale independent
- + Interpretable
- Captures linear correlation
- Does not care about whether the predictions are close to the true values

Performance metrics: Binary classification

Cases

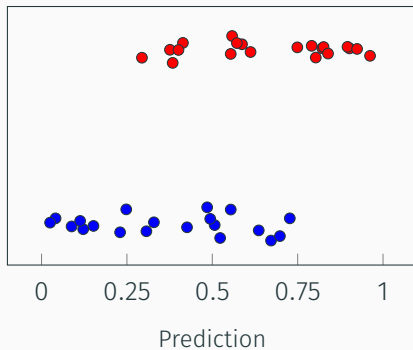


Controls

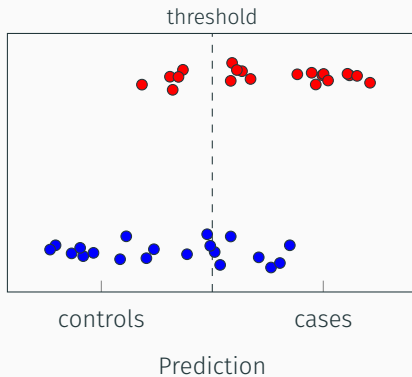
Performance metrics: Binary classification



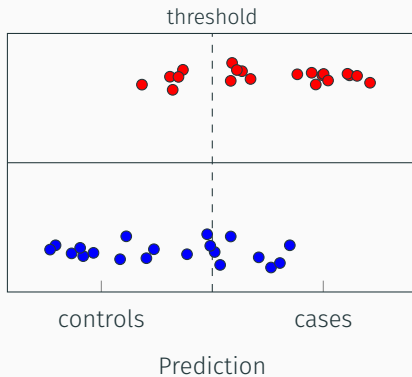
Performance metrics: Binary classification



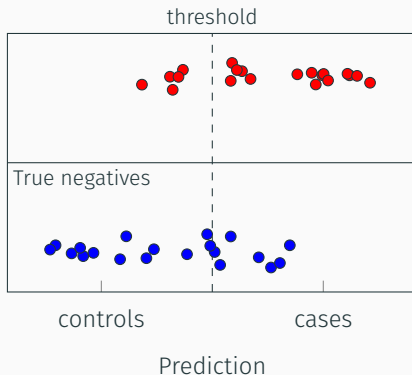
Performance metrics: Binary classification



Performance metrics: Binary classification

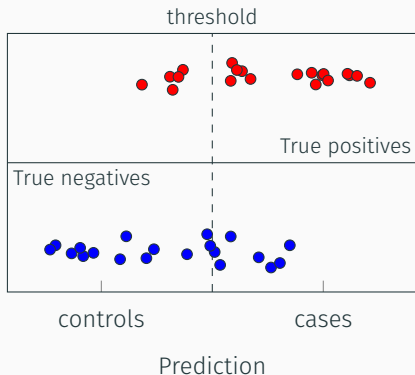


Performance metrics: Binary classification



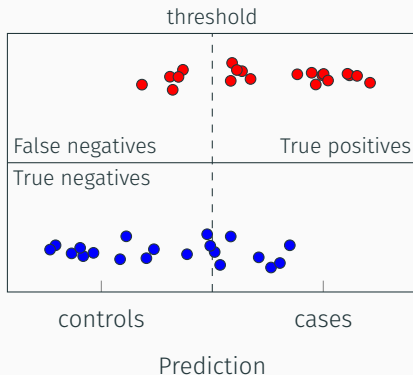
	TN

Performance metrics: Binary classification



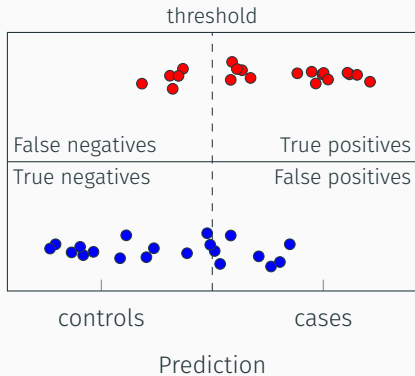
TP	
	TN

Performance metrics: Binary classification



TP	FN
	TN

Performance metrics: Binary classification



TP	FN
FP	TN

Performance metrics: Binary classification

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

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

Accuracy

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

Performance metrics: Binary classification

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

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

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

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

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

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 benign condition)

Performance metrics: Binary classification

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

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

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

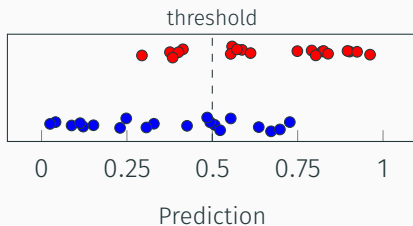
$$\frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}$$

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

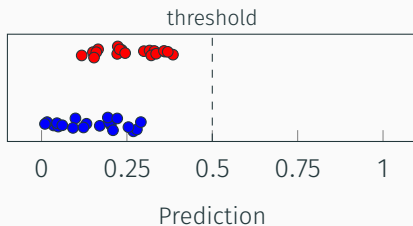
Balanced accuracy

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

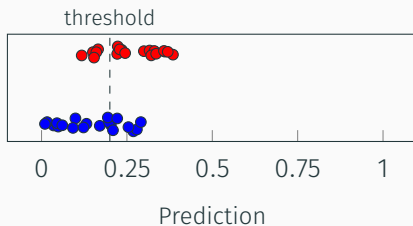
Performance metrics: Binary classification



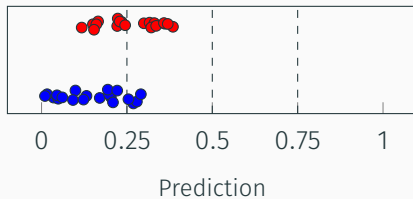
Performance metrics: Binary classification



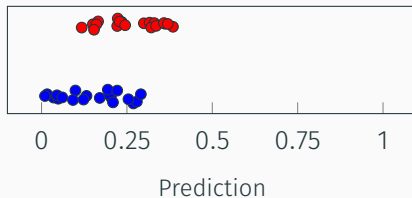
Performance metrics: Binary classification



Performance metrics: Binary classification

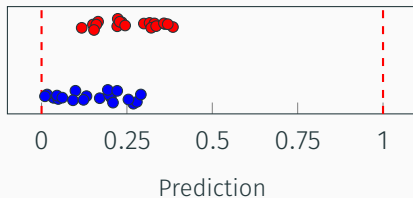


Performance metrics: Binary classification



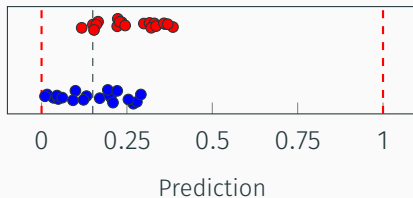
threshold	TPR	FPR

Performance metrics: Binary classification



threshold	TPR	FPR
0	0	0
1	1	1

Performance metrics: Binary classification



threshold	TPR	FPR
0	0	0
1	1	1

Strategies for model assessment



UNIVERSITETET
I OSLO

Statistical inference:

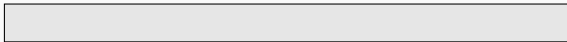
asdf

Predictive modelling:

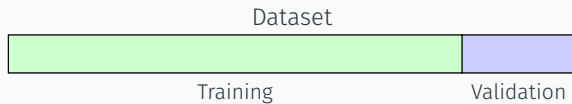
asdf

Model assessment: Validation set

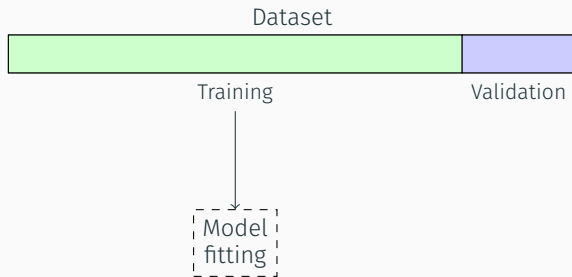
Dataset



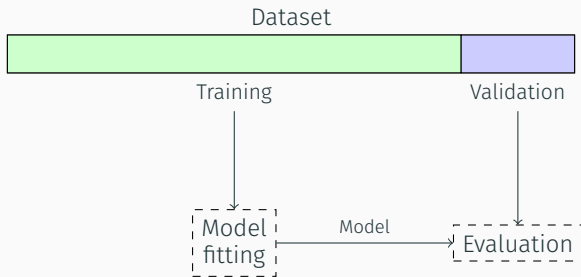
Model assessment: Validation set



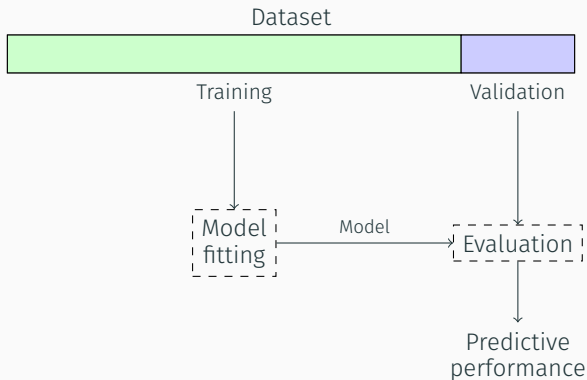
Model assessment: Validation set



Model assessment: Validation set



Model assessment: Validation set



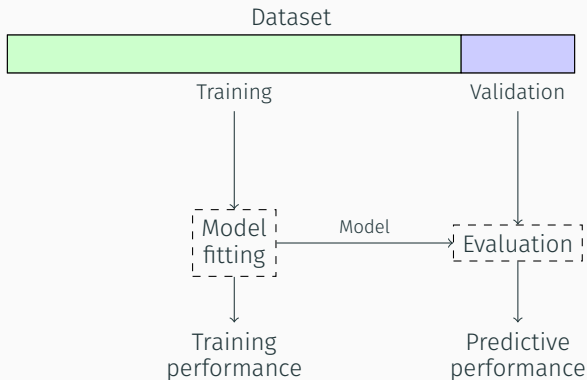
Model assessment: Validation set

In the validation set approach we split the dataset into two subsets, and use for training the model and the other for testing performance.

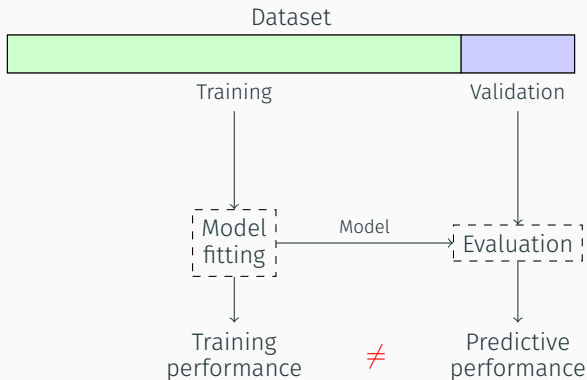
- + Accurate estimate of out-of-sample error
- + Simple
- Highly variable, depends on the exact split
- Only uses a subset of data for training models
- Gives a point estimate of the error, without confidence intervals



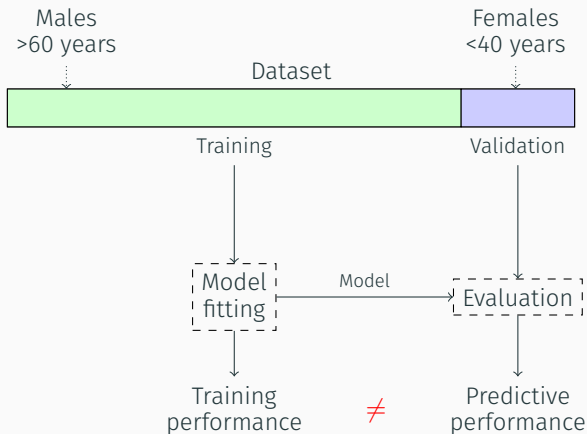
Model assessment: Validation set



Model assessment: Validation set



Model assessment: Validation set



Stratification:

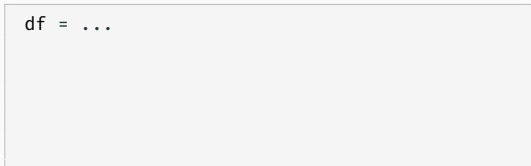
Ensuring all folds of the dataset are similar in terms of some given characteristics.

Model assessment: Stratification

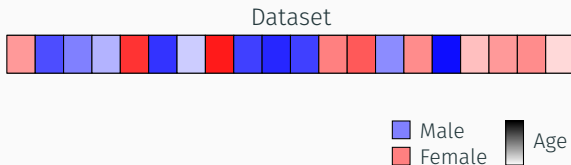
Dataset



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

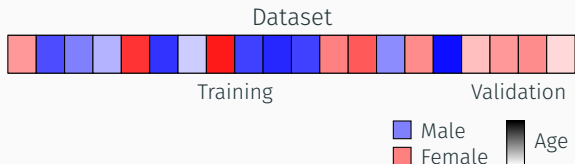


Model assessment: Stratification



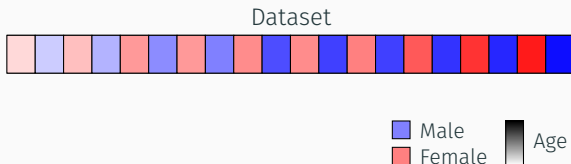
```
In[1]: df = ...
```

Model assessment: Stratification



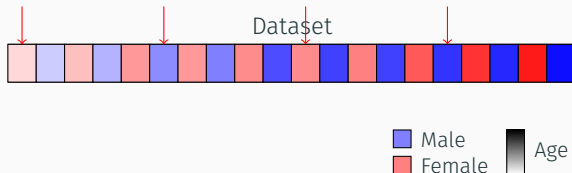
```
In[1]: df = ...  
  
train = df.iloc[:int(len(df) * 0.8)]  
validation = df.iloc[int(len(df) * 0.8):]
```

Model assessment: Stratification



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

Model assessment: 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 assessment: Stratification

Stratification:

Ensuring all folds of the dataset are similar in terms of some given characteristics.

- Helps alleviate the risk of training performance \gg 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 assessment: Leave-one-out cross-validation

Dataset

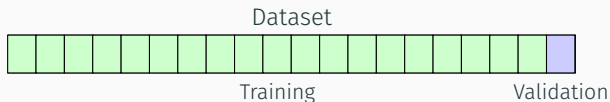


Model assessment: Leave-one-out cross-validation

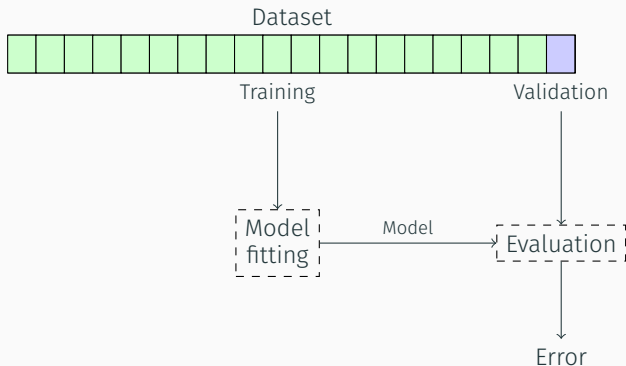
Dataset



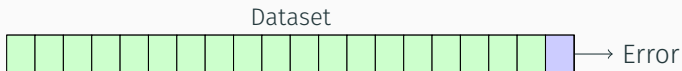
Model assessment: Leave-one-out cross-validation



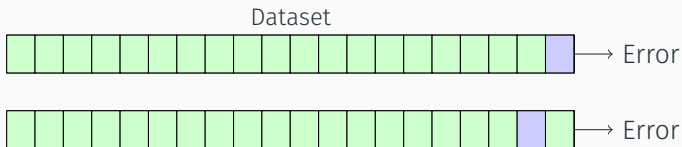
Model assessment: Leave-one-out cross-validation



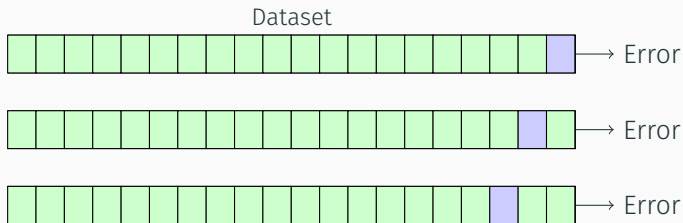
Model assessment: Leave-one-out cross-validation



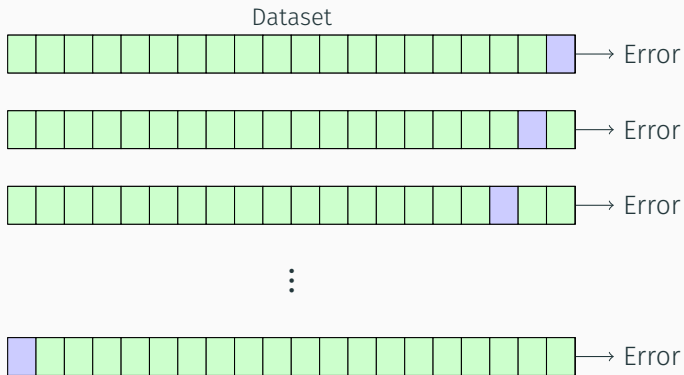
Model assessment: Leave-one-out cross-validation



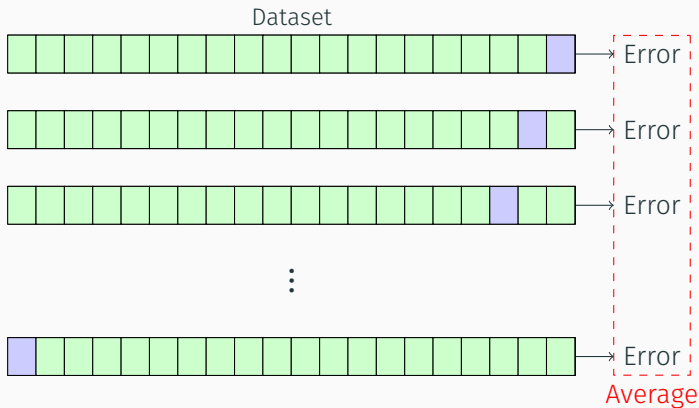
Model assessment: Leave-one-out cross-validation



Model assessment: Leave-one-out cross-validation



Model assessment: Leave-one-out cross-validation



Model assessment: Leave-one-out cross-validation

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

- + Uses all data to train models
- + Not dependent on arbitrary data splits
- Computationally expensive
- Effectively gives a point estimate of the error

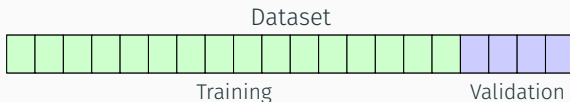


Model assessment: Cross-validation

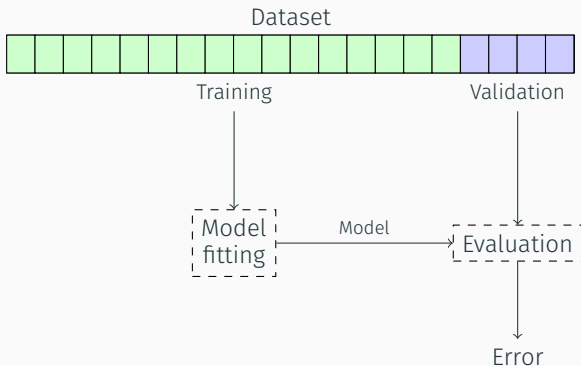
Dataset



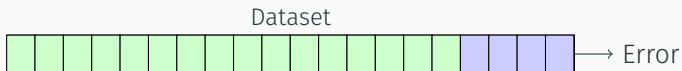
Model assessment: Cross-validation



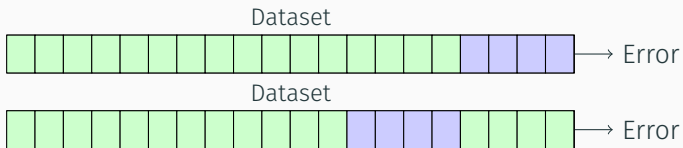
Model assessment: Cross-validation



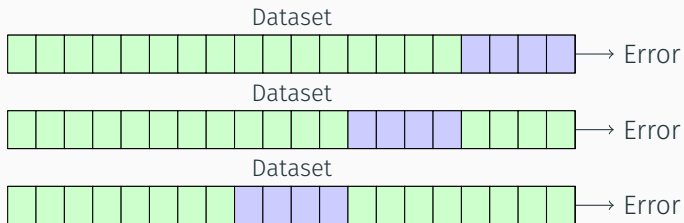
Model assessment: Cross-validation



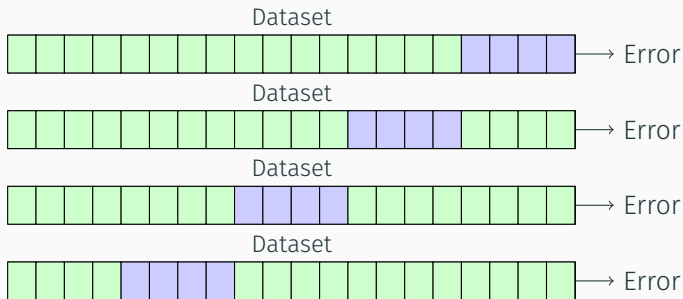
Model assessment: Cross-validation



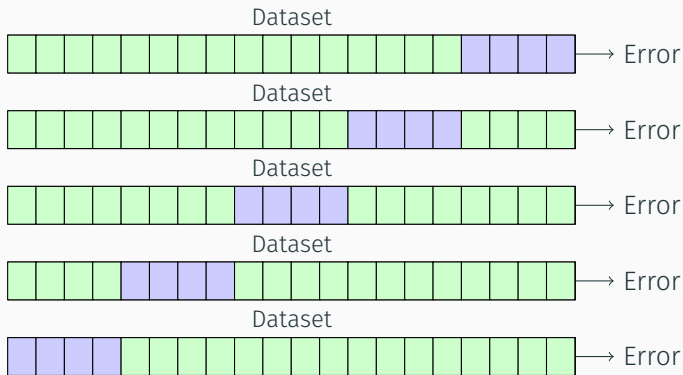
Model assessment: Cross-validation



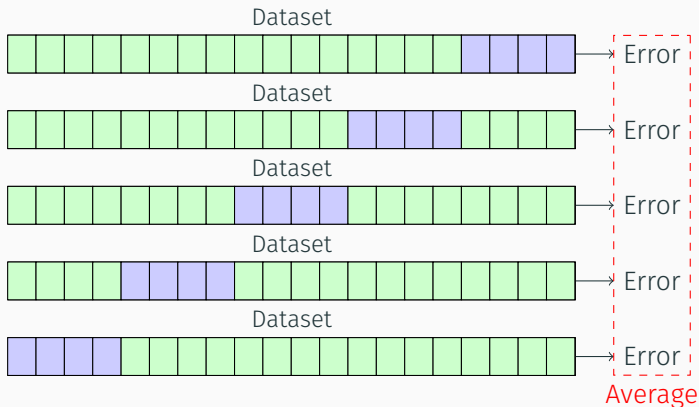
Model assessment: Cross-validation



Model assessment: Cross-validation



Model assessment: Cross-validation



Model assessment: Cross-validation

Fits k models for $n > k$ datapoints, each leaving n/k datapoints out for 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**

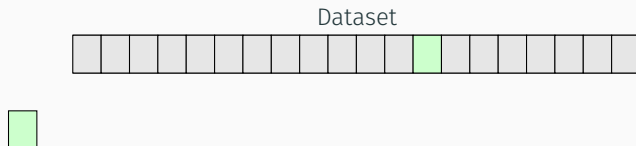


Model assessment: The bootstrap

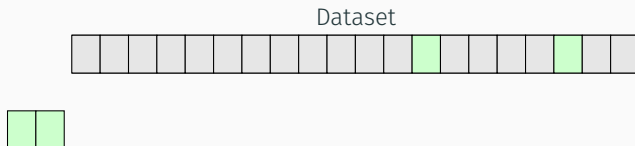
Dataset



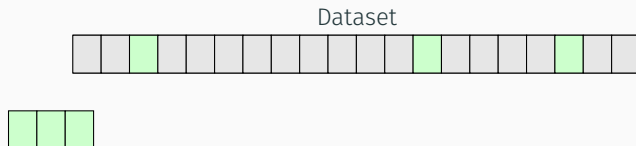
Model assessment: The bootstrap



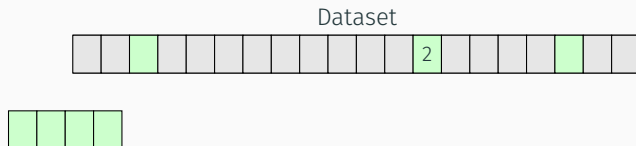
Model assessment: The bootstrap



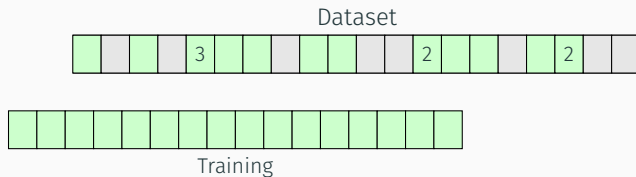
Model assessment: The bootstrap



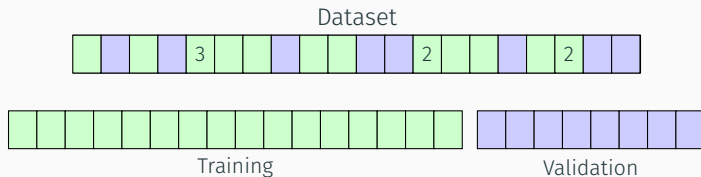
Model assessment: The bootstrap



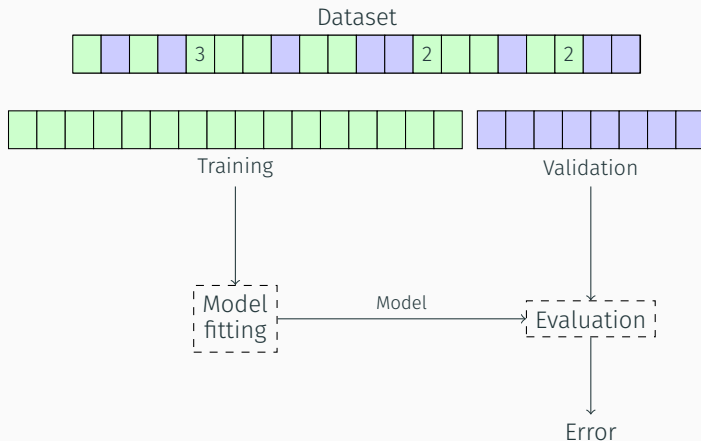
Model assessment: The bootstrap



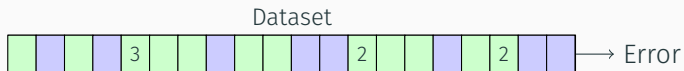
Model assessment: The bootstrap



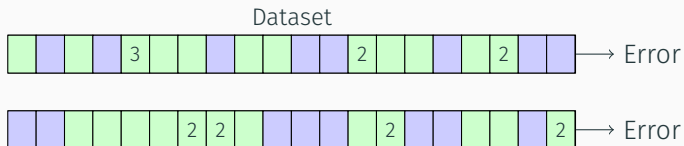
Model assessment: The bootstrap



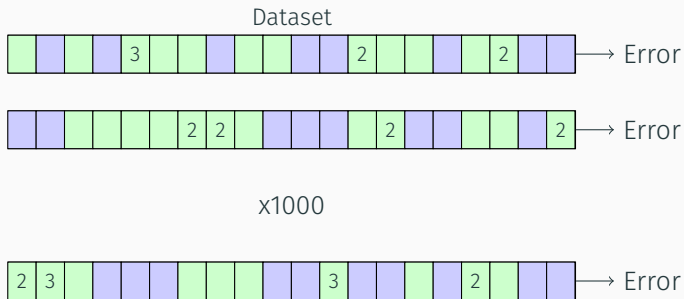
Model assessment: The bootstrap



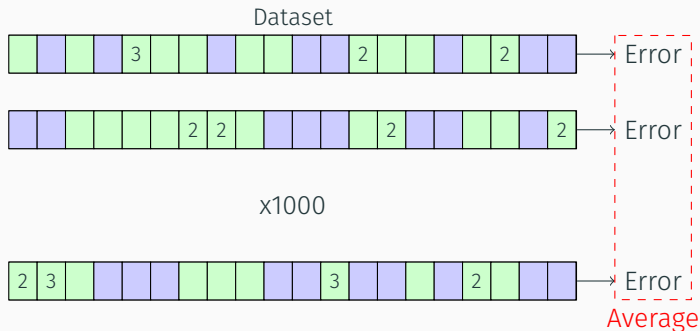
Model assessment: The bootstrap



Model assessment: The bootstrap



Model assessment: The bootstrap



Model assessment: The bootstrap

Fits x models with m datapoints each, sampled from the original dataset with replacement.

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



Model assessment: Summary

- Model assessment should **always** happen out-of-sample
- If n is big (≥ 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 parameters



Model selection and assessment



UNIVERSITETET
I OSLO

Model selection and assessment

- Model assessment 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 assessment

- Model assessment 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

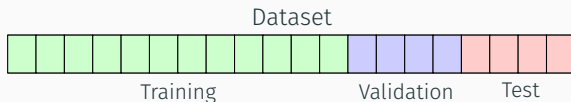


Model selection and assessment: Train/validation/test

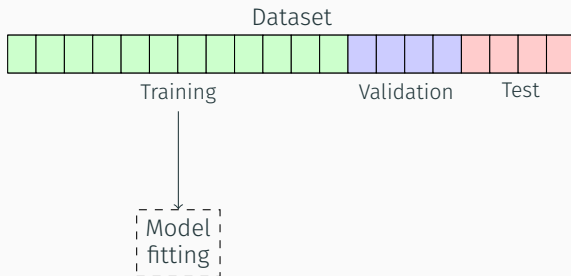
Dataset



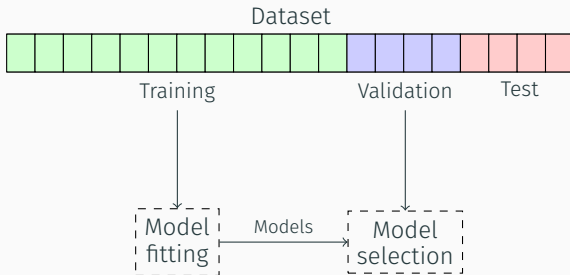
Model selection and assessment: Train/validation/test



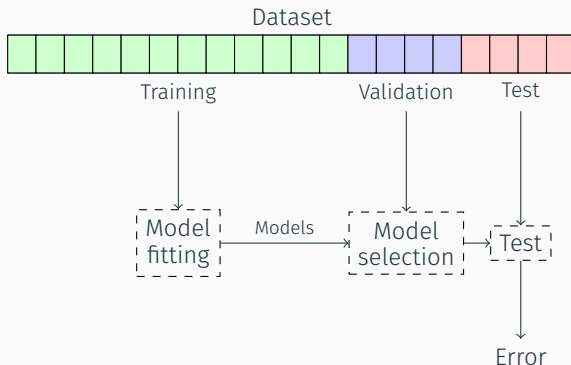
Model selection and assessment: Train/validation/test



Model selection and assessment: Train/validation/test



Model selection and assessment: Train/validation/test

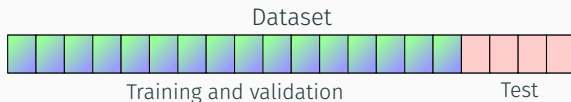


Model selection and assessment: Nests cross-validation

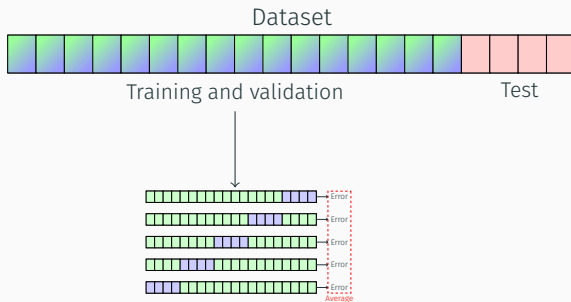
Dataset



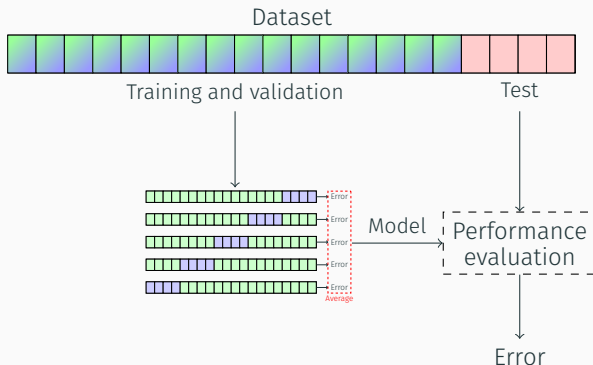
Model selection and assessment: Nested cross-validation



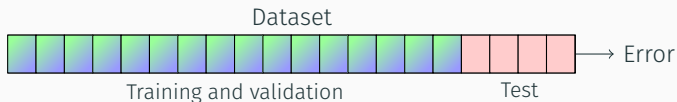
Model selection and assessment: Nested cross-validation



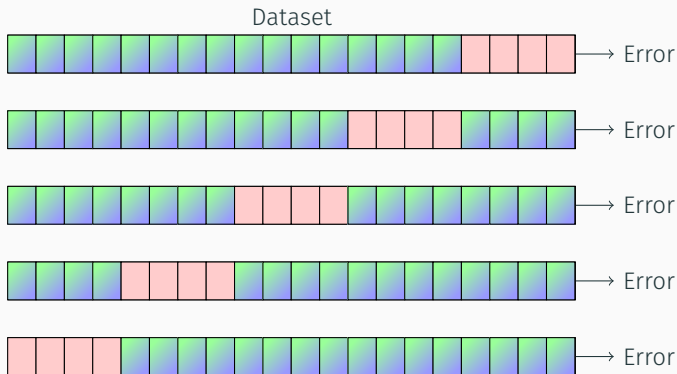
Model selection and assessment: Nested cross-validation



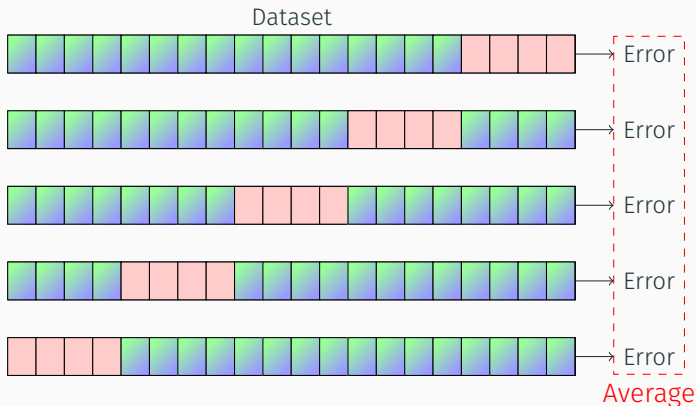
Model selection and assessment: Nested cross-validation



Model selection and assessment: Nested cross-validation



Model selection and assessment: Nested cross-validation



Model selection and assessment: Summary

- Whenever a choice is made on the basis of performance in a dataset, the performance of the chosen model on that dataset is going to be biased.
- If n is big (≥ 10000), a single train/validation/test split is often sufficient
- If possible, use nested cross-validation to select the best model *and* estimate the out-of-sample error

