# Lecture 8
# Estimating Test Error

16.05.2024

Slides from Elements of ML
by Isabel Valera and Jilles Vreeken

UNIVERSITÄT DES SAARLANDES

ZBI

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

# Introduction

**Resampling** or **subsampling** denotes methods that repeatedly draw samples from the training data to

- **learn** about the **variability** of the fitted models as the training set changes (model variability)
- **estimate** the test **error** which is hard to estimate (model assessment)
- **optimize** model **performance** in terms of test error (model selection)

Resampling methods can be **computationally expensive**

- repeated model fitting on a potentially large number of data sets
- today's computing power allows for this expense

# Cross Validation

The prime technique for estimating test error

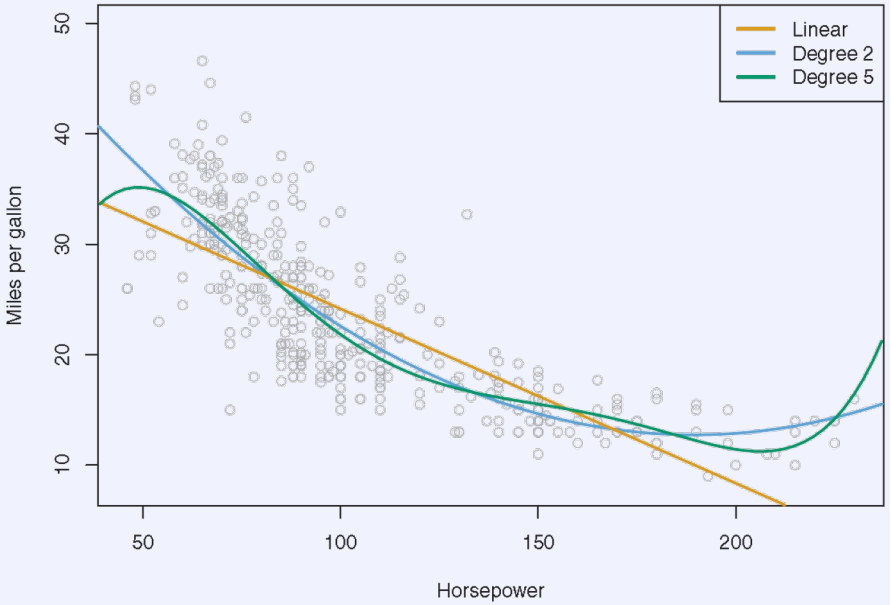
- also often used for model selection
- main idea: part of the training data is set aside for testing
  - this is known as the **test set**
- here exemplified with regression

- **Cross-validation can be performed in different ways, next we discuss different alternatives.**

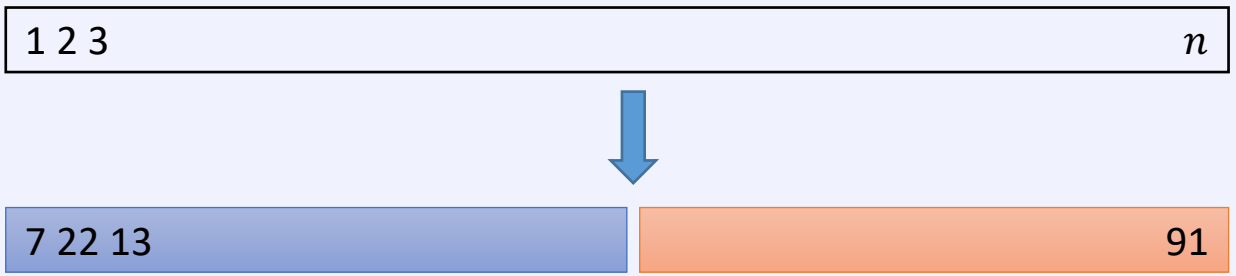
# 1. Validation Set Approach

Randomly divide training data into a **training** (training set, blue) and **testing** (validation set, orange)

- model is fit on the training set
- test error is estimated on the validation set (here half the data)
- here applied to the Auto data set
  - 392 observations
- nonlinear dependence of **mpg** on **horsepower**



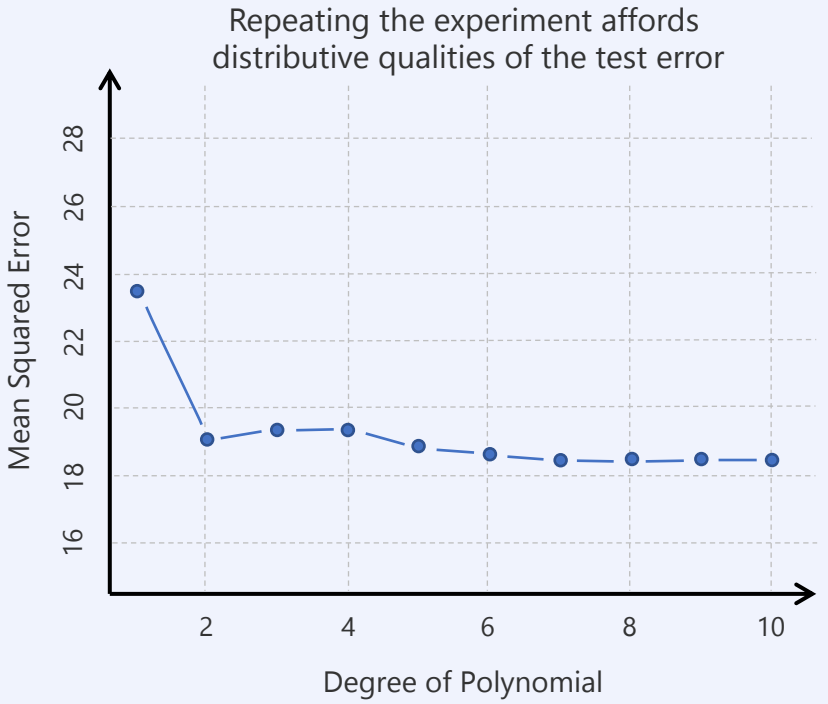

$$\texttt{mpg} = \beta_0 + \beta_1 \times \texttt{horsepower} + \beta_2 \times \texttt{horsepower}^2 + \varepsilon$$



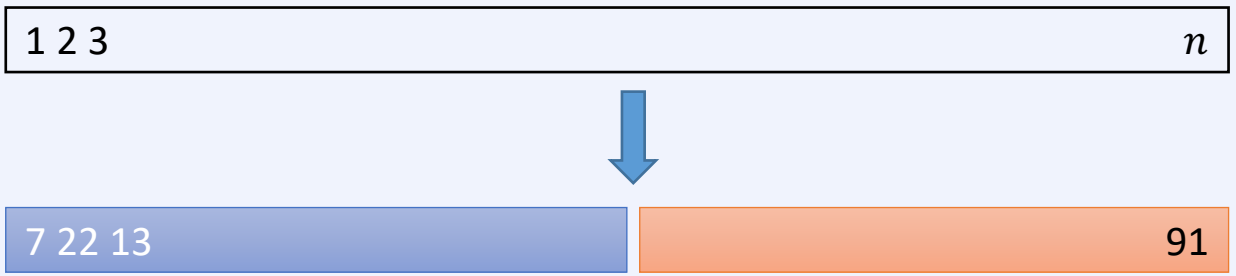
# 1. Validation Set Approach

Randomly divide training data into a **training** (training set, blue) and **testing** (validation set, orange)

- model is fit on the training set
- test error is estimated on the validation set (here half the data)
- here applied to the Auto data set
  - 392 observations
- nonlinear dependence of `mpg` on `horsepower`




*Validation error for one random 50-50 splits into training and validation set*



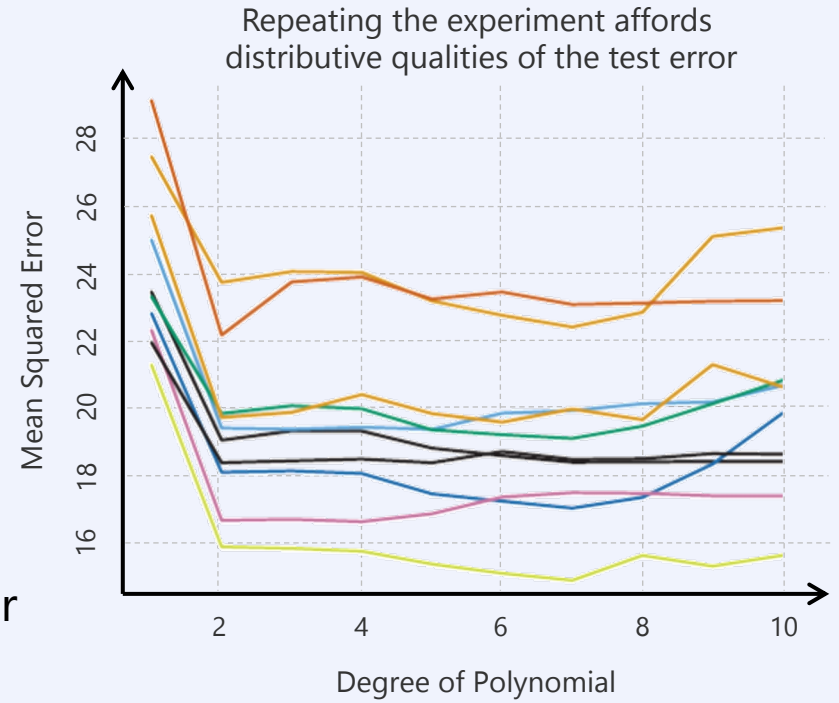
# 1. Validation Set Approach

## Observations

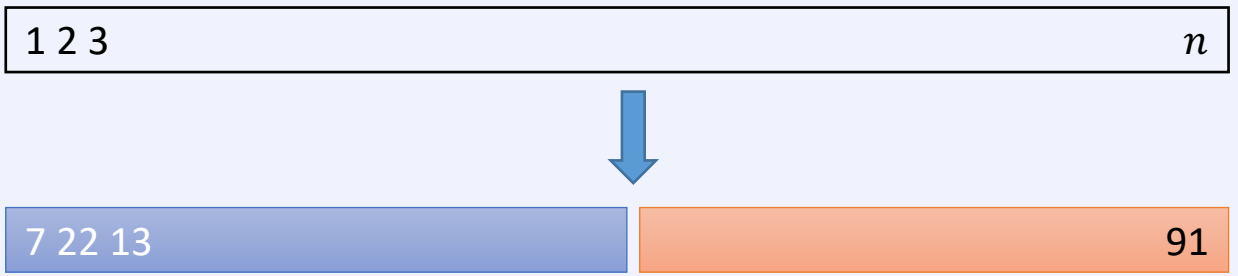
1. test error can vary widely
2. quadratic term affords large improvement of test error
3. higher-order terms not of benefit

## Problems

1. high variability of estimated test error
2. training set becomes smaller; model may be suboptimal (undertrained); test error may be overestimated



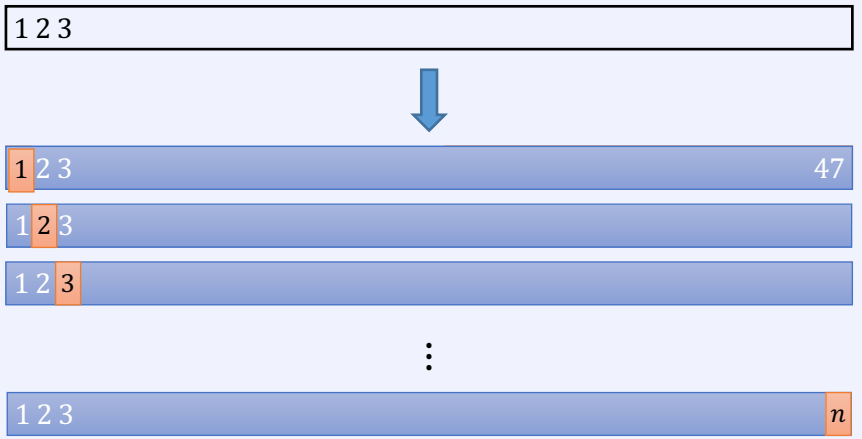

*Validation error for one random 50-50 splits into training and validation set*



# 2. Leave-one-out Cross Validation (LOOCV)

Set **one data point aside** for testing

- advantages
  - training set is as large as can be, thus there is little bias
- problems
  - because only one point is tested, there is high variance




Repeating this for each data point averages out the variance

$$MSE_i = (y_i - \hat{y}_i)^2$$

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

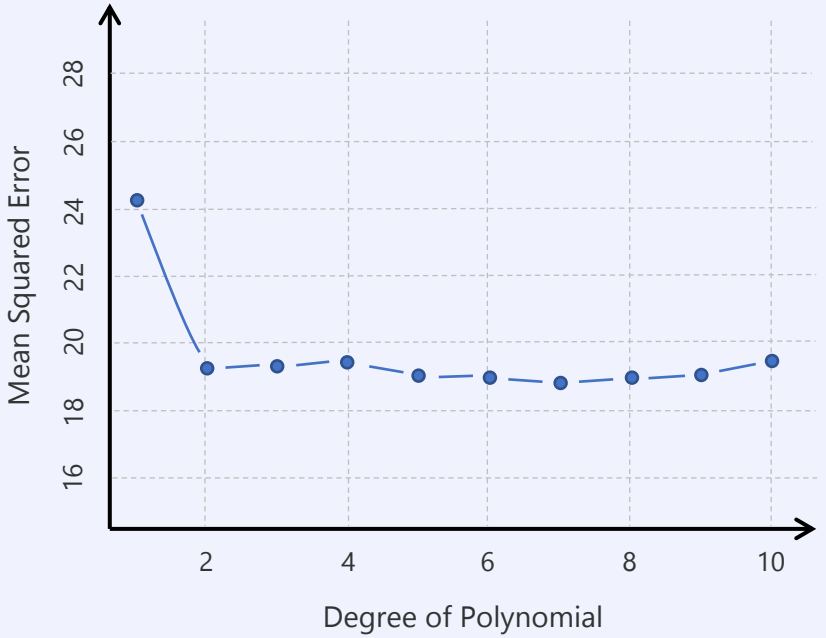
This process is **deterministic**

- repeating yields the same result

# 2. Leave-one-out Cross Validation (LOOCV)

Set **one data point aside** for testing

- advantages
  - training set is as large as can be, thus there is little bias
- problems
  - because only one point is tested, there is high variance




Repeating this for each data point averages out the variance

$$MSE_i = (y_i - \hat{y}_i)^2$$

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

This process is **deterministic**

- repeating yields the same result

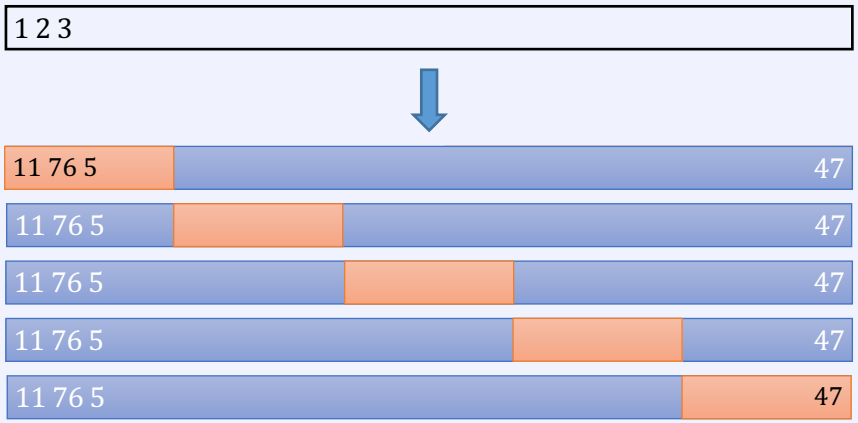
# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$




In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV

# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

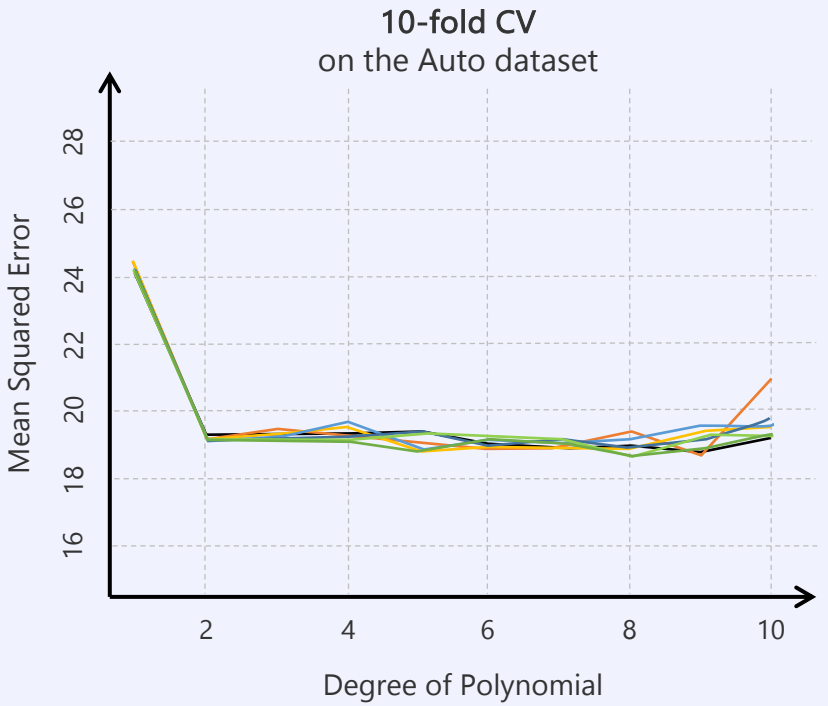
- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV




- 10-fold CV was run nine times, each with a different random split of the data
- variance of the test errors reduced by averaging test errors over each fold

# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

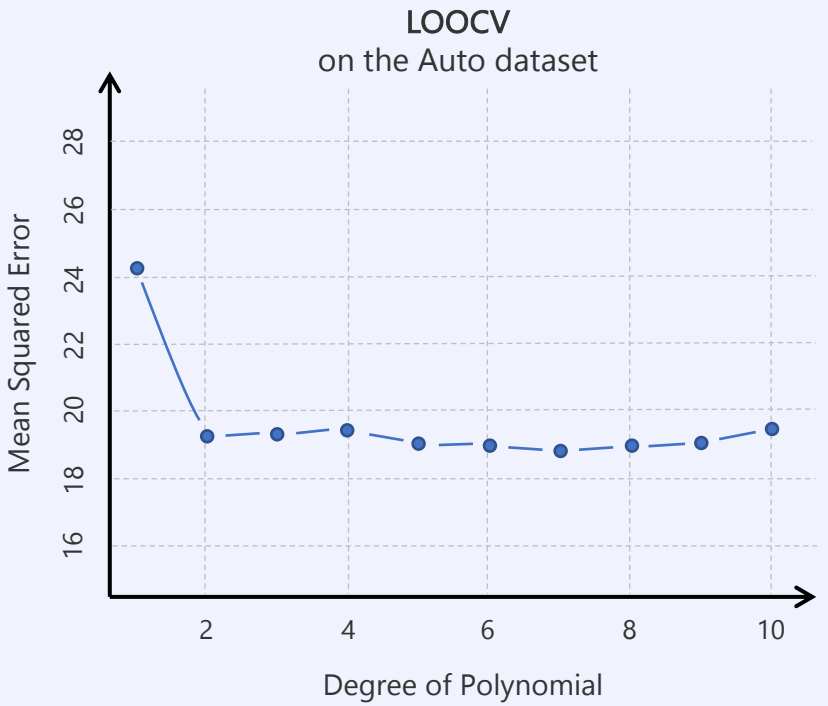
- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV



# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

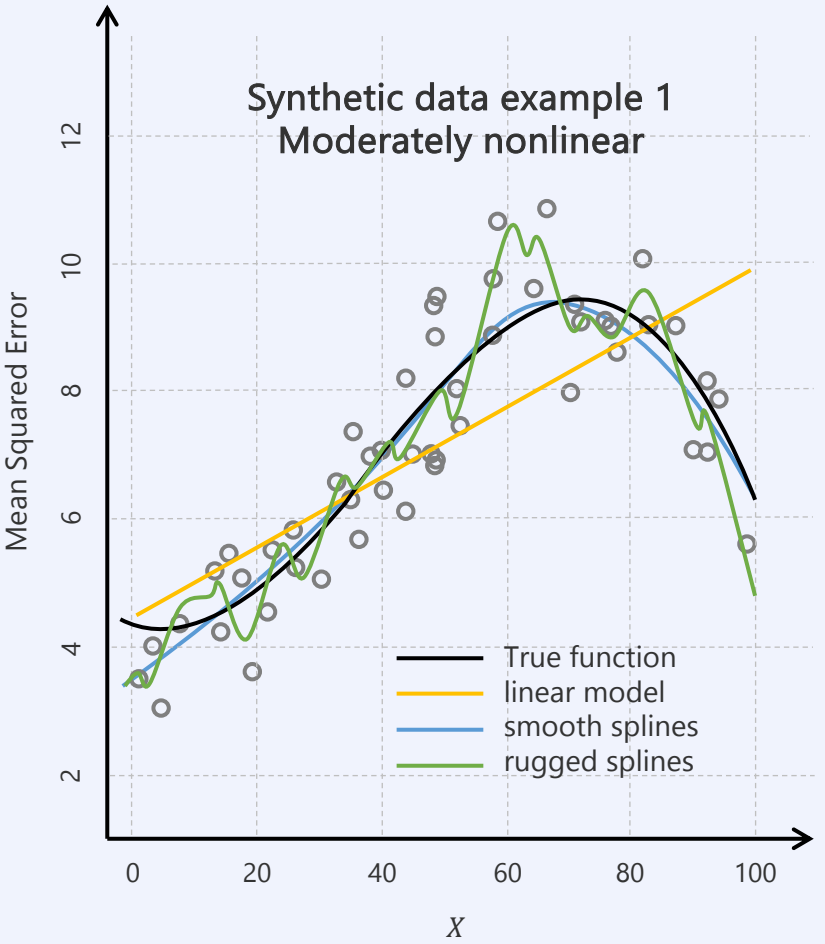
- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV



# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

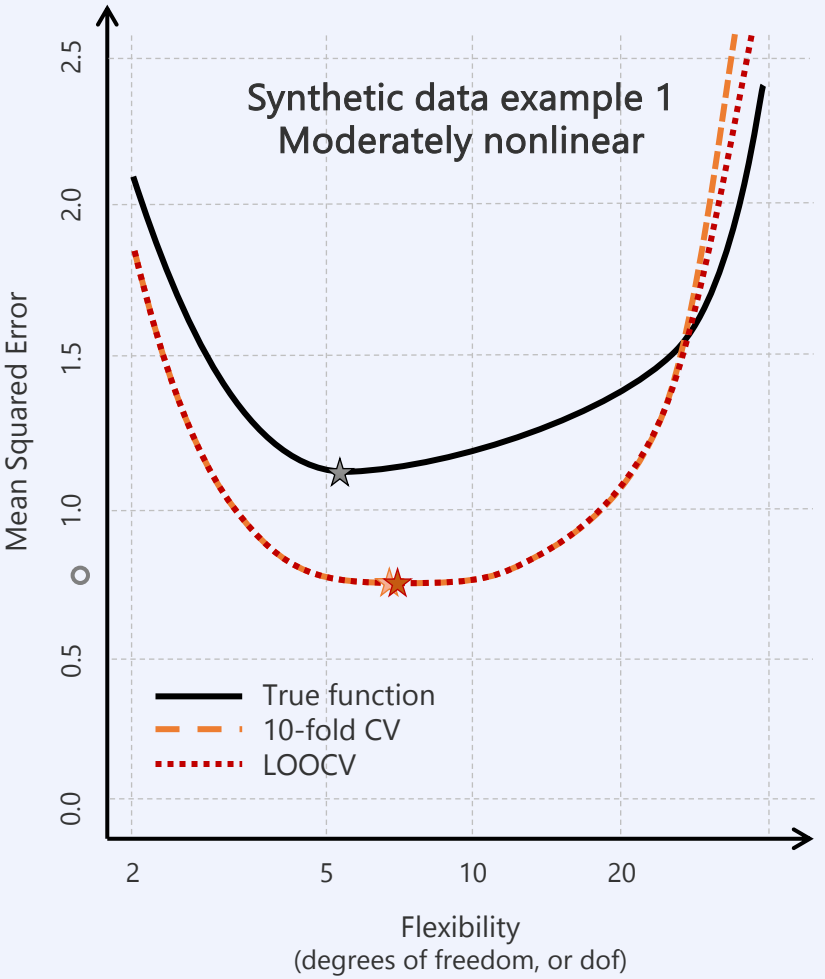
$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV

*Test error is underestimated*




*Stars indicate curve minima*

# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

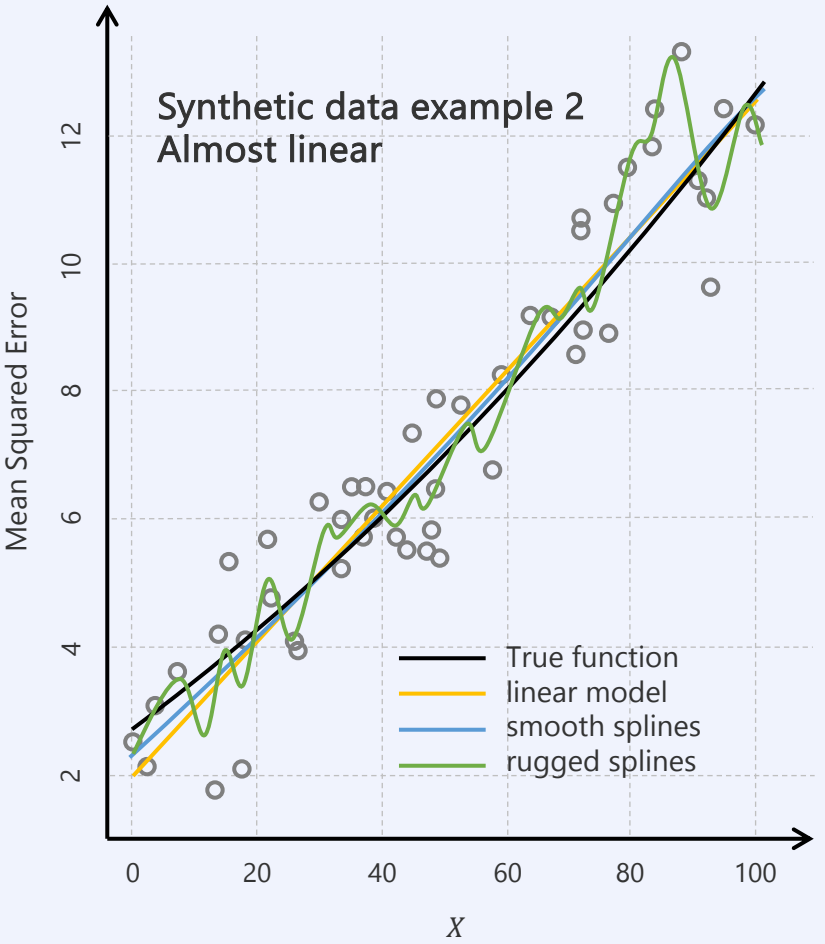
- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV



# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

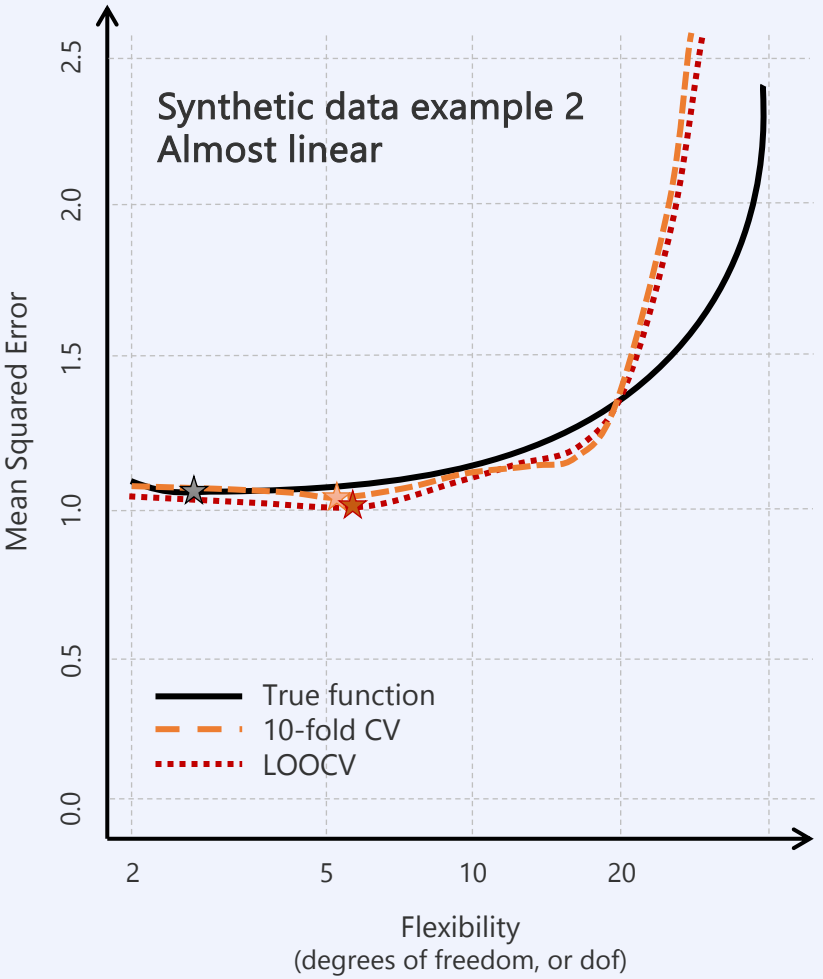
$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV

*Test error is overestimated for complex models*




*Stars indicate curve minima*

# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

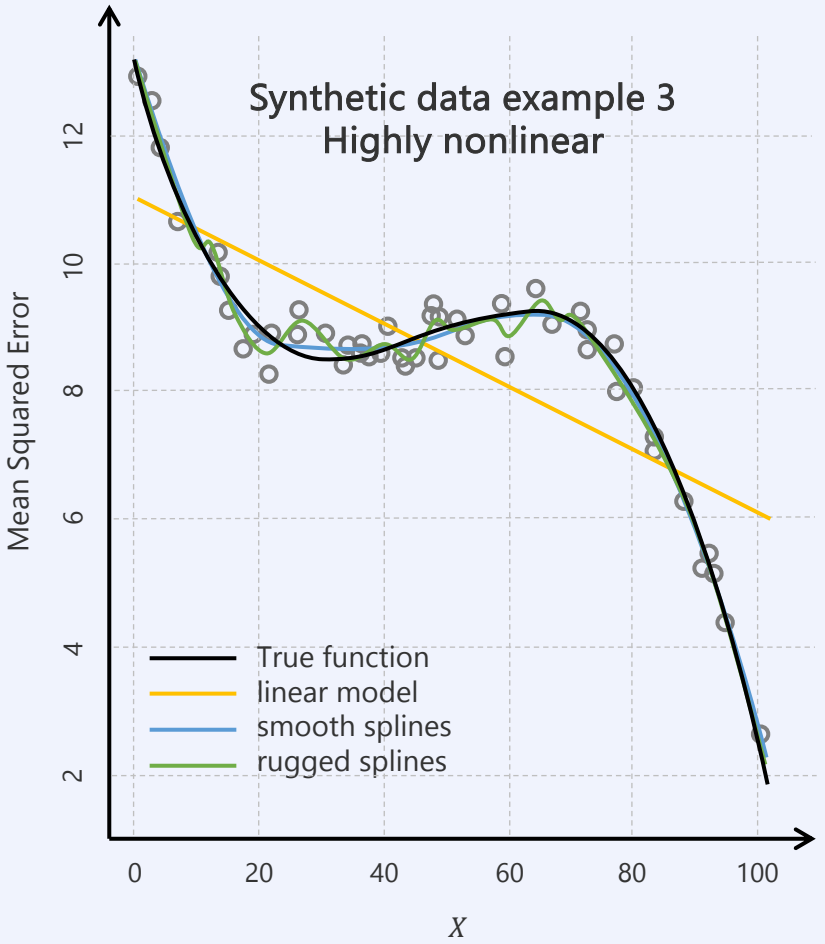
- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV



# 3. $k$-fold Cross Validation

Randomly divide the data into $k$ folds

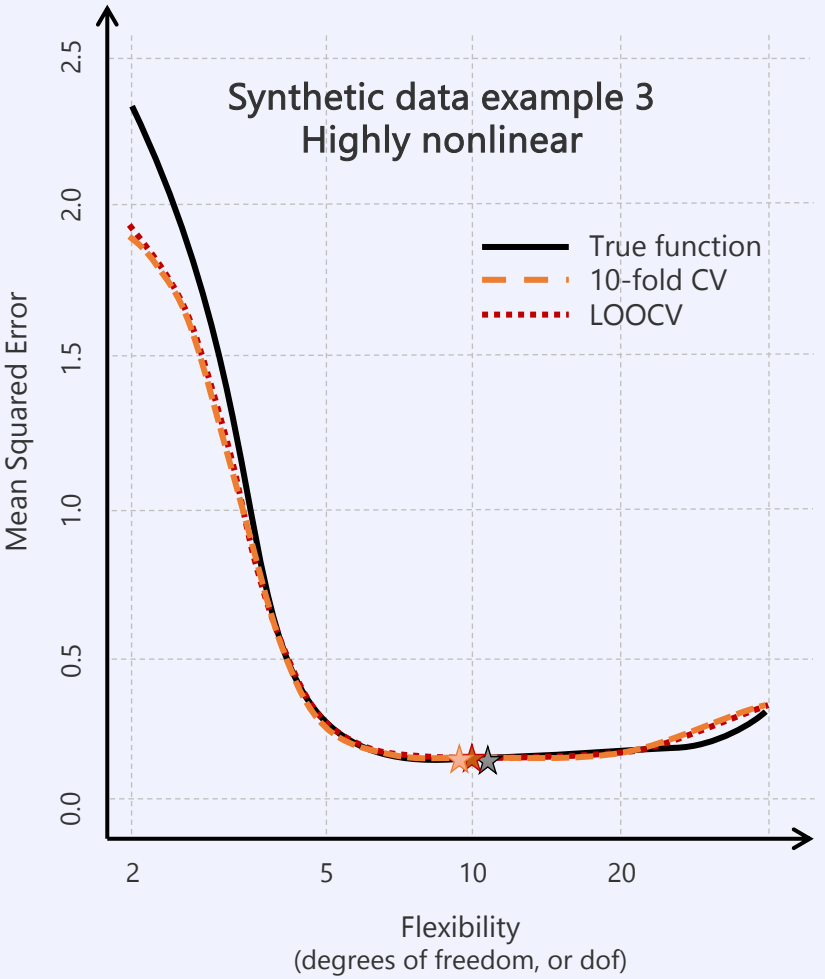
- in $k$ repetitions, select each fold once for testing, with the remaining $k-1$ folds for training
- this gives $k$ estimates of the test error
- the final estimate is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

- LOOCV is $k$-fold CV with $k = n-1$

In practice, we use $k = 5, 10$

- we get $k$ samples of the test error
- saves computing time but may increase bias over LOOCV



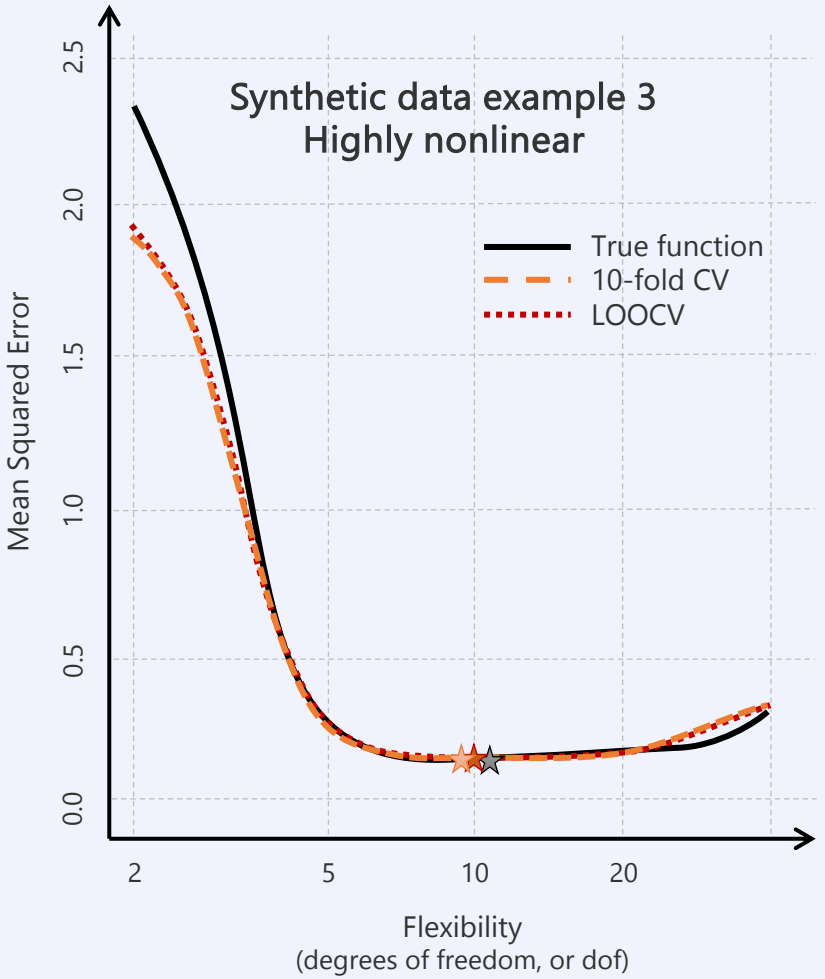

*Stars indicate curve minima*

# 3. $k$-fold Cross Validation

Often, we are more interested in the **location of the minimum** (the complexity of the optimal model) than in the test error itself

- test errors vary
- with CV we can estimate the variance of the test error

**General rule:** Choose the simplest model that is within 1 standard error of the optimum



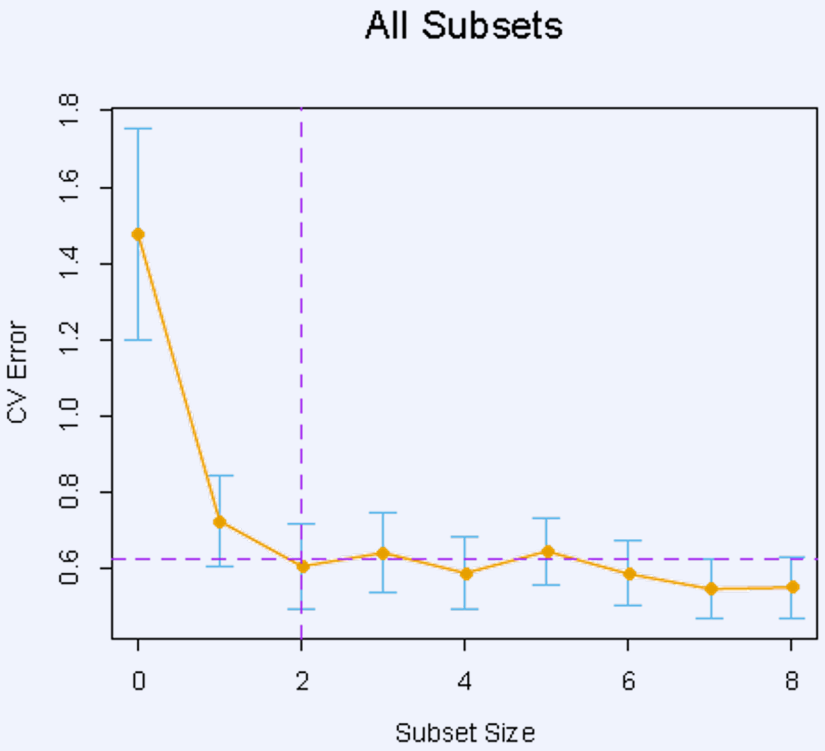

*Stars indicate curve minima*

# 3. $k$-fold Cross Validation

Often, we are more interested in the **location of the minimum** (the complexity of the optimal model) than in the test error itself

- test errors vary
- with CV we can estimate the variance of the test error

**General rule:** Choose the simplest model that is within 1 standard error of the optimum



# Bias-Variance Tradeoff for $k$-fold CV

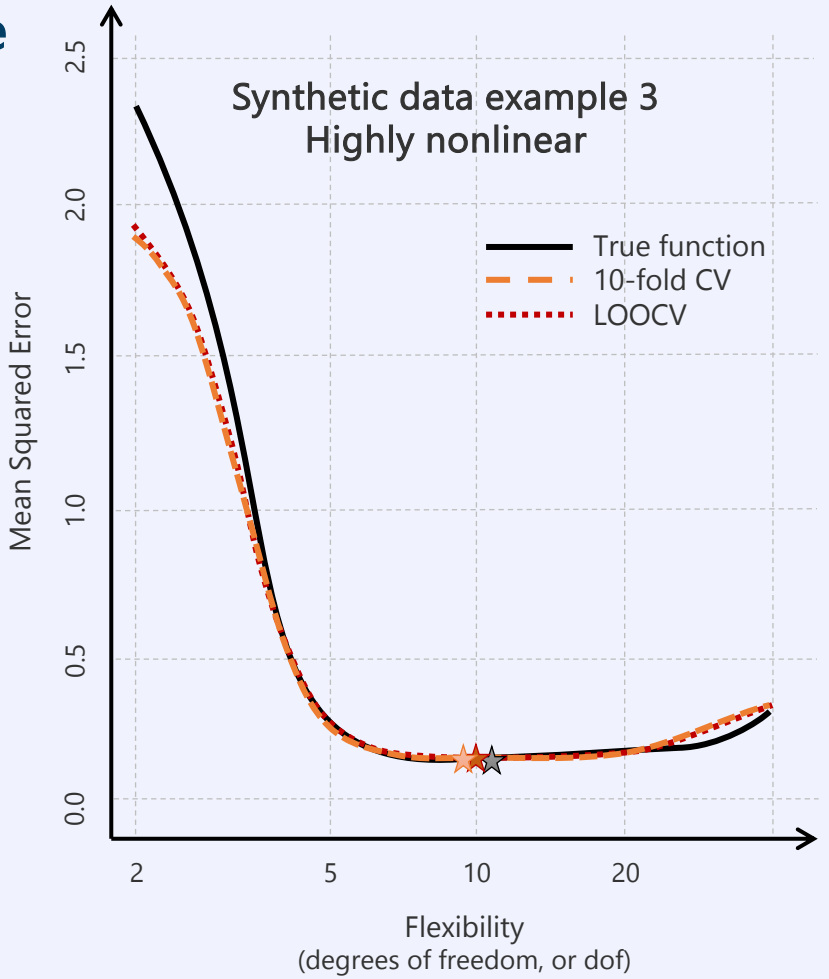
Surprisingly, $k$-fold CV often gives **more accurate** error estimates than LOOCV

- this is due to the bias-variance tradeoff

Increasing size of the training set **reduces bias** but **increases variance**

- LOOCV averages over models trained on **almost the same data**, hence, all **outputs** are **highly correlated**
- $k$-fold CV has **less overlap** in training data, and outputs are **less correlated**
- variance of the average of the outcomes of $k$ i.d. events with correlation $\rho$, each with variance $\sigma^2$ is $\rho\sigma^2 + \frac{1-\rho}{k}\sigma^2$

**identically distributed**




*Stars indicate curve minima*

# Cross Validation for Classification

Here, we use **misclassification error**

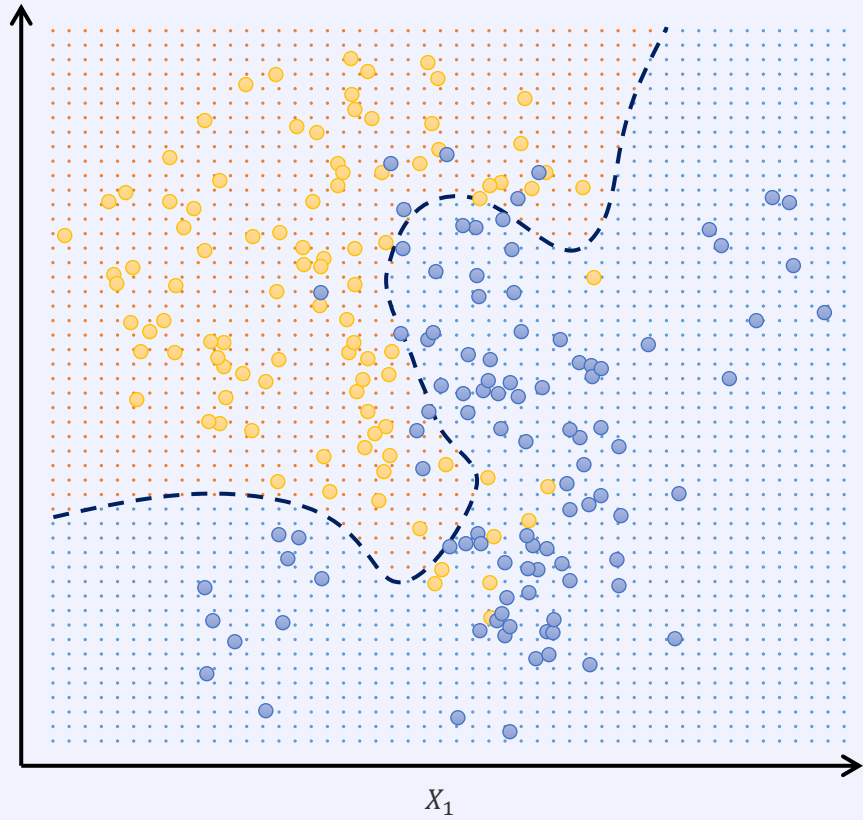
- e.g. for LOOCV

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i \;,\; \mathrm{Err}_i = \mathrm{I}(\mathrm{y_i} \neq \hat{y}_i)$$

But it can be done similarly to get estimates of other performance metrics.

**Example** Logistic regression

- simulated two-dimensional data
- Bayes error 0.133



# Cross Validation for Classification

We use **misclassification error**

- e.g. for LOOCV

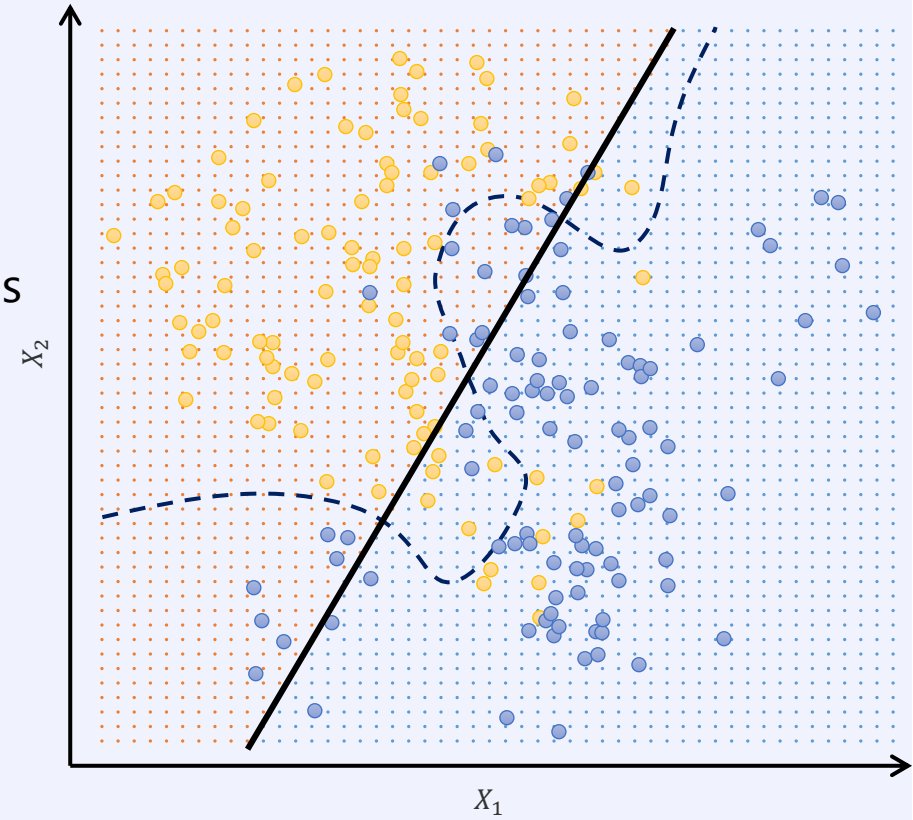
$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i \;,\; \mathrm{Err}_i = \mathrm{I}(\mathrm{y_i} \neq \hat{y}_i)$$

But it can be done similarly to get estimates of other performance metrics.

**Example** Logistic regression

- simulated two-dimensional data from Chapter 2
- Bayes error 0.133

1. standard linear logistic regression
   - true test error 0.201 (accessible since data are simulated)




*Degree = 1*

# Cross Validation for Classification

We use **misclassification error**

- e.g. for LOOCV

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i \;,\; \mathrm{Err}_i = \mathrm{I}(\mathrm{y_i} \neq \hat{y}_i)$$

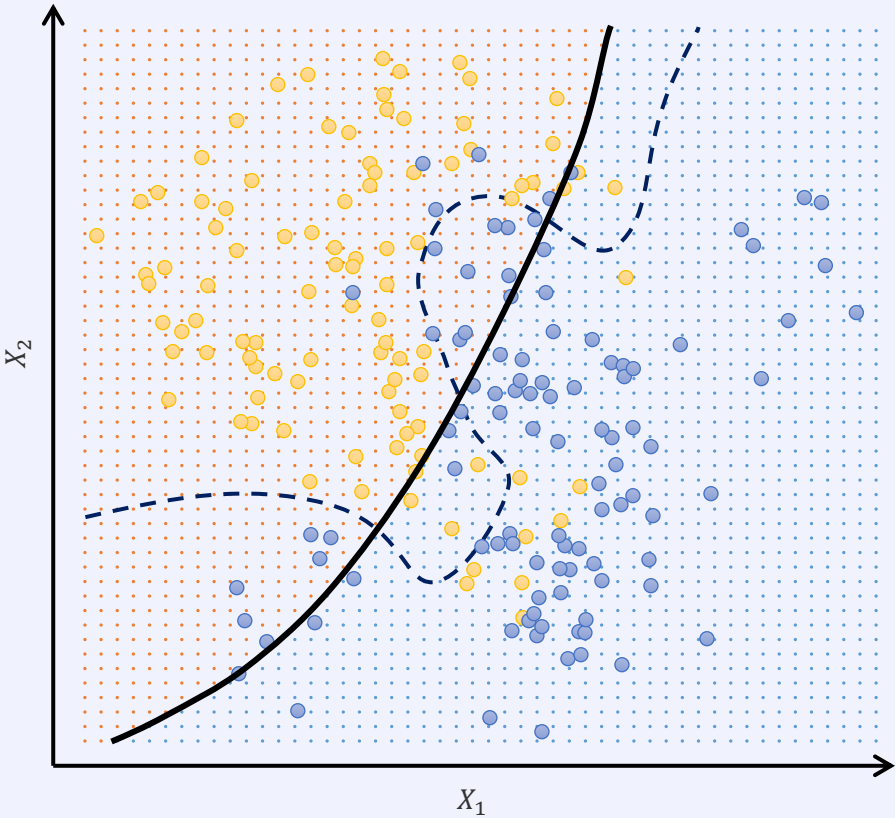
**Example** Logistic regression

- simulated two-dimensional data from Chapter 2
- Bayes error 0.133

2. quadratic logistic regression

$$\log\frac{p}{1-p} = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2$$

   - true test error 0.197




*Degree = 2*

# Cross Validation for Classification

We use **misclassification error**

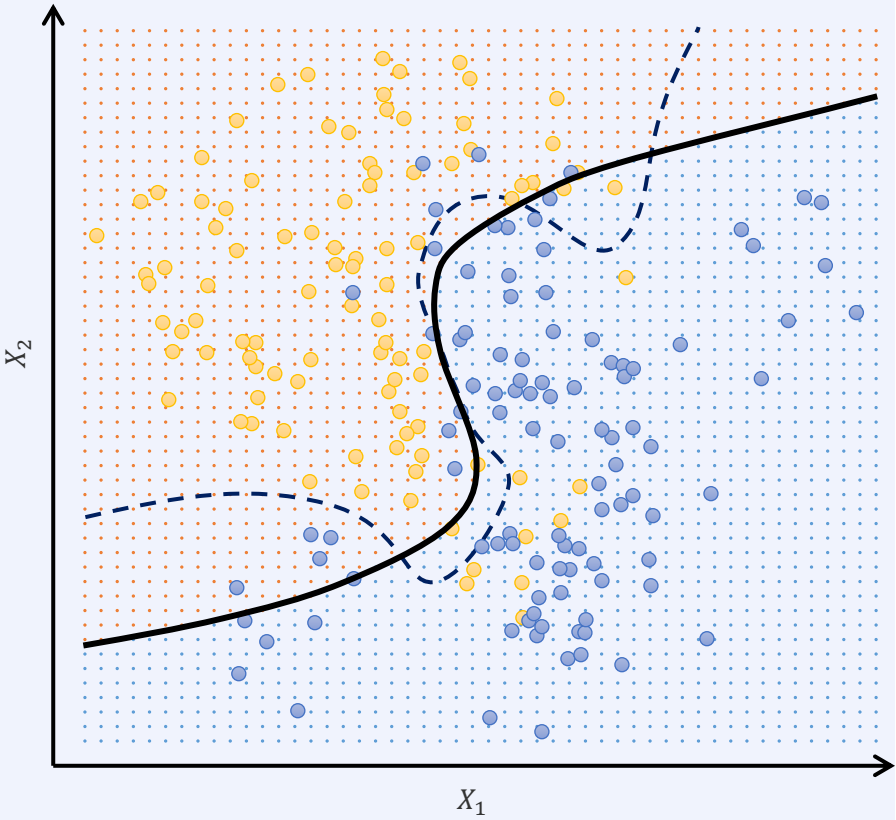
- e.g. for LOOCV

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i \;,\; \mathrm{Err}_i = \mathrm{I}(\mathrm{y_i} \neq \hat{y}_i)$$

**Example** Logistic regression

- simulated two-dimensional data from Chapter 2
- Bayes error 0.133

3. cubic logistic regression
   - true test error 0.160




*Degree = 3*

# Cross Validation for Classification

We use **misclassification error**

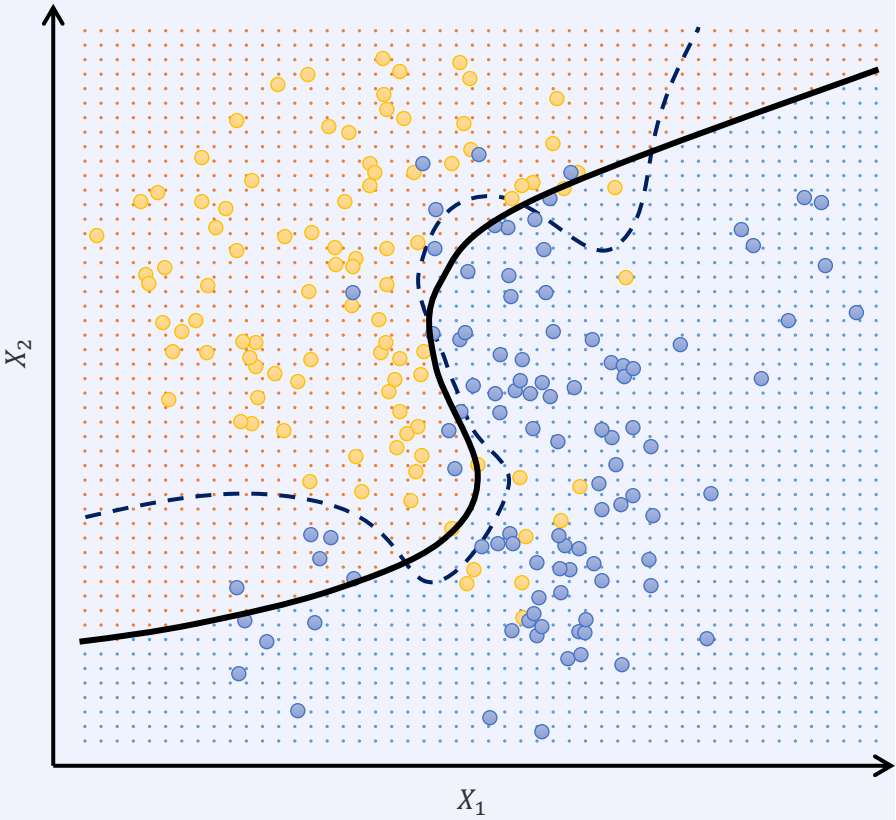
- e.g. for LOOCV

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i \;,\; \mathrm{Err}_i = \mathrm{I}(\mathrm{y_i} \neq \hat{y}_i)$$

**Example** Logistic regression

- simulated two-dimensional data from Chapter 2
- Bayes error 0.133

4. quartic logistic regression
   - slightly increased test error




*Degree = 4*

# Cross Validation for Classification

We use **misclassification error**

- e.g. for LOOCV

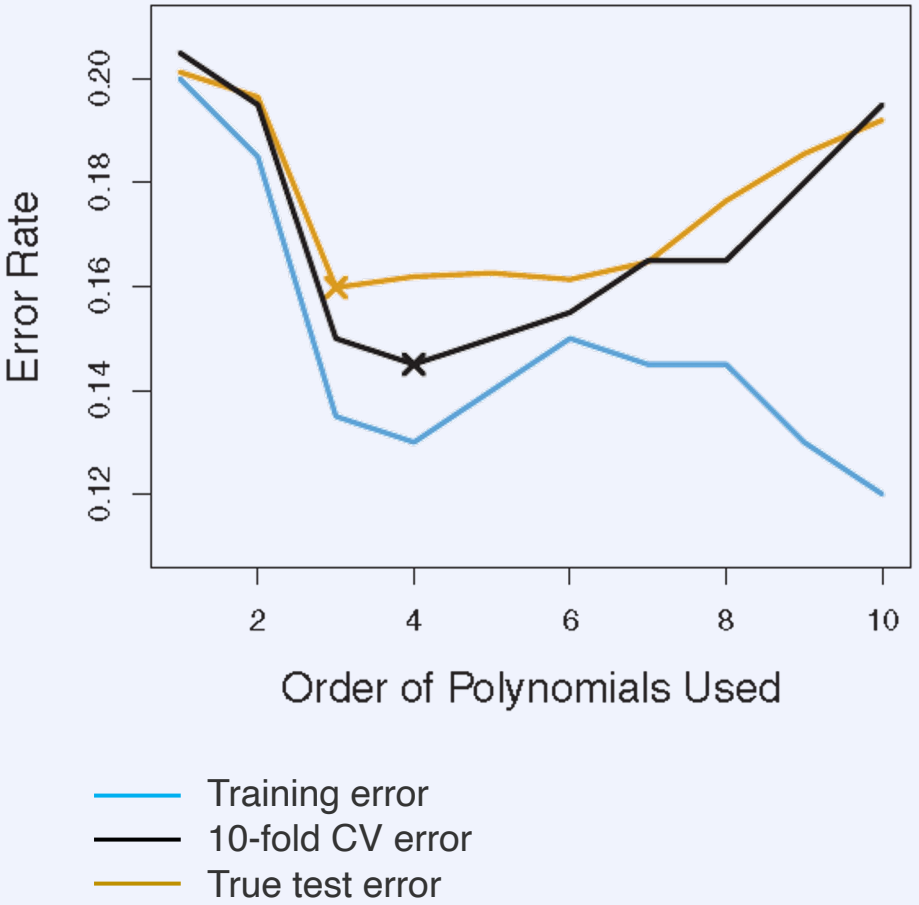
$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i \;,\; \mathrm{Err}_i = \mathrm{I}(\mathrm{y_i} \neq \hat{y}_i)$$

**Example** Logistic regression

- simulated two-dimensional data
- Bayes error 0.133

CV estimates of test error

- for logistic regression



# Doing Cross Validation Right

Assume a scenario with many features (e.g. from genomics)

Data generation process

- $n$=50 samples, two balanced classes
- $d$=5000 quantitative standard Gaussian features, independent of class labels

Model building process

1. select 100 features with strong univariate correlation with the output
2. train several classifiers on this subset of features
3. use CV to select the final classifier

Resulting average test error is 3%.

**How is that possible?**

**The true test error of any classifier is 50% as features are independent of the class variable.**

- the chosen features have an **unfair advantage**, because they were chosen on the basis of **all samples**.
- this obviates the separation of training, validation and test set.

# Doing Cross Validation Right

Dataset:

$N$=50 samples, two balanced classes
$p$=5000 quantitative standard
Gaussian features, independent of class labels


## Wrong way

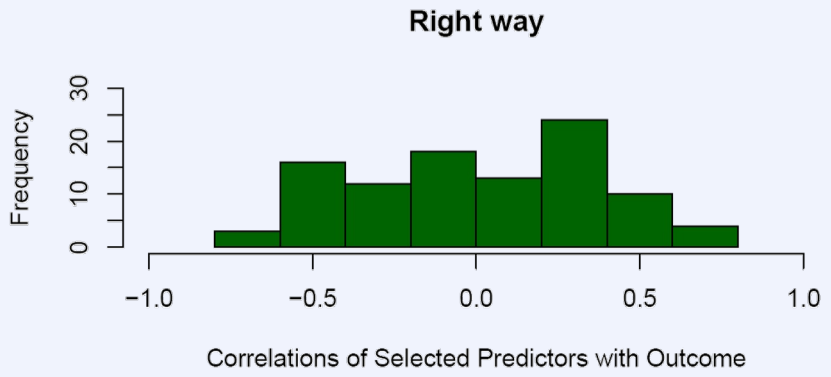
1. select 100 features based on highest correlation over all 50 samples
2. choose random set of 10 samples
3. compute correlations of the 100 selected features with the output over just these 10 samples

Mean correlation of selected features with outcome is 28%

# Doing Cross Validation Right

Dataset:

$N$=50 samples, two balanced classes
$p$=5000 quantitative standard
Gaussian features, independent of class labels


## Correct implementation

1. divide samples into $K$ classes ($K$-fold CV)
2. for each $k$=1,..., $K$ do
    1. find subset of features highly correlated with the outcome on all samples except those on fold $\boldsymbol{k}$
    2. use these to build multivariate model
    3. use classifier to predict the classes of the samples in fold $k$
3. accumulate errors on all $K$ folds to compute the correlation

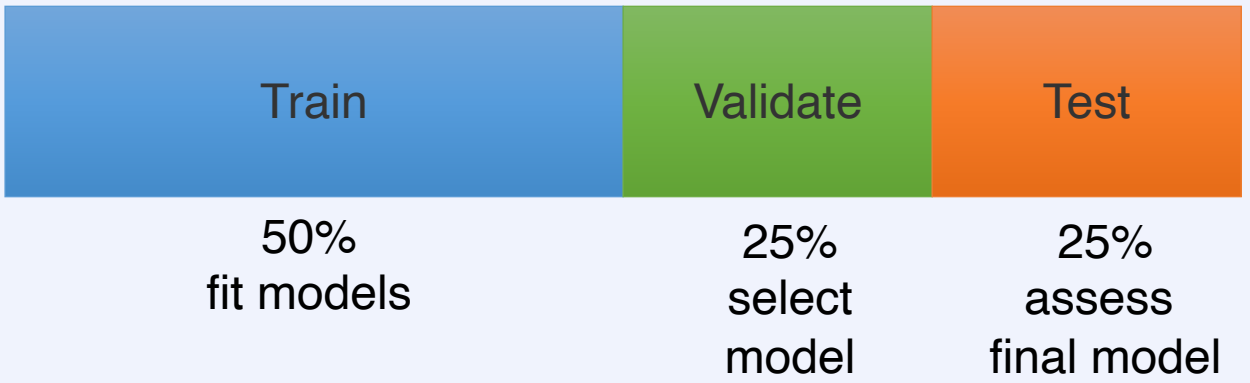
**In a multistep procedure, CV must be applied to entire sequence of steps!**

- initial unsupervised feature preprocessing (e.g., clustering) can be done before samples are left out (because it does not use class labels), e.g., select 50 predictors with highest variances across the training set

# Train-Validation-Test Paradigm

In practice, we should divide the data **three-way:**

1. training set for training/learning models
2. validation set for comparatively assessing model performance in order to select a model
3. test set in order to estimate the performance of the selected model, i.e., estimate how well the selected model generalizes.




Note: Partition percentages depend on the available amount of data.
Also, we can substitute tranining-validation approach by K-fold CV