

# **Introduction to Statistical Learning with Applications**

CM3: Cross-validation, model selection,  
and bias-variance

**Pedro L. C. Rodrigues**

# IN OUR PREVIOUS EPISODES...

Our current workflow

1

We want to estimate the values of  $Y$  based on predictors  $X_1, \dots, X_p$

2

We are given a set of  $N$  examples of  $y_i$  with corresponding  $x_{i1}, \dots, x_{ip}$

3

We estimate parameters  $\hat{\beta}$  that minimize  $\frac{1}{N} \sum_{i=1}^N (y_i - \beta^T x_i)^2$

4

We do statistical inference on the values of the  $\hat{\beta}_j$

$$x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}$$

# IN OUR PREVIOUS EPISODES...

We do **statistical inference** on the values of  $\hat{\beta}_j$

1

We are given a dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

2

We assume that each data point is

3

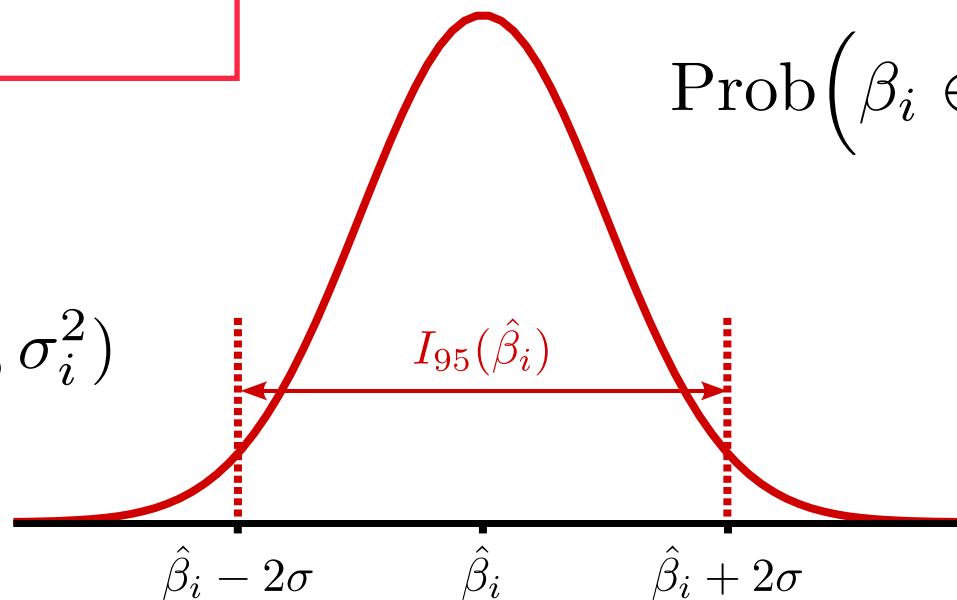
We assume the noise is **Gaussian** IID

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{bmatrix} \longrightarrow \hat{\beta}_i \sim \mathcal{N}(\beta_i, \sigma_i^2)$$

$$y_i = \beta_0 + \sum_{k=1}^p \beta_k x_{ik} + \varepsilon_i$$

$$\text{Prob}\left(\beta_i \in I_{95}(\hat{\beta}_i)\right) = 0.95$$



## IN OUR PREVIOUS EPISODES...

We do **statistical inference** on the values of  $\hat{\beta}_j$

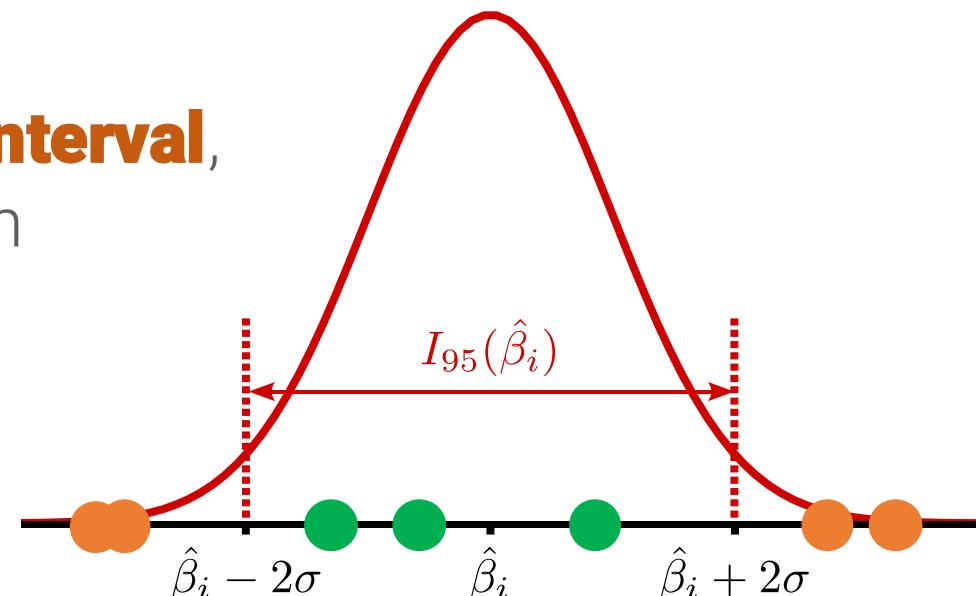
Note that even if we had data generated with  $\beta_i = 0$  our estimate is a RV

$$\hat{\beta}_i \sim \mathcal{N}(0, \sigma^2) \rightarrow \text{Very likely that } \hat{\beta}_i \neq 0$$

Therefore, we would like to check whether the number that we get for this estimate is one that could **indeed come from such distribution**

If it is a **number too far off from the confidence interval**, then it might in fact mean that the distribution with zero-mean is not a good one to describe the data

This is the intuition behind the statistical tests!



## IN OUR PREVIOUS EPISODES...

However, in practice we're often more interested in the **predictions!**

Is there a natural way of describing the statistics of the predicted values?

Remember that if we assume a linear model for the data, then

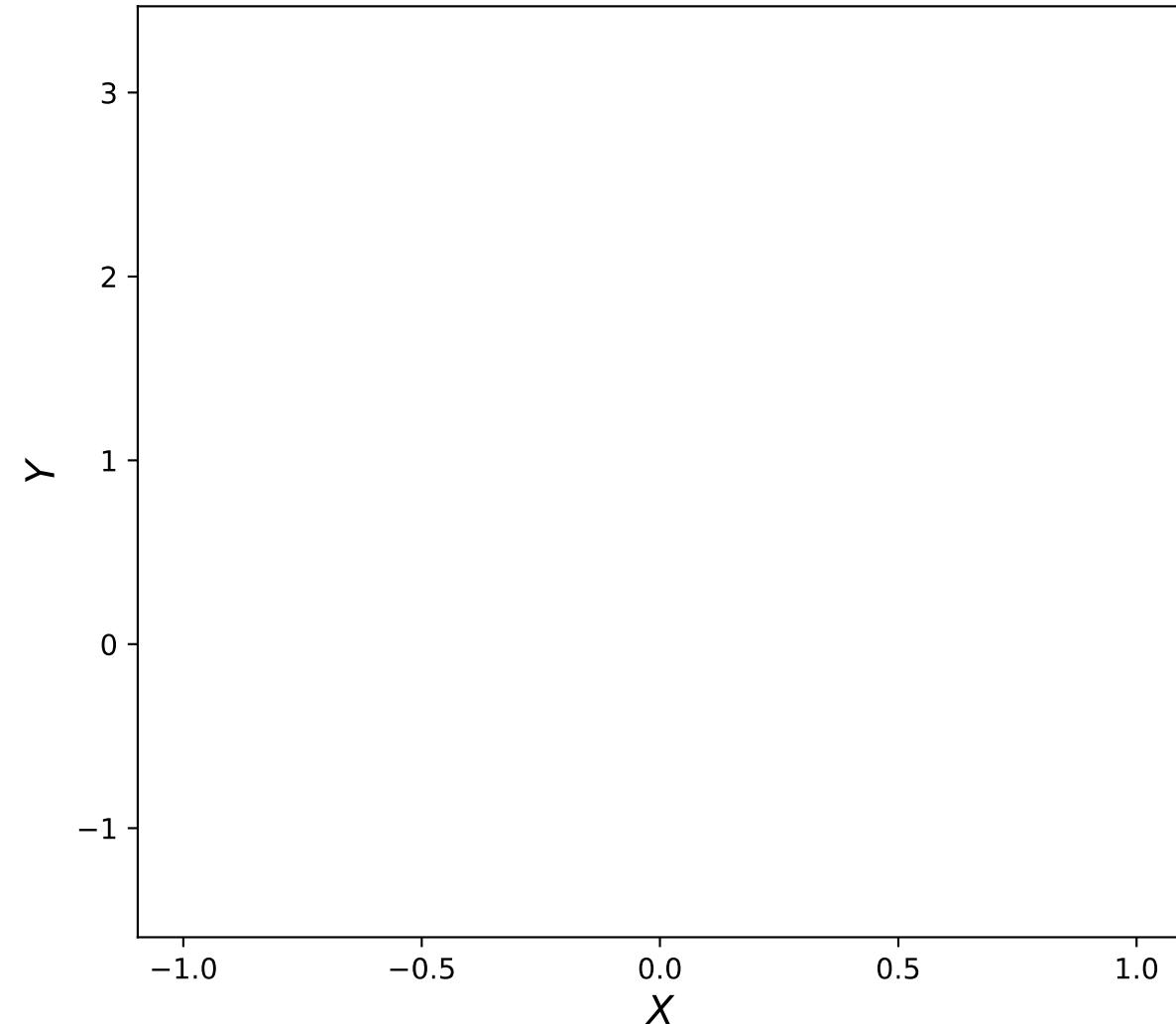
$$\mathbb{E}[Y \mid X = x] = m(x) = \beta_0 + \sum_{k=1}^p \beta_k x_k = x^\top \beta$$

Using our limited dataset, we can build

$$\hat{m}(x) = x^\top \hat{\beta} \sim \mathcal{N}(x^\top \beta, \Sigma_m) \text{ with } \Sigma_m = x^\top \Sigma_\beta x$$

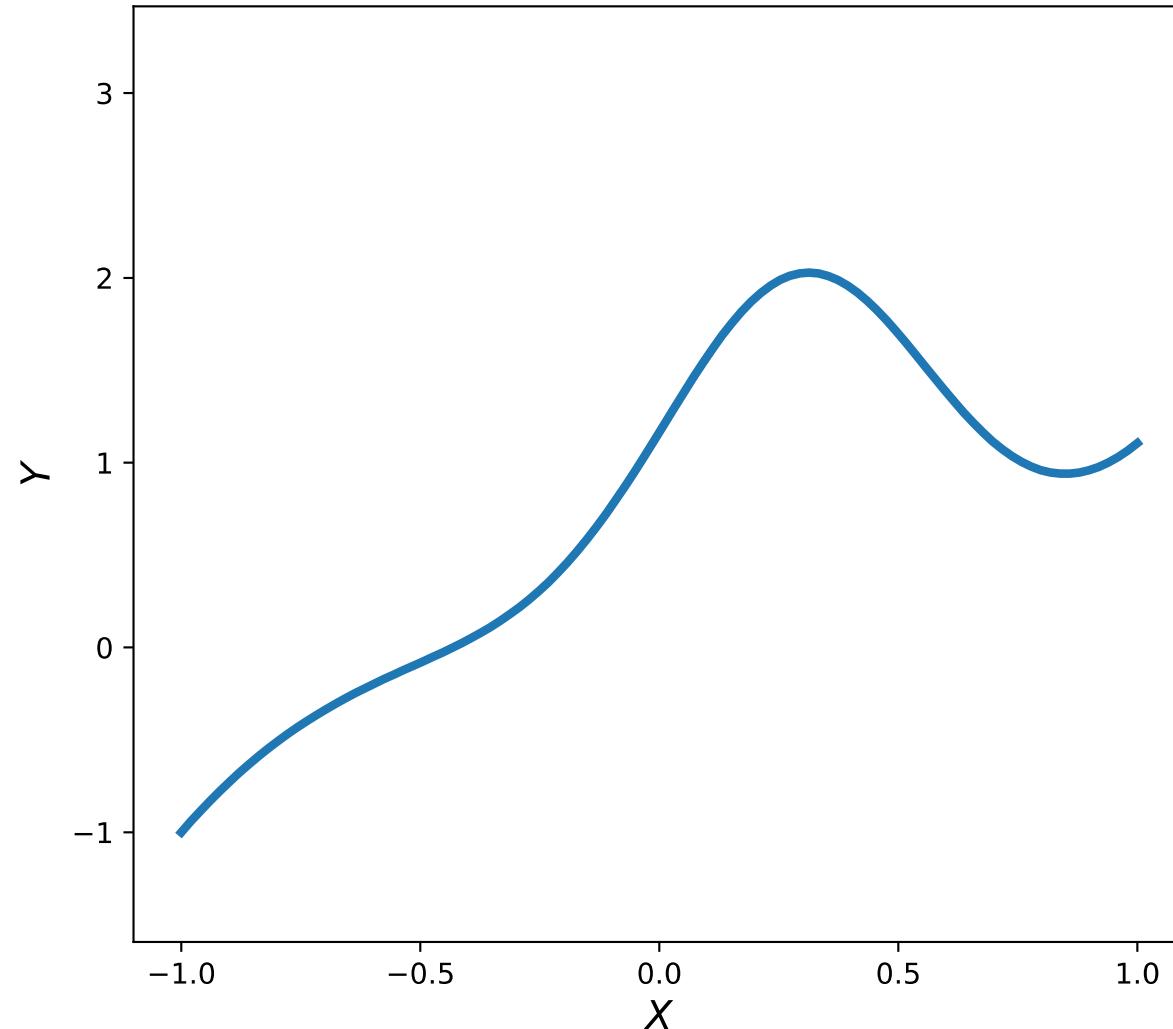
## IN OUR PREVIOUS EPISODES...

But what if we can't assume a linear model? Nor Gaussian?



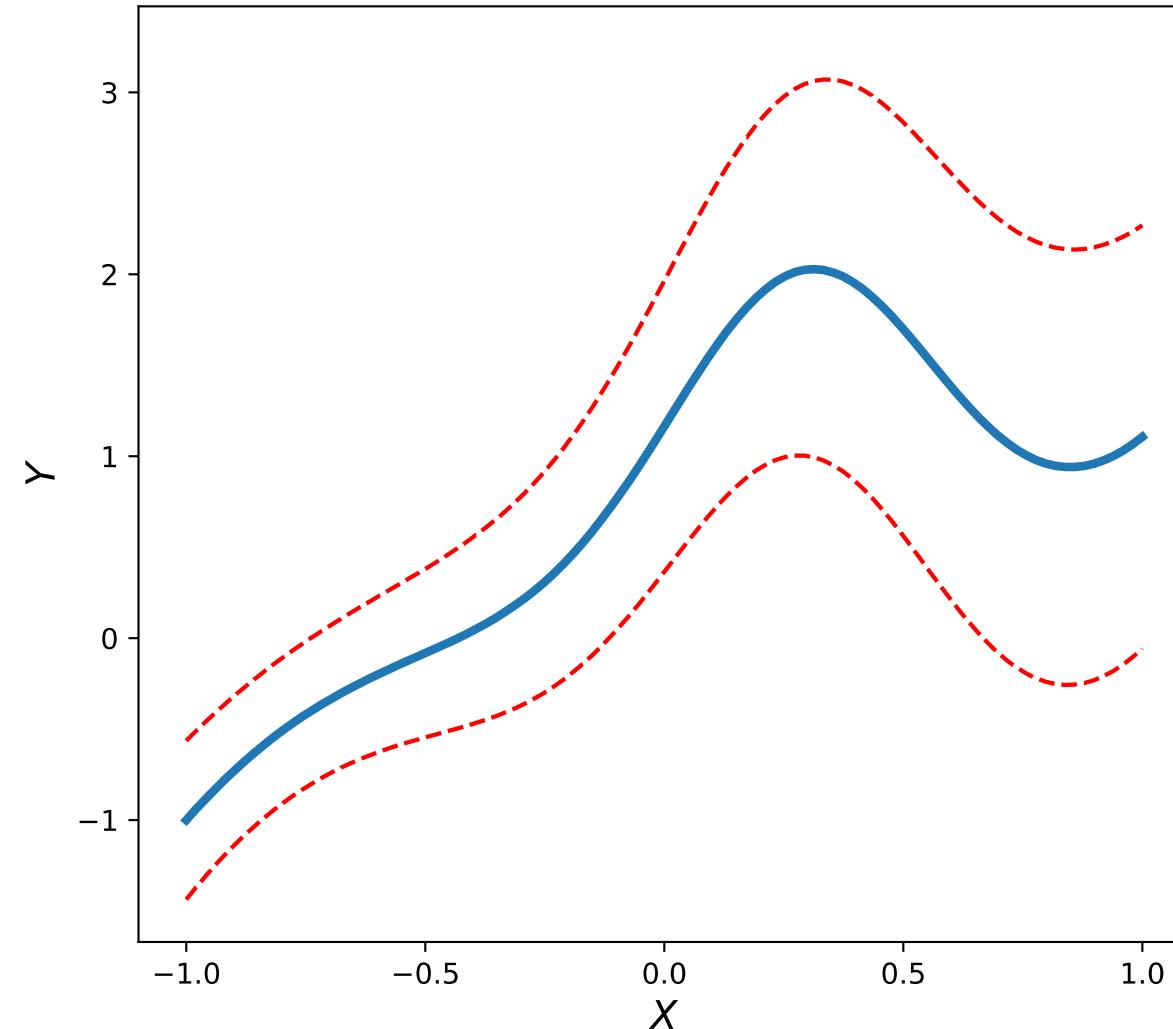
## IN OUR PREVIOUS EPISODES...

But what if we can't assume a linear model? Nor Gaussian?



## IN OUR PREVIOUS EPISODES...

But what if we can't assume a linear model? Nor Gaussian?



# IN OUR PREVIOUS EPISODES...

But what if we can't assume a linear model? Nor Gaussian?

The image shows a composite screenshot of two web pages. The top portion is a red header from arXiv with the logo and navigation. Below it is a white section for 'Computer Science > Machine Learning'. It includes a timestamp, the title 'A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification', and the authors' names. The bottom portion is a GitHub repository page for 'awesome-conformal-prediction'. It shows basic repository stats (main branch, 1 branch, 1 tag), a search bar, and a code switcher. A recent commit by 'valeman' is listed. On the right, there's an 'About' section describing the project as a professionally curated list of resources.

arXiv > cs > arXiv:2107.07511

Computer Science > Machine Learning

[Submitted on 15 Jul 2021 (v1), last revised 7 Dec 2022 (this version, v6)]

# A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification

Anastasios N. Angelopoulos, Stephen Bates

awesome-conformal-prediction Public

Watch 11 Fork 44 Star 526

main 1 Branch 1 Tag

Go to file t Add file Code

valeman Update README.md 329ea75 · yesterday 1,099 Commits

documents Add files via upload 2 years ago

About

A professionally curated list of awesome Conformal Prediction videos, tutorials, books, papers, PhD and MSc theses, articles and open-source libraries.

# IN OUR PREVIOUS EPISODES...

Our current workflow

1

We want to estimate the values of  $Y$  based on predictors  $X_1, \dots, X_p$

2

We are given a set of  $N$  examples of  $y_i$  with corresponding  $x_{i1}, \dots, x_{ip}$

3

We estimate parameters  $\hat{\beta}$  that minimize  $\frac{1}{N} \sum_{i=1}^N (y_i - \beta^T x_i)^2$

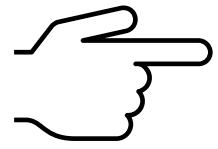
4

We do statistical inference on the values of the  $\hat{\beta}_j$

5

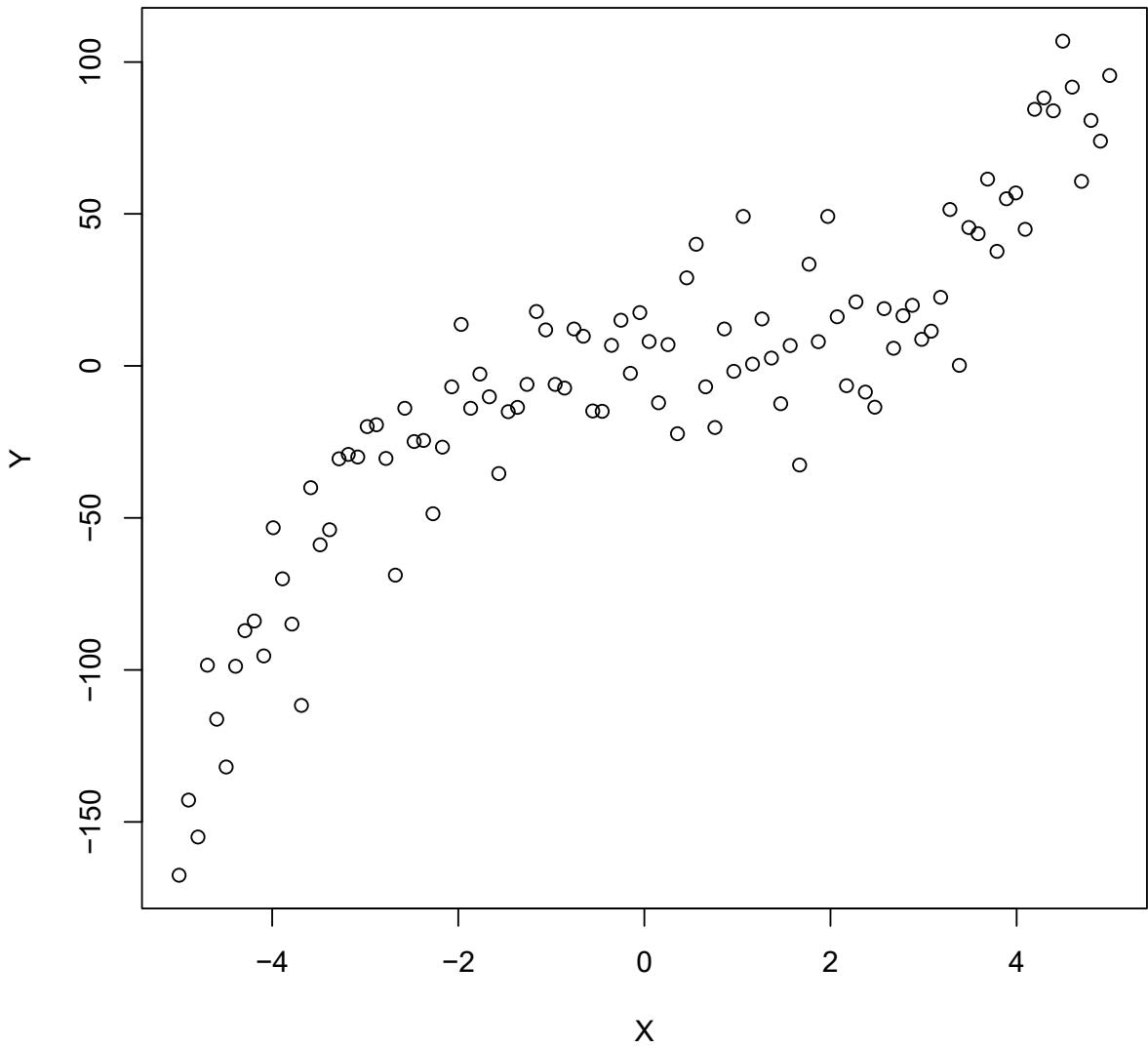
How can we **assess the quality** of a model?

$$x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}$$



- What makes a model **good**?
- Estimating the quality of a model
- Comparing and selecting models

# What makes a model **good**?



Suppose our data is generated as

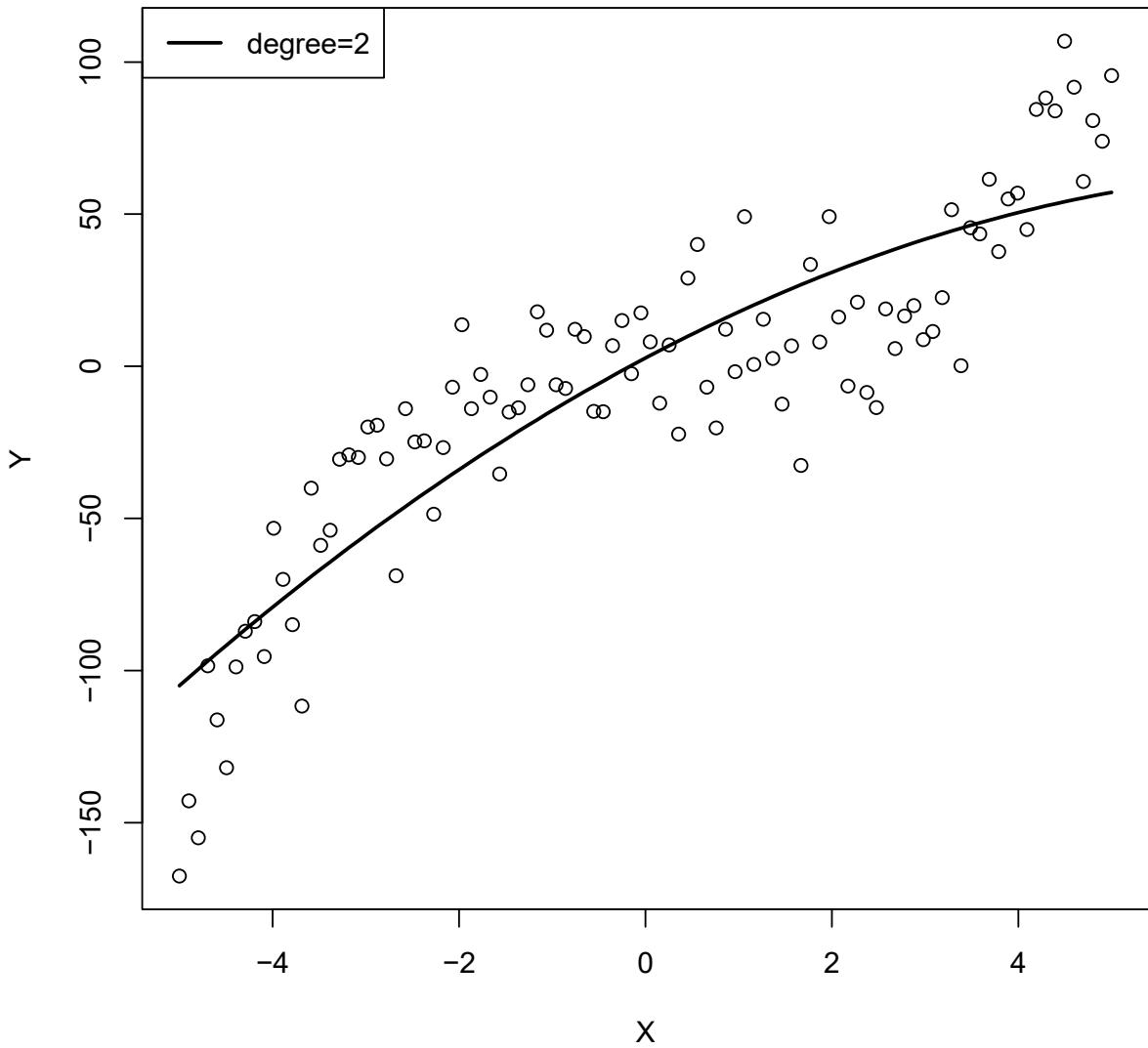
$$Y = \beta_0 + \sum_{i=1}^d \beta_i X^i + \varepsilon$$

we can run multiple linear regression with

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^d \end{bmatrix}$$

**What's the best choice for  $d$  ?**

# What makes a model **good**?



Suppose our data is generated as

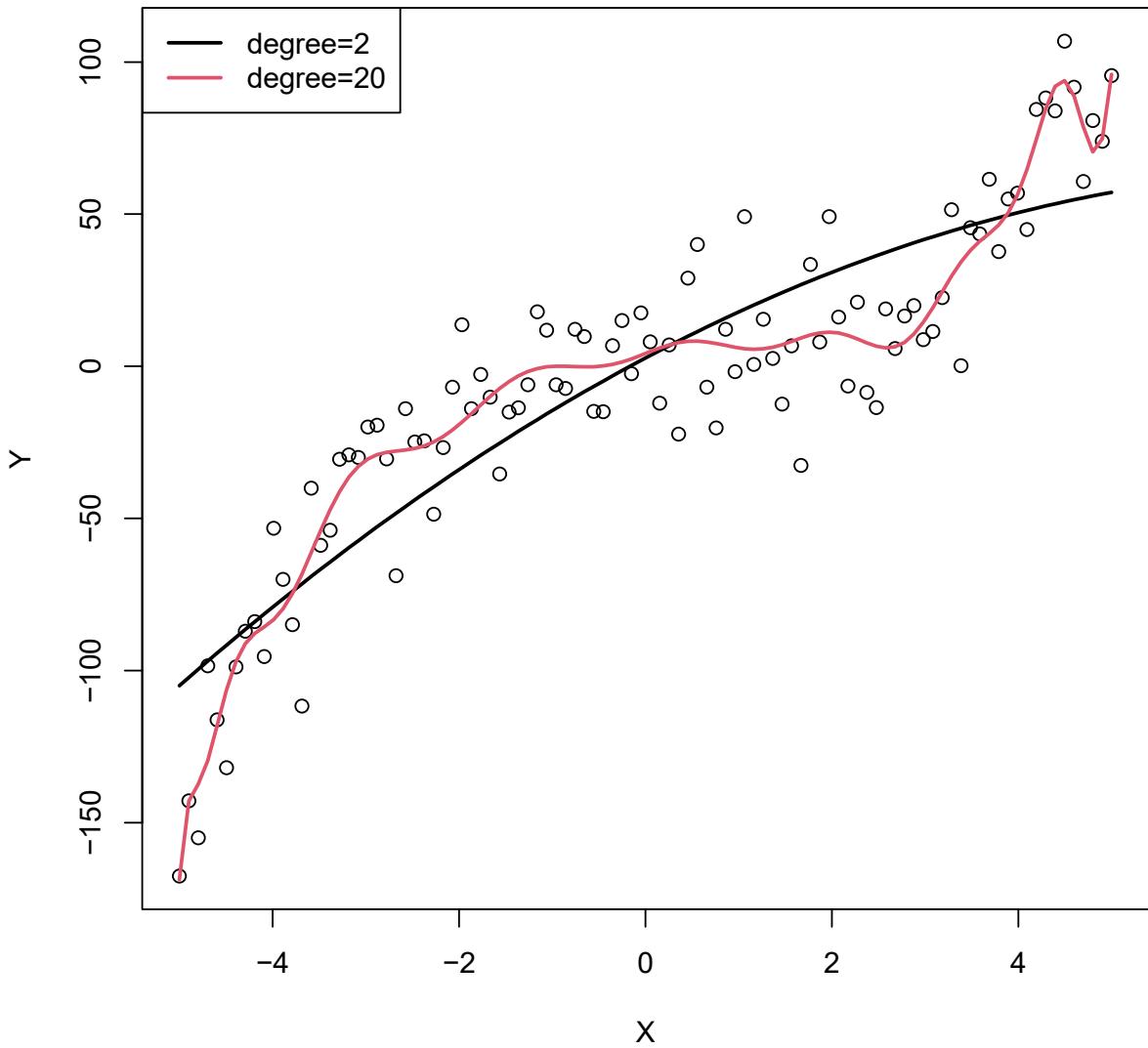
$$Y = \beta_0 + \sum_{i=1}^d \beta_i X^i + \varepsilon$$

we can run multiple linear regression with

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^d \end{bmatrix}$$

**What's the best choice for  $d$  ?**

# What makes a model **good**?



Suppose our data is generated as

$$Y = \beta_0 + \sum_{i=1}^d \beta_i X^i + \varepsilon$$

we can run multiple linear regression with

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^d \end{bmatrix}$$

**What's the best choice for  $d$  ?**

# What makes a model **good**? – The generalization error

Remember that our goal in regression is to minimize the generalization error:

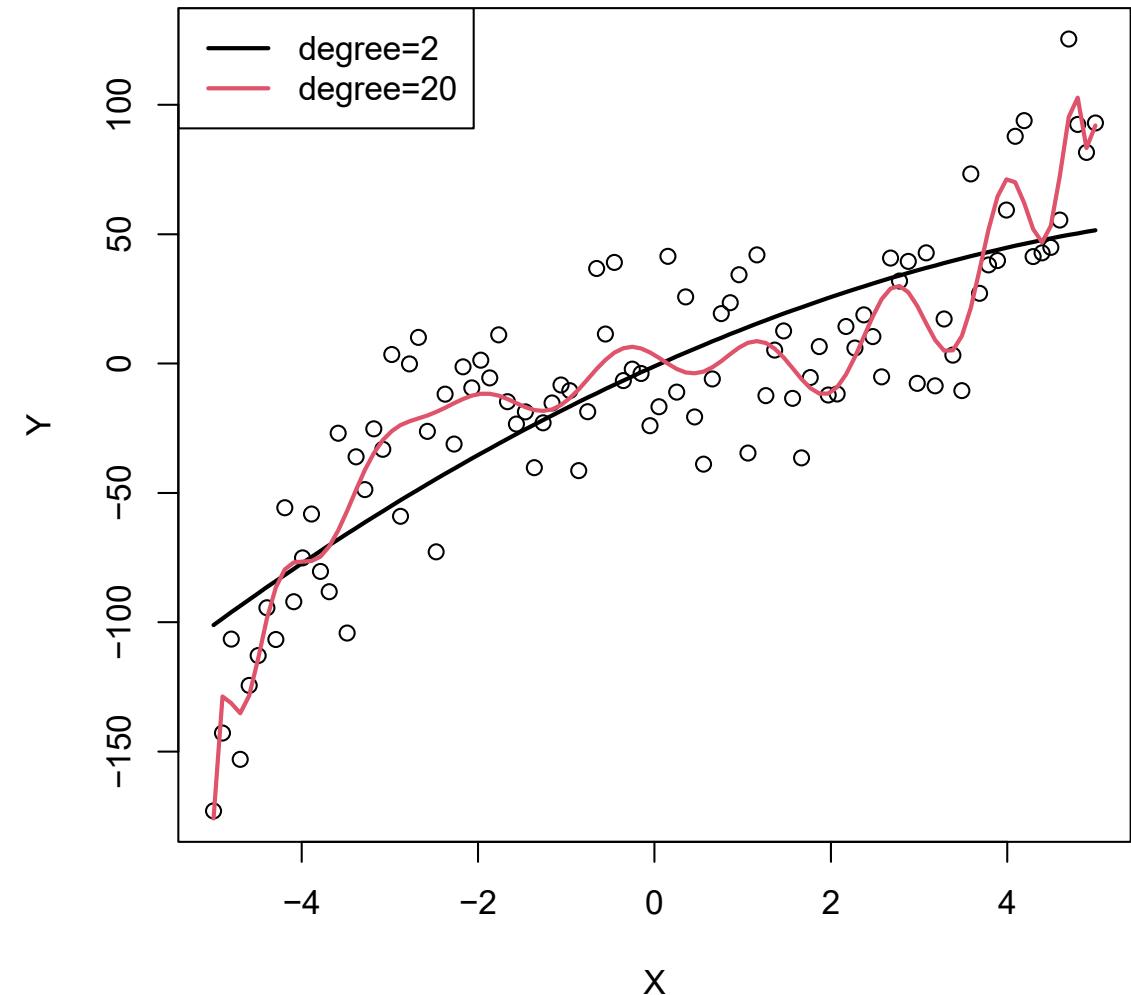
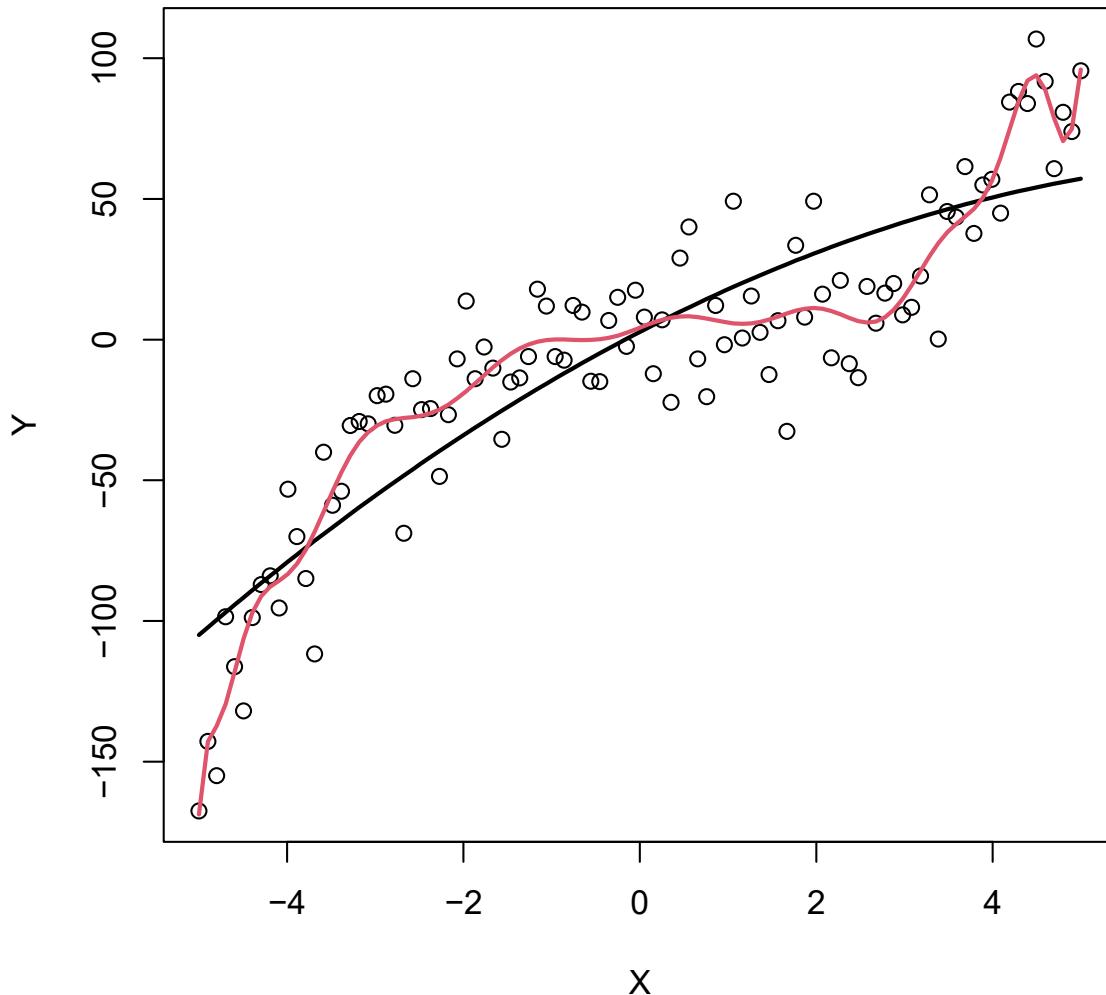
$$\mathcal{L}(r) = \mathbb{E}_{Y,X} \left[ (Y - r(X))^2 \right]$$

Intuitively, we can expect this error to be decomposed into a few informative parts:

- The **irreducible error**: can we ever predict  $Y$  from  $X$  with zero prediction error? Probably not, since we always assume having some observation error  $\varepsilon$  in the data model.
- The **estimation bias**: when estimating the conditional expectation, we always have to choose a family of approximators, which may not be sufficiently flexible.
- The **estimation variance**: for families of approximators that are too flexible, it might happen that for every small change in the observed data points, the estimates change a lot.

# What makes a model **good**?

**Estimation variance.** The same model fit on two slightly different datasets.



What makes a model **good**?



**THE BEST WAY TO  
EXPLAIN OVERFITTING**

# What makes a model **good**? – The bias-variance tradeoff

Remember that the data model is assumed to be  $Y = r(X) + \varepsilon$  with  $\text{Var}(\varepsilon) = \sigma^2$

We use a training dataset  $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^{i=N}$  to estimate  $\hat{r}_{\mathcal{D}}$

For new observation  $x$  we write the bias-variance decomposition conditioned on  $x$  as

$$\begin{aligned}\mathcal{L}(\hat{r}_{\mathcal{D}}(x)) &= \mathbb{E}_{Y,X} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \mid X = x \right] \\ &= \mathbb{E}_{Y,X} \left[ (Y - r(X) + r(X) - \hat{r}_{\mathcal{D}}(X))^2 \mid X = x \right] \\ &= \mathbb{E}_{Y,X} \left[ (Y - r(X))^2 \mid X = x \right] + \mathbb{E}_{Y,X} \left[ (r(X) - \hat{r}_{\mathcal{D}}(X))^2 \mid X = x \right] \\ &= \mathbb{E}_{Y,X} \left[ (Y - r(X))^2 \mid X = x \right] + \left( \mathbb{E}_{Y,X} [r(X) - \hat{r}_{\mathcal{D}}(X) \mid X = x] \right)^2 + \text{Var}(\hat{r}_{\mathcal{D}}(X) \mid X = x)\end{aligned}$$

---

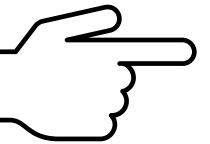
Irreducible error  $\sigma^2$

---

Squared estimation bias

---

Estimation variance

- What makes a model **good**?
- 
- Estimating the quality of a model
  - Comparing and selecting models

# Estimating the quality of a model

How can we estimate the generalization error of an approximator model in practice?

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{Y,X} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \right]$$



We don't  
know the  $p(Y, X)$

# Estimating the quality of a model

How can we estimate the generalization error of an approximator in practice?

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{Y,X} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \right] \approx \frac{1}{M} \sum_{i=1}^M (y_i - \hat{r}_{\mathcal{D}}(x_i))^2 = L(\hat{r}_{\mathcal{D}}, \mathcal{X}) \text{ with } \mathcal{X} = \{(y_i, x_i)\}_{i=1}^{i=M}$$



We don't  
know the  $p(Y, X)$

So we approximate it  
with M data points

Note that for  $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^{i=N}$  we have  $\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{F}} L(r, \mathcal{D})$

We want to minimize one  
quantity but can only estimate  
a proxy of it

# One-slide reminder of the law of large numbers (LLN)

Whenever we want to approximate an expectation, we can use the LLN

$$\mathbb{E}_X[f(X)] = \int f(x)p(x)dx \approx \frac{1}{N} \sum_{n=1}^N f(x_i)$$

with  $N$  **independent** and identically distributed samples  $x_i \sim p(x)$

This result will be useful many times in our class and is ubiquitous in machine learning

**Approximate weighted integral**  **Average of random IID samples**

This is also used in computer graphics, statistical physics, econometry, etc.

# Estimating the quality of a model

How can we estimate the generalization error of an approximator in practice?

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{Y,X} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \right] \approx \frac{1}{M} \sum_{i=1}^M (y_i - \hat{r}_{\mathcal{D}}(x_i))^2 = L(\hat{r}_{\mathcal{D}}, \mathcal{X}) \text{ with } \mathcal{X} = \{(y_i, x_i)\}_{i=1}^M$$

↑  
We don't know the  $p(Y, X)$

So we approximate it with M data points

Note that for  $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^N$  we have  $\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{F}} L(r, \mathcal{D})$

We want to minimize one quantity but can only estimate a proxy of it

**Question:**

Can we say that  $\mathcal{L}(\hat{r}_{\mathcal{D}}) \approx L(\hat{r}_{\mathcal{D}}, \mathcal{D})$  ?

**ABSOLUTELY NOT!**



In fact  $\mathcal{L}(\hat{r}_{\mathcal{D}}) \geq L(\hat{r}_{\mathcal{D}}, \mathcal{D})$  i.e. the true generalization error is larger than the estimated one

# The optimism of the training error

Different ways of seeing the problem:

- Intuitively, since  $\hat{r}_{\mathcal{D}} = \underset{r \in \mathcal{F}}{\operatorname{argmin}} L(r, \mathcal{D})$  then  $L(\hat{r}_{\mathcal{D}}, \mathcal{D})$  will always look small...

# The optimism of the training error

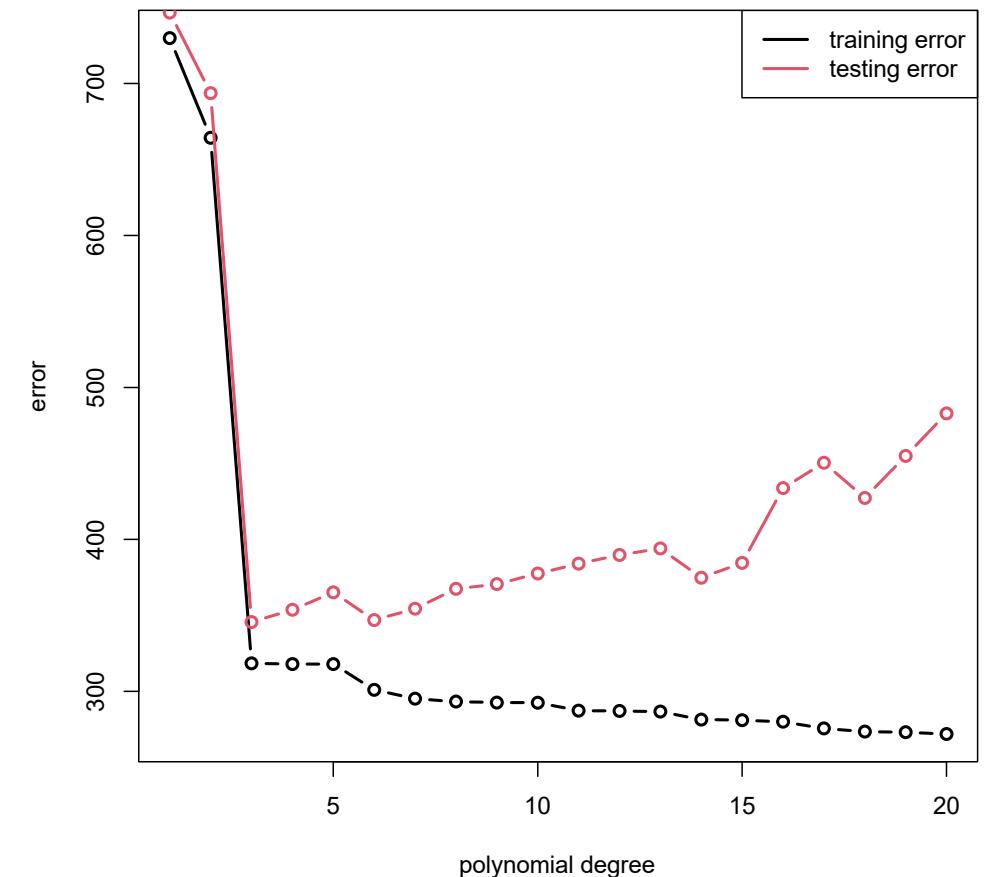
Different ways of seeing the problem:

- Intuitively, since  $\hat{r}_{\mathcal{D}} = \underset{r \in \mathcal{F}}{\operatorname{argmin}} L(r, \mathcal{D})$  then  $L(\hat{r}_{\mathcal{D}}, \mathcal{D})$  will always look small...
- Consider our example with polynomials
  - Increasing the degree improves training error
  - But the testing error grows!

$L(\hat{r}_{\mathcal{D}}, \mathcal{D})$  is the **training** error

$L(\hat{r}_{\mathcal{D}}, \mathcal{X})$  with  $\mathcal{X} \neq \mathcal{D}$  is the **testing** error

Error on unseen data



# The optimism of the training error

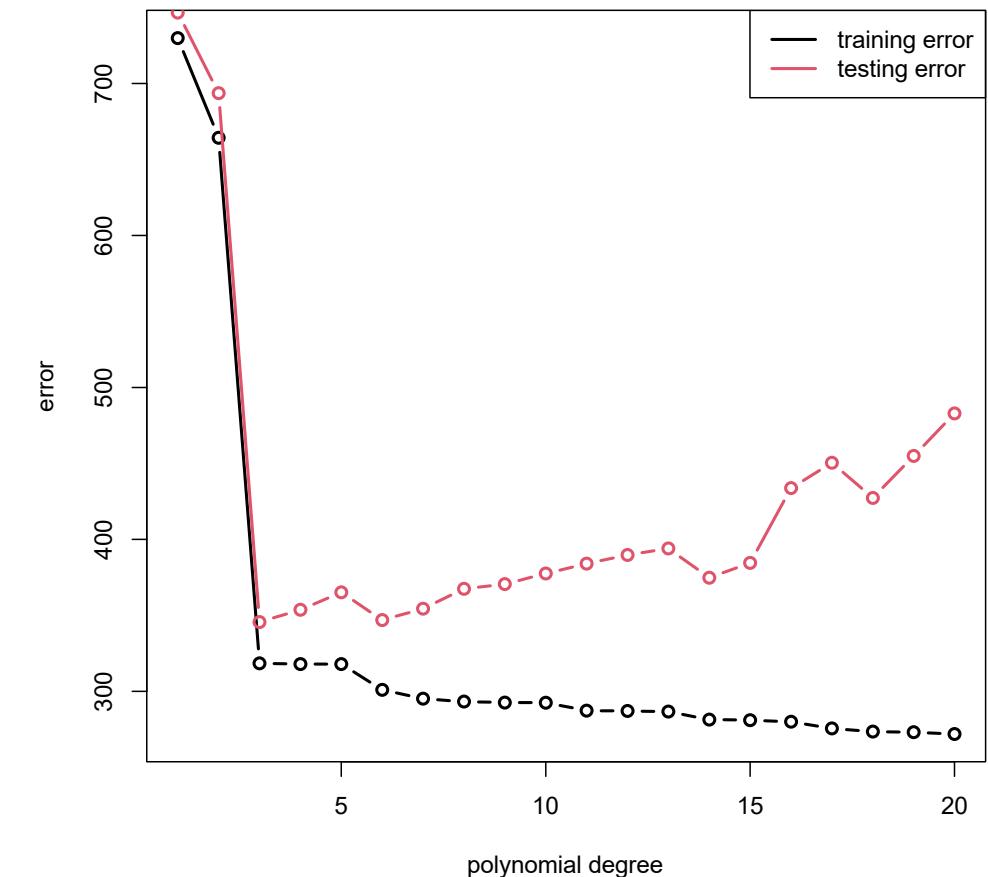
Different ways of seeing the problem:

- Intuitively, since  $\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{F}} L(r, \mathcal{D})$  then  $L(\hat{r}_{\mathcal{D}}, \mathcal{D})$  will always look small...
- Consider our example with polynomials
  - Increasing the degree improves training error
  - But the testing error grows!
- Mathematical illustration on linear regression

$$\mathcal{D} = \left\{ (x_i, y_i) \right\}_{i=1}^{i=N} \quad \mathcal{D}' = \left\{ (x_i, y'_i) \right\}_{i=1}^{i=N} \quad Y = \beta_0 + \sum_{k=1}^p \beta_k X_k + \varepsilon$$

(same predictors but different observations because of noise)

$$\mathbb{E}[L(\hat{r}_{\mathcal{D}}, \mathcal{D}')] = \mathbb{E}[L(\hat{r}_{\mathcal{D}}, \mathcal{D})] + \frac{2}{N} \sigma^2(p+1)$$



# The optimism of the training error

1

We are given a dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

# The optimism of the training error

1

We are given a dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

2

We train regressor on  $\mathcal{D}$  that minimizes

$$\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N (y_i - r(x_i))^2$$

# The optimism of the training error

1

We are given a dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

2

We train regressor on  $\mathcal{D}$  that minimizes

$$\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N (y_i - r(x_i))^2$$

3

We want to estimate the generalization error of  $\hat{r}_{\mathcal{D}}$  which is defined as per

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{(X,Y)} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \right]$$

# The optimism of the training error

1

We are given a dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

3

We want to estimate the generalization error of  $\hat{r}_{\mathcal{D}}$  which is defined as per

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{(X,Y)} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \right]$$

2

We train regressor on  $\mathcal{D}$  that minimizes

$$\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N (y_i - r(x_i))^2$$

4

But we can only obtain an approximation

$$L(\hat{r}_{\mathcal{D}}, \mathcal{X}) = \frac{1}{M} \sum_{i=1}^M (y_i - \hat{r}_{\mathcal{D}}(x_i))^2$$

such that  $\mathcal{X} = \{x_i, y_i\}_{i=1}^M$   
 $(x_i, y_i) \sim p(x, y)$

# The optimism of the training error

1

We are given a dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

3

We want to estimate the generalization error of  $\hat{r}_{\mathcal{D}}$  which is defined as per

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{(X,Y)} \left[ (Y - \hat{r}_{\mathcal{D}}(X))^2 \right]$$

!

Beware the optimism of training error!

$$L(\hat{r}_{\mathcal{D}}, \mathcal{D}) \leq L(\hat{r}_{\mathcal{D}}, \mathcal{X})$$

This was the criterion used during the training

2

We train regressor on  $\mathcal{D}$  that minimizes

$$\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N (y_i - r(x_i))^2$$

4

But we can only obtain an approximation

$$L(\hat{r}_{\mathcal{D}}, \mathcal{X}) = \frac{1}{M} \sum_{i=1}^M (y_i - \hat{r}_{\mathcal{D}}(x_i))^2$$

such that  $\mathcal{X} = \{x_i, y_i\}_{i=1}^M$   
 $(x_i, y_i) \sim p(x, y)$

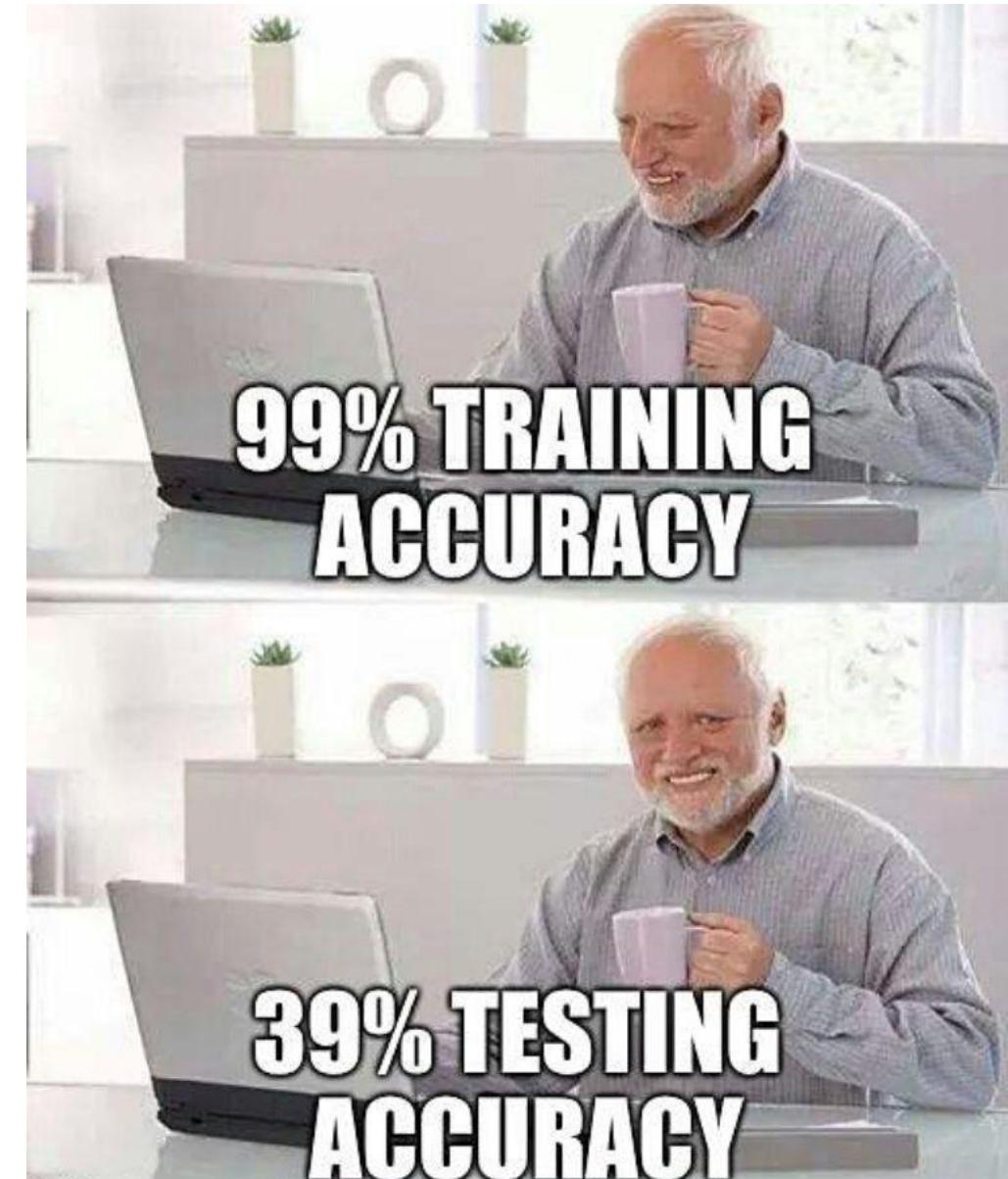
# The optimism of the training error

## Conclusion:

- We can only know if a model is good or not if we **correctly evaluate** its performance
- The objective function that we minimize when training a model (**training error**) is never the same as the one we are actually interested in minimizing (**test error**)

$$\hat{r}_{\mathcal{D}} = \operatorname{argmin}_{r \in \mathcal{F}} \frac{1}{M} \sum_{i=1}^M \left( y_i - r(x_i) \right)^2$$

$$\mathcal{L}(\hat{r}_{\mathcal{D}}) = \mathbb{E}_{(X,Y)} \left[ \left( Y - \hat{r}_{\mathcal{D}}(X) \right)^2 \right]$$



# Cross-validation

We are given a dataset with  $N$  datapoints  $\mathcal{D} = \left\{ (y_i, x_i) \right\}_{i=1}^{i=N}$

How can we estimate the **generalization error** of a model we train on this dataset?

**Remember:** We should evaluate the model on samples that were not used for training

# Cross-validation

We are given a dataset with  $N$  datapoints  $\mathcal{D} = \left\{ (y_i, x_i) \right\}_{i=1}^{i=N}$

How can we estimate the **generalization error** of a model we train on this dataset?

**Remember:** We should evaluate the model on samples that were not used for training

- **Strategy 1 : Single hold-out test point**

We fit a model on the first  $N-1$  training samples, calling it  $\hat{r}^{(-N)}$

Treat the last sample  $(X_N, Y_N)$  as a test sample and estimate generalization error as

$$\mathcal{L}(\hat{r}) \approx \left( y_N - \hat{r}^{(-N)}(x_N) \right)^2$$

The estimator is easy to calculate but we can expect a rather large variance.

# Cross-validation

We are given a dataset with  $N$  datapoints  $\mathcal{D} = \left\{ (y_i, x_i) \right\}_{i=1}^{i=N}$

How can we estimate the **generalization error** of a model we train on this dataset?

**Remember:** We should evaluate the model on samples that were not used for training

- **Strategy 2 : Leave one out cross-validation (LOOCV)**

We fit  $N$  models on  $N-1$  training samples, holding out  $x_i$  at each time to get  $\hat{r}^{(-i)}$

The test error is approximate as

$$\mathcal{L}(\hat{r}) \approx \frac{1}{N} \sum_{i=1}^N \left( y_i - \hat{r}^{(-i)}(x_i) \right)^2$$

The variance decreases, but the computational burden is much higher

# Cross-validation

We are given a dataset with  $N$  datapoints  $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^{i=N}$

How can we estimate the **generalization error** of a model we train on this dataset?

**Remember:** We should evaluate the model on samples that were not used for training

- **Strategy 3 : K-Fold cross-validation**

Split the training dataset randomly into  $K$  folds so to have  $\mathcal{D}_1 \cup \dots \cup \mathcal{D}_K = \mathcal{D}$

For  $k=1,\dots,K$  fit a model  $\hat{r}^{(-k)}$  on a training set except that excludes  $\mathcal{D}_k$

$$\mathcal{L}(\hat{r}) \approx \frac{1}{K} \sum_{k=1}^K \left( \frac{1}{N_k} \sum_{i \in \mathcal{D}_k} (y_i - \hat{r}^{(-k)}(x_i))^2 \right)$$

↑

Take average over folds

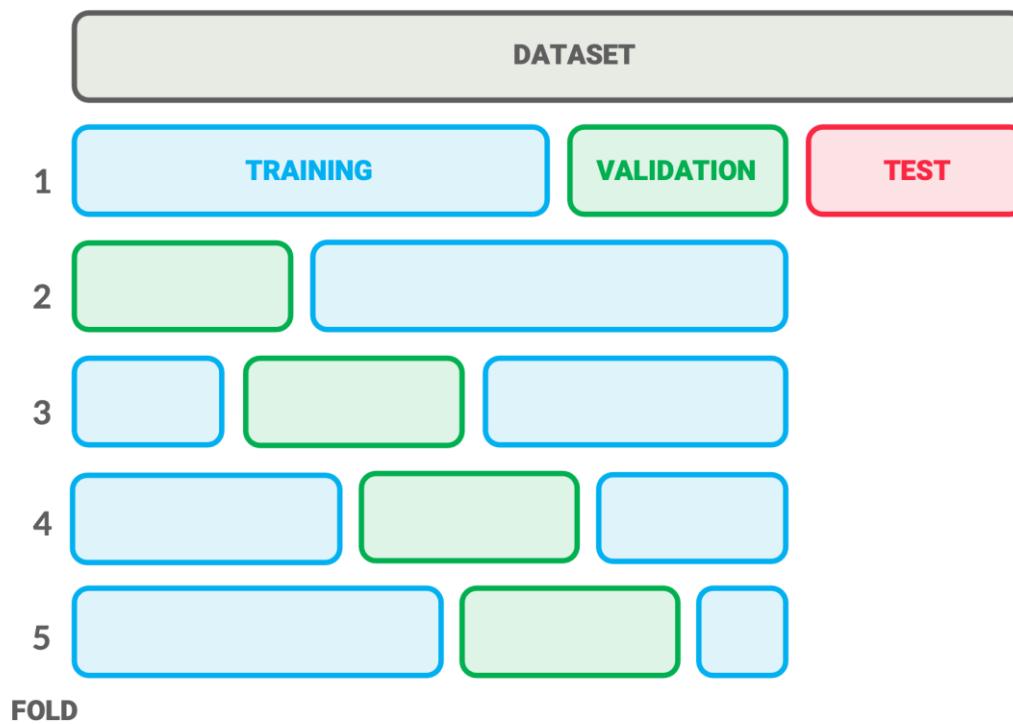
Estimate test error for each fold

# Cross-validation

## Strategy 3 : K-Fold cross-validation

Note that the errors calculated in each fold are not IID random variables, since the models share some of their data points.

↳ **Question:** What does this imply?



Using 

```
from sklearn.model_selection import KFold
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
kf = KFold(n_splits=5)
dataset = fetch_california_housing()
lm = LinearRegression()
X = dataset.data
y = dataset.target
scores = []
for idx_train, idx_test in kf.split(X):
    X_train, y_train = X[idx_train], y[idx_train]
    X_test, y_test = X[idx_test], y[idx_test]
    lm.fit(X_train, y_train)
    scores.append(lm.score(X_test, y_test))
print(np.mean(scores))
```

See the documentation for more details

# Cross-validation

But there are **several** other strategies!

<a href="#"><u>GroupKFold</u></a>	K-fold iterator variant with non-overlapping groups.
<a href="#"><u>GroupShuffleSplit</u></a>	Shuffle-Group(s)-Out cross-validation iterator.
<a href="#"><u>KFold</u></a>	K-Fold cross-validator.

<a href="#"><u>ShuffleSplit</u></a>	Random permutation cross-validator.
<a href="#"><u>StratifiedGroupKFold</u></a>	Stratified K-Fold iterator variant with non-overlapping groups.
<a href="#"><u>StratifiedKFold</u></a>	Stratified K-Fold cross-validator.
<a href="#"><u>StratifiedShuffleSplit</u></a>	Stratified ShuffleSplit cross-validator.
<a href="#"><u>TimeSeriesSplit</u></a>	Time Series cross-validator.

and more...

# Cross-validation

(1) Example with the **categorical variables** from last week

```
from sklearn.model_selection import KFold, ShuffleSplit
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
cv = KFold(n_splits=4)
results_cv = cross_validate(
    regressor, X, y, cv=cv, scoring='neg_mean_squared_error')
print(f'MSE: {-results_cv["test_score"].mean():.2f}')

regressor = LinearRegression()
cv = ShuffleSplit(n_splits=4)
results_cv = cross_validate(
    regressor, X, y, cv=cv, scoring='neg_mean_squared_error')
print(f'MSE: {-results_cv["test_score"].mean():.2f}')
```

17.50

4.60

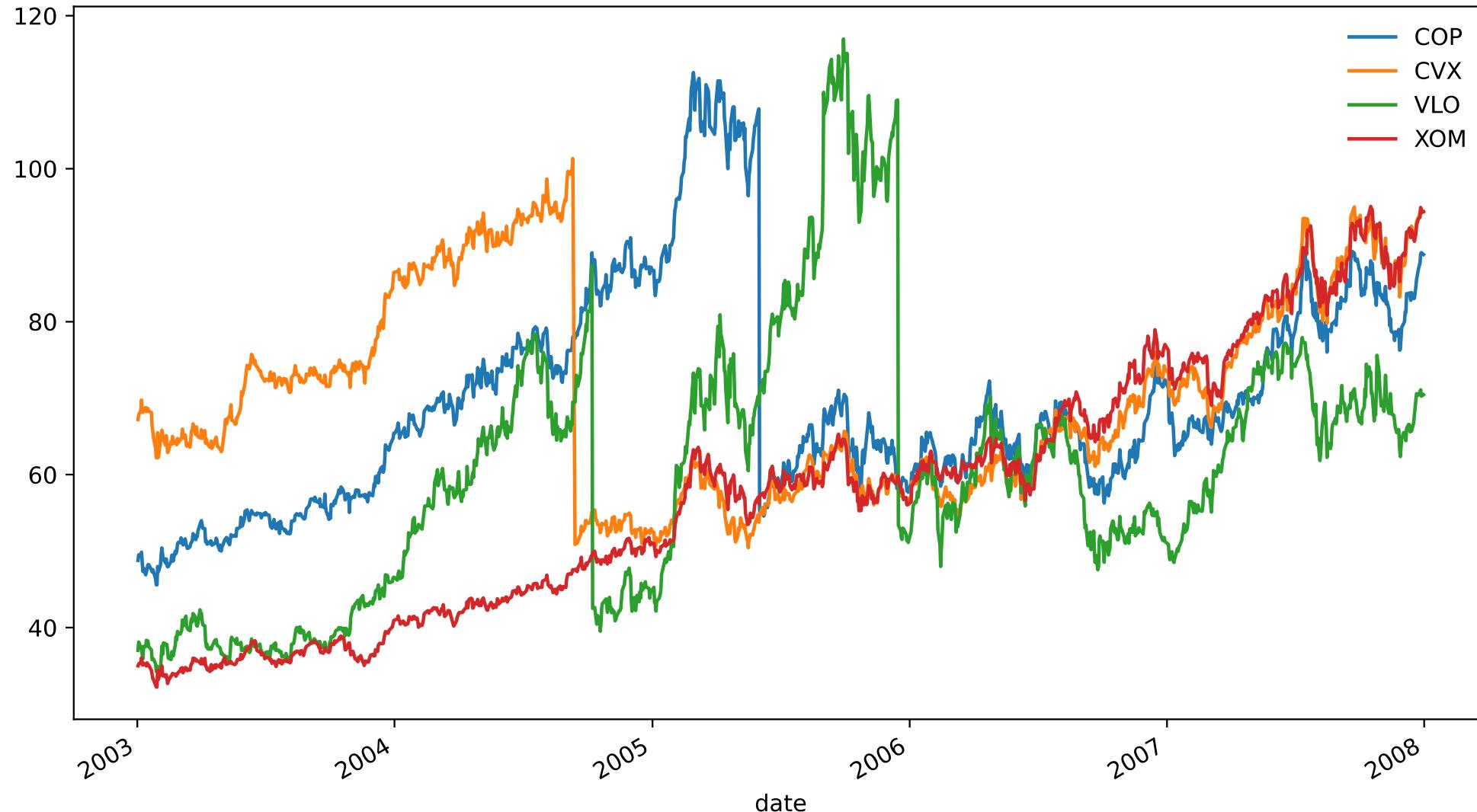
What is going on? 😅

In [85]: df  
Out [85]:

	life	rpm	brand
0	18.73	610	A
1	14.52	950	A
2	17.43	720	A
3	14.54	840	A
4	13.44	980	A
5	24.39	530	A
6	13.34	680	A
7	22.71	540	A
8	12.68	890	A
9	19.32	730	A
10	30.16	670	B
11	27.09	770	B
12	25.40	880	B
13	26.05	1000	B
14	33.49	760	B
15	35.62	590	B
16	26.07	910	B
17	36.78	650	B
18	34.95	810	B
19	43.67	500	B

# Cross-validation

(2) Example with **financial** time series: predict CVX's quotes based on other quotes



# Cross-validation

(2) Example with **financial** time series: predict CVX's quotes based on other quotes

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import ShuffleSplit, cross_validate
X = quotes.drop(columns=["CVX"])
y = quotes["CVX"]
regressor = GradientBoostingRegressor()
cv = ShuffleSplit(n_splits=10)
results_cv = cross_validate(regressor, X, y, cv=cv)
print(f'Mean R2: {results_cv["test_score"].mean():.2f}')
```

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

0.95



This looks like an almost perfect prediction!  Does it sound right to you?

# Cross-validation

(2) Example with **financial** time series: predict CVX's quotes based on other quotes

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import ShuffleSplit, cross_validate
X = quotes.drop(columns=["CVX"])
y = quotes["CVX"]
regressor = GradientBoostingRegressor()
cv = ShuffleSplit(n_splits=10)
results_cv = cross_validate(regressor, X, y, cv=cv)
print(f'Mean R2: {results_cv["test_score"].mean():.2f}')
```

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

0.95



This looks like an almost perfect prediction! 🤖 Does it sound right to you?

```
from sklearn.model_selection import TimeSeriesSplit
cv = TimeSeriesSplit(n_splits=10)
results_cv = cross_validate(regressor, X, y, cv=cv)
print(f'Mean R2: {results_cv["test_score"].mean():.2f}')
```

-3.10

Disappointing, but closer to reality...

# Cross-validation

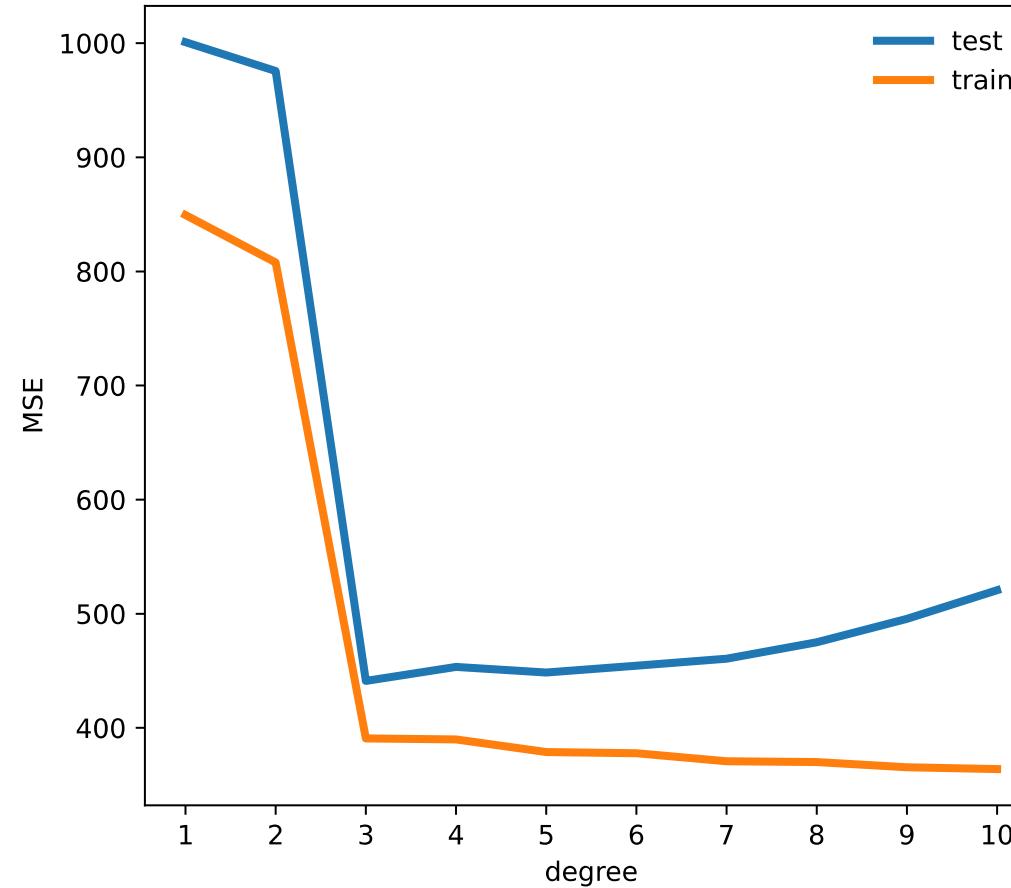
## (3) Choosing the best order for the polynomial approximating a dataset

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from sklearn.preprocessing import PolynomialFeatures
4 from sklearn.pipeline import make_pipeline
5 from sklearn.linear_model import LinearRegression
6 from sklearn.model_selection import GridSearchCV, ShuffleSplit
7
8 np.random.seed(6)
9
10 # generate dataset
11 N = 100
12 x = np.linspace(-5, +5, N)
13 y = x - x**2 + x**3 + 20 * np.random.randn(N)
14 X = x.reshape(-1, 1)
15
16 # instantiate pipeline
17 poly = PolynomialFeatures(include_bias=False)
18 lr = LinearRegression()
19 pipe = make_pipeline(poly, lr)
20
21 # instantiate grid search cv
22 degrees_array = np.arange(1, 10+1)
23 parameters = {'polynomialfeatures_degree':degrees_array}
24 cv = ShuffleSplit(n_splits=10)
25 est = GridSearchCV(estimator=pipe, param_grid=parameters, cv=cv,
26 | | | | | scoring='neg_mean_squared_error', return_train_score=True)
27
28 # get cv results
29 est.fit(X, y)
30 cv_test_scores = -est.cv_results_['mean_test_score']
31 cv_train_scores = -est.cv_results_['mean_train_score']
32
33 # plot results
34 fig, ax = plt.subplots(figsize=(6, 5.4))
35 ax.plot(degrees_array, cv_test_scores, c='C0', lw=3, label='test')
36 ax.plot(degrees_array, cv_train_scores, c='C1', lw=3, label='train')
37 ax.set_xlabel('degree')
38 ax.set_ylabel('MSE')
39 ax.set_xticks(degrees_array)
40 fig.savefig('CM3_cv_figure.pdf', format='pdf')
```

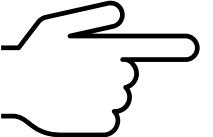
We're using several scikit-learn important concepts : transformers, pipelines, grid search

# Cross-validation

## (3) Choosing the best order for the polynomial approximating a dataset



We're using several scikit-learn important concepts : transformers, pipelines, grid search

- What makes a model **good**?
  - Estimating the quality of a model
  - Comparing and selecting models
- 

# Comparing and selecting models

Suppose we are given a dataset with  $p$  predictors.

We want to estimate a linear model with only a subset of them.

There are mainly **three strategies** for doing this properly:

- **Shrinkage** fits a model with all  $p$  predictors but using a modified loss function that drives some parameters to zero automatically. (Razan will talk about this)
- We can use **dimensionality reduction** techniques to project the  $p$  predictors to a lower-dimensional subspace. (We will see how to do this in CM4 and TP2)
- In **subset selection** we identify a subset of the predictors that seems the most adequate and then fit a model with them.

# Comparing and selecting models

When  $p$  is large, testing all  $2^p$  possible models can be very time consuming...

So we prefer to proceed greedily with e.g. **forward stepwise selection**

**(1)** Let  $\mathcal{M}_0$  denote a model with no predictors (i.e. just the intercept)

**(2)** For  $k = 0, \dots, p - 1$

**(a)** Consider all  $p - k$  models that augment the predictors in  $\mathcal{M}_k$  by one extra predictor

**(b)** Choose the **best** among these  $p - k$  models and call it  $\mathcal{M}_{k+1}$

**(3)** Select single **best** model among  $\mathcal{M}_0, \dots, \mathcal{M}_p$  using cross-validated prediction error

**Example:** Consider the mtcars dataset – we want to predict **mpg**

Description:

The data was extracted from the 1974 *\_Motor Trend\_* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models).

Format:

A data frame with 32 observations on 11 variables.

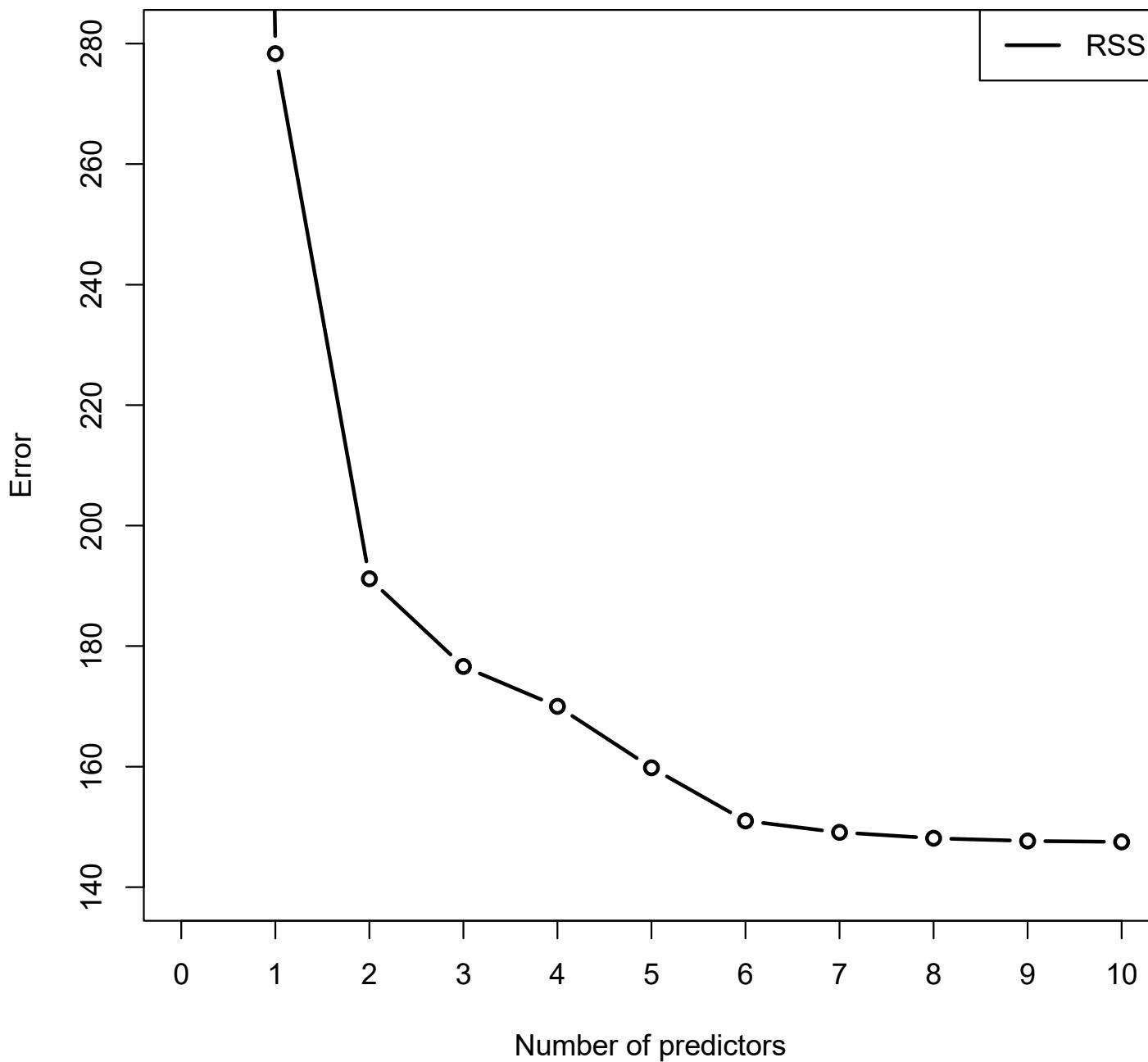
```
[, 1] mpg Miles/(US) gallon
 [, 2] cyl Number of cylinders
 [, 3] disp Displacement (cu.in.)
 [, 4] hp Gross horsepower
 [, 5] drat Rear axle ratio
 [, 6] wt Weight (lb/1000)
 [, 7] qsec 1/4 mile time
 [, 8] vs V/S
 [, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors
```

## OLS Regression Results

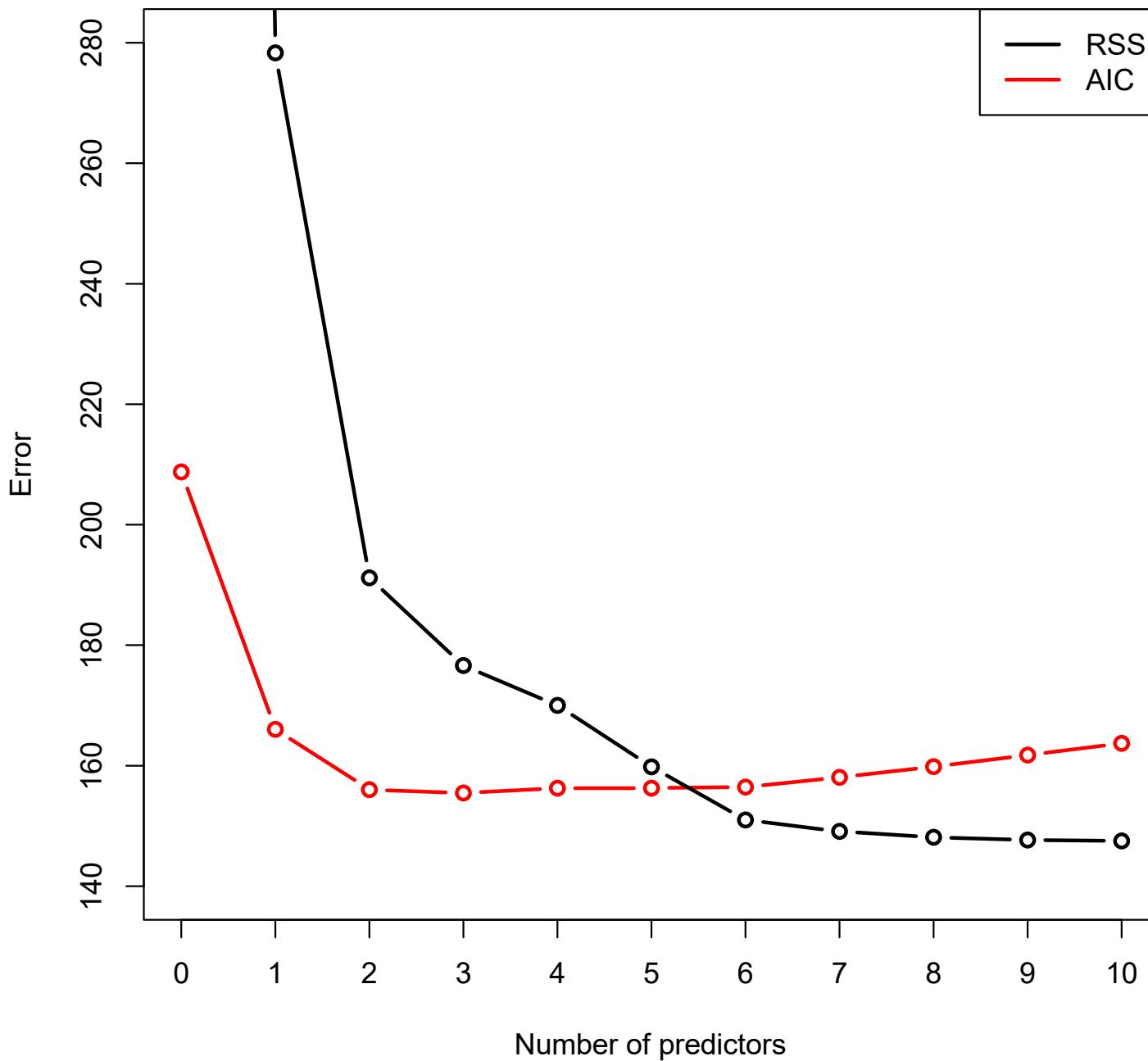
Dep. Variable:	mpg	R-squared:	0.869
Model:	OLS	Adj. R-squared:	0.807
Method:	Least Squares	F-statistic:	13.93
Date:	Fri, 27 Dec 2024	Prob (F-statistic):	3.79e-07
Time:	15:09:11	Log-Likelihood:	-69.855
No. Observations:	32	AIC:	161.7
Df Residuals:	21	BIC:	177.8
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
cyl	-0.1114	1.045	-0.107	0.916	-2.285	2.062
disp	0.0133	0.018	0.747	0.463	-0.024	0.050
hp	-0.0215	0.022	-0.987	0.335	-0.067	0.024
drat	0.7871	1.635	0.481	0.635	-2.614	4.188
wt	-3.7153	1.894	-1.961	0.063	-7.655	0.224
qsec	0.8210	0.731	1.123	0.274	-0.699	2.341
vs	0.3178	2.105	0.151	0.881	-4.059	4.694
am	2.5202	2.057	1.225	0.234	-1.757	6.797
gear	0.6554	1.493	0.439	0.665	-2.450	3.761
carb	-0.1994	0.829	-0.241	0.812	-1.923	1.524
intercept	12.3034	18.718	0.657	0.518	-26.623	51.229

## Forward stepwise selection



## Forward stepwise selection



## Forward stepwise selection

