

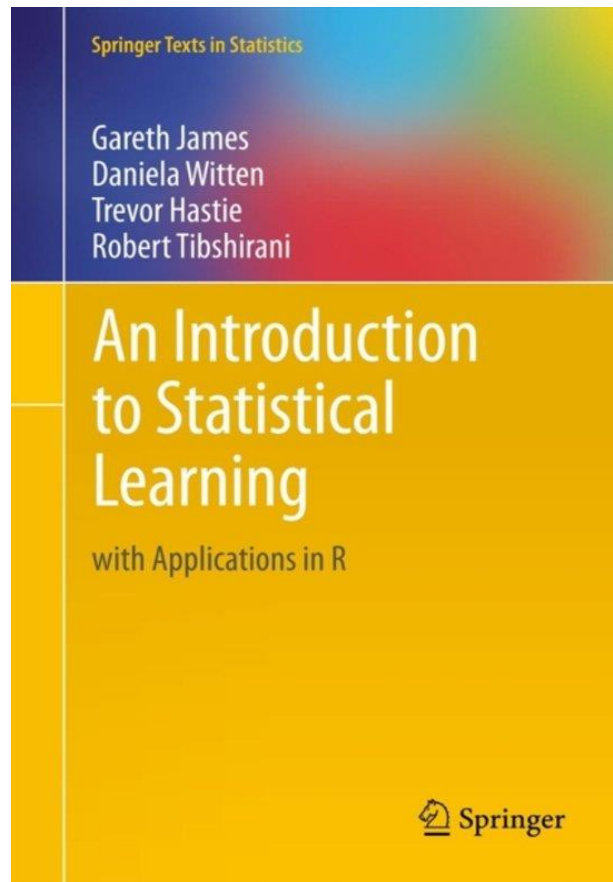


# Regression

from the data science perspective

Based on the lecture Supervised learning - regression part I, from the course Data analysis and visualization

Emmeke Aarts &  
Daniel Oberski



Date	Topic	Presented by
8 november	Statistical learning: an introduction	Dr. Emmeke Aarts
29 november	Regression from the data science perspective	<del>Dr. Dave Hessen</del> Dr. Emmeke Aarts
6 december	Classification	Dr. Gerko Vink
10 januari	Resampling Methods	Dr. Gerko Vink
7 februari	Regularization	Dr. Maarten Cruijf
7 maart	Moving beyond linearity	Dr. Maarten Cruijf
4 april	Tree-Based models	Dr. Emmeke Aarts
9 mei	Support vector Machines	Dr. Daniel Oberski
6 juni	Unsupervised learning	Prof. Dr. Peter van der Heijden

# Last time

---

- What is statistical learning
- Accuracy versus interpretability
- Supervised versus unsupervised learning
- Regression versus classification
- Model accuracy & bias-variance trade off
- Potential benefits for social scientist
- Software

# Today

---

- Introduction
- Overview of regression
- Measuring model accuracy
- Conclusion

Based on the lecture Supervised learning - regression part I, from the course Data analysis and visualization by Daniel Oberski, Nikolaj Tollenaar and Peter van der Heijden

# Important concepts today

---

- Prediction function
- k-nearest neighbors (KNN)
- Metrics for model evaluation
- Bias and variance (tradeoff)
- Training-validation-test set paradigm (or “Train/dev/test”)

# Regression

---

$$y = f(x) + \epsilon$$

There are usually a bunch of  $x$ 's. We keep notation simple by saying  $x$  might be a vector of  $p$  predictors.

# Regression

---

$$y = f(x) + \epsilon$$

$y$ : Observed outcome;

$x$ : Observed predictor(s);

$f(x)$ : Prediction function, to be estimated;

$\epsilon$ : Unobserved residuals, just defined as the “irreducible error”,  $\epsilon = y - f(x)$

The higher the variance of the irreducible error,  
 $\text{variance}(\epsilon) = \sigma^2$ , the less we can explain.

# Different goals of regression

---

## **Prediction:**

- Given  $x$ , work out  $f(x)$ .

## **Inference:**

- “Is  $x$  related to  $y$ ?”
- “How is  $x$  related to  $y$ ?”
- “How precise are parameters of  $f(x)$  estimated from the data?”

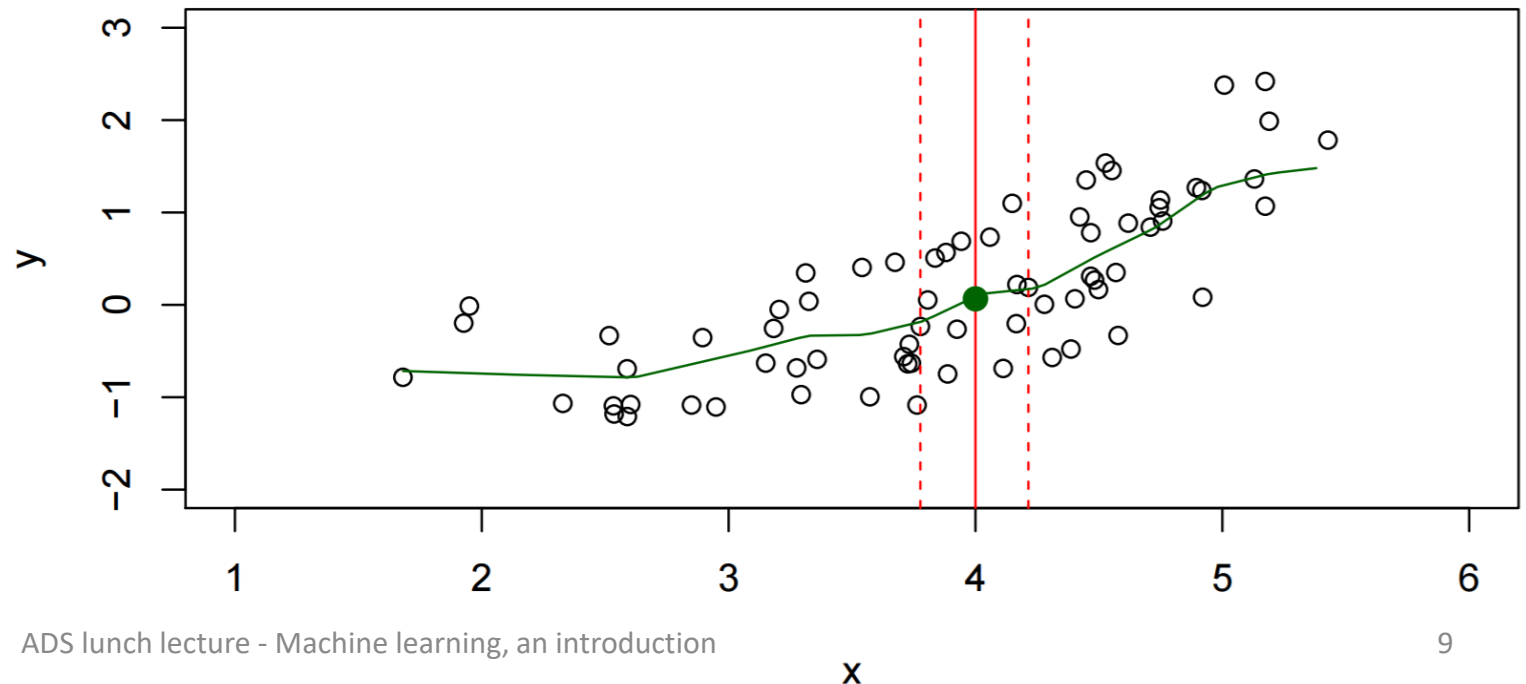


# Estimating $f(y)$ with k-nearest neighbors

- Typically we have no data points  $x = 4$  exactly.
- Instead, take a “neighborhood” of points around 4 and predict its average:

(From James et al.)

$$\hat{f}(x) = n^{-1} \sum_{i=1}^n (y_i | x_i \in \text{neighborhood}(x))$$



# Why not kNN

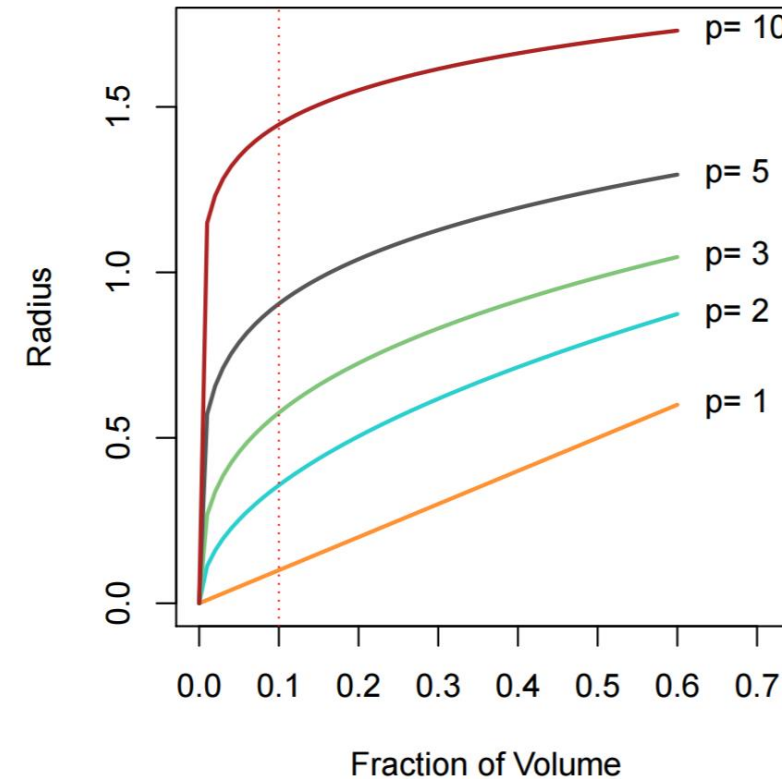
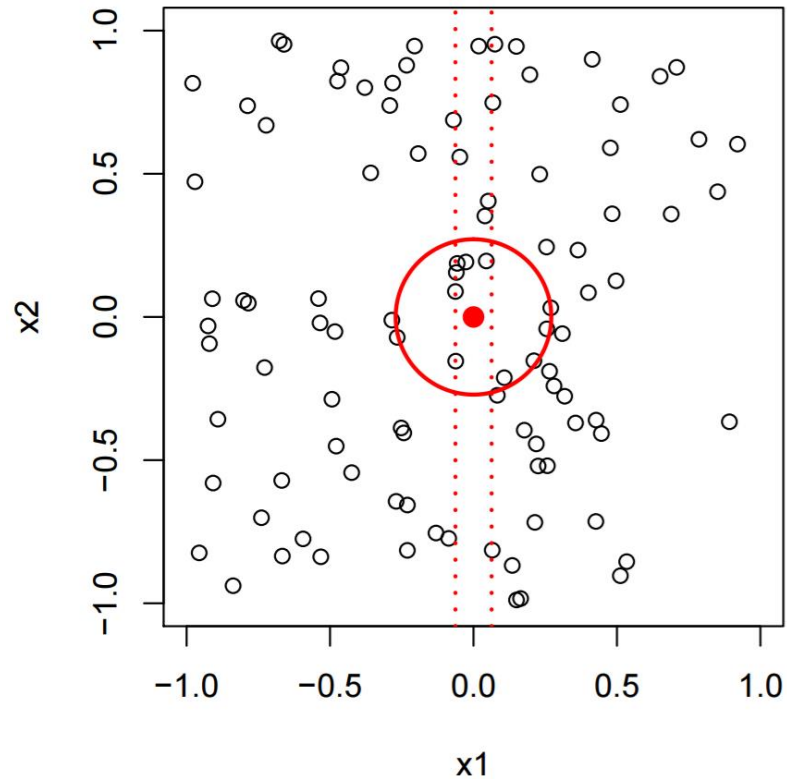
---

- kNN is intuitive and can work well with not too many predictors;
- When there are many (say, 5 or more) predictors, kNN breaks down:
  - The *closest* points on tens of predictors *simultaneously* may actually be far away.
  - “Curse of dimensionality”

# Why kNN does not work with many predictors


10% Neighborhood

(From James et al.)

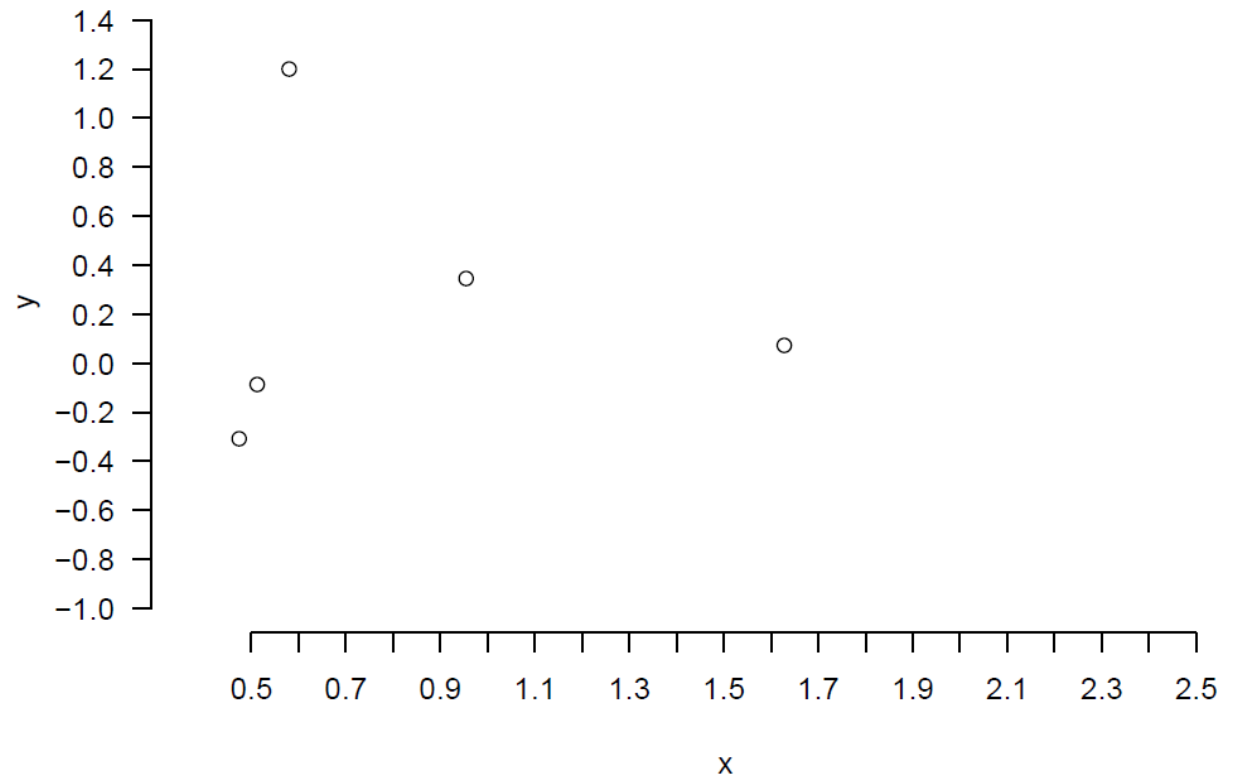


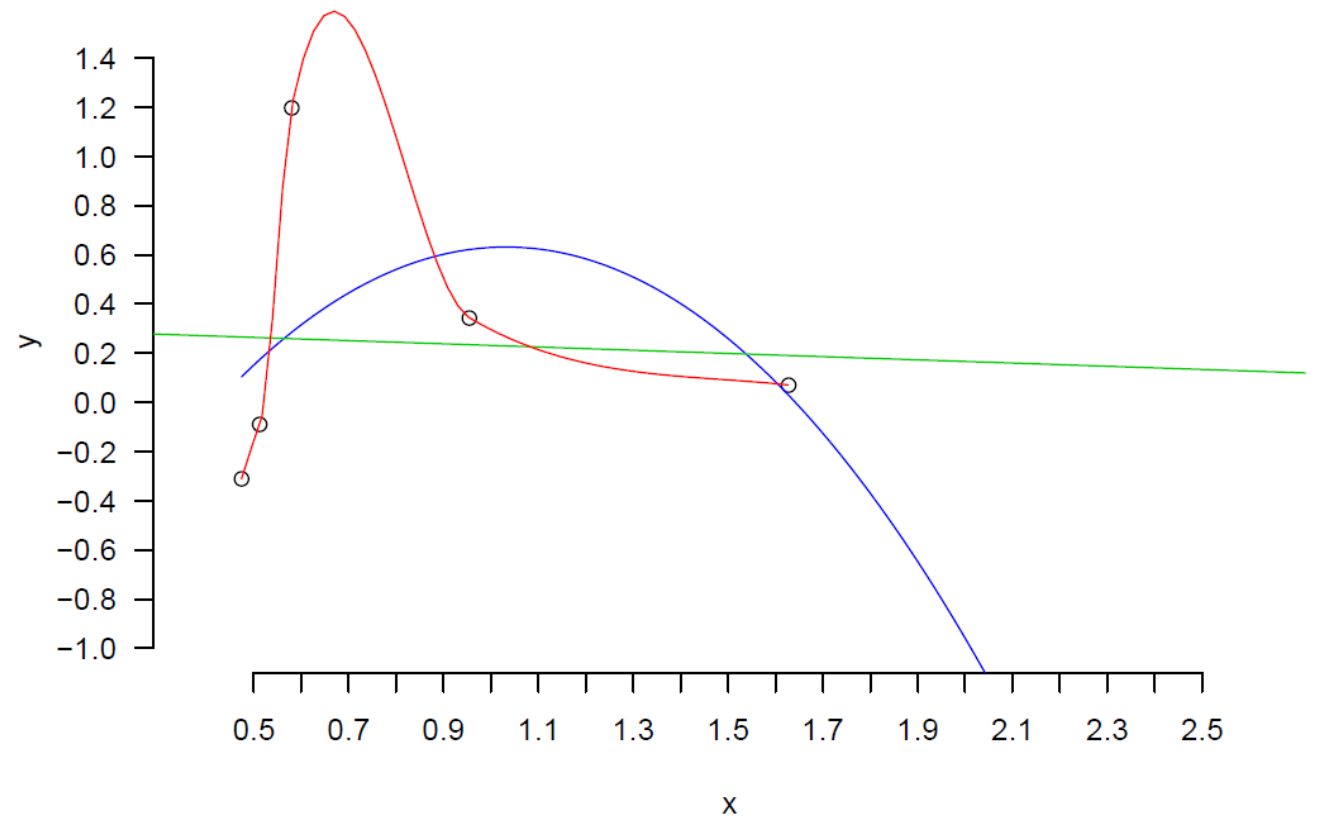


# An exercise in prediction

- 
- I am going to show you a data set, and we are going to try to estimate  $f(x)$  and predict  $y$ ;
  - I generated this data set myself using R, so I know the true  $f(x)$  and distribution of  $\epsilon$ .

- predict (however you want):  
 $y$  for  $x = 0.6, 1.6, 2.0$





## Linear regression:

$$y_i = b_0 + b_1x_i + \epsilon_i \text{ (so } f(x) = b_0 + b_1x \text{)}$$

## Linear regression with quadratic term:

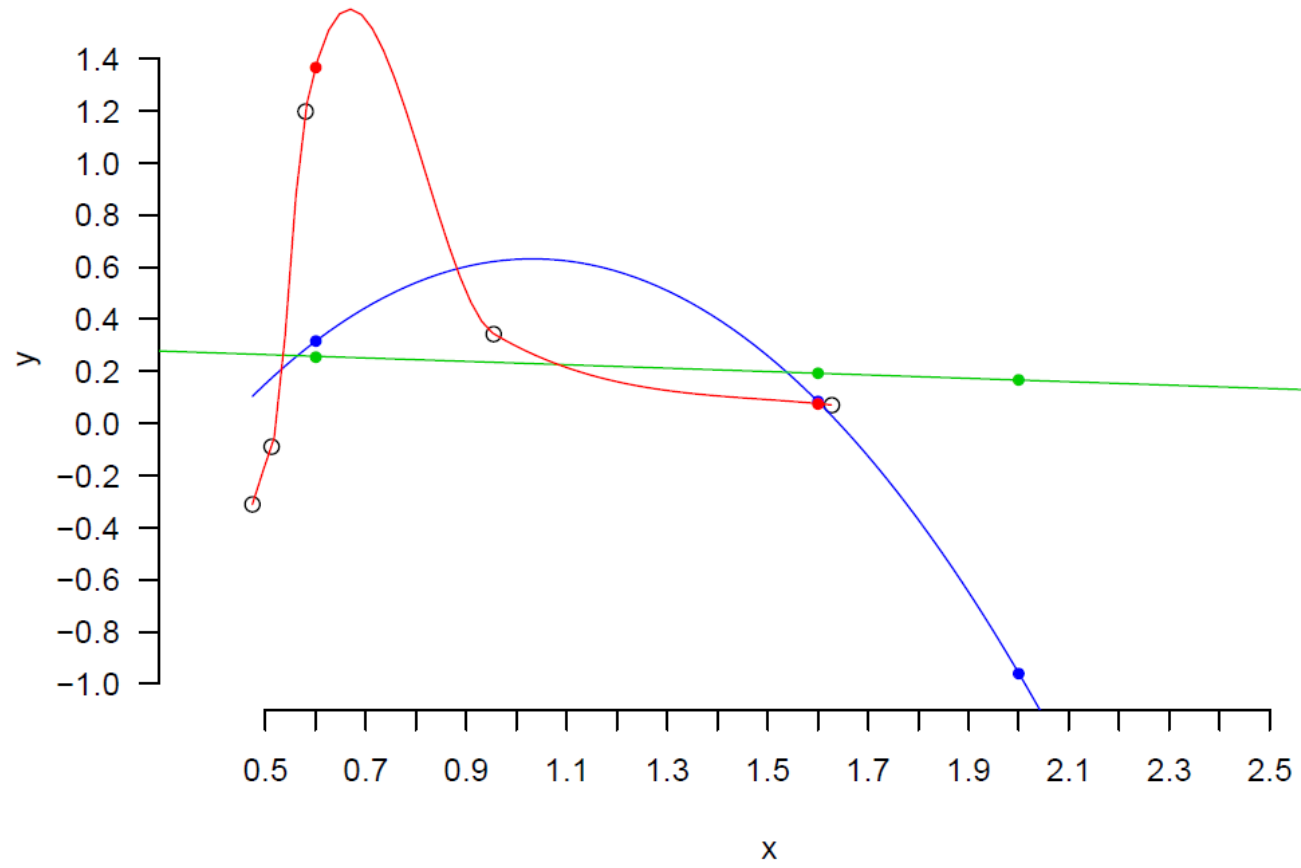
$$y_i = b_0 + b_1x_i + b_2x_i^2 + \epsilon_i$$

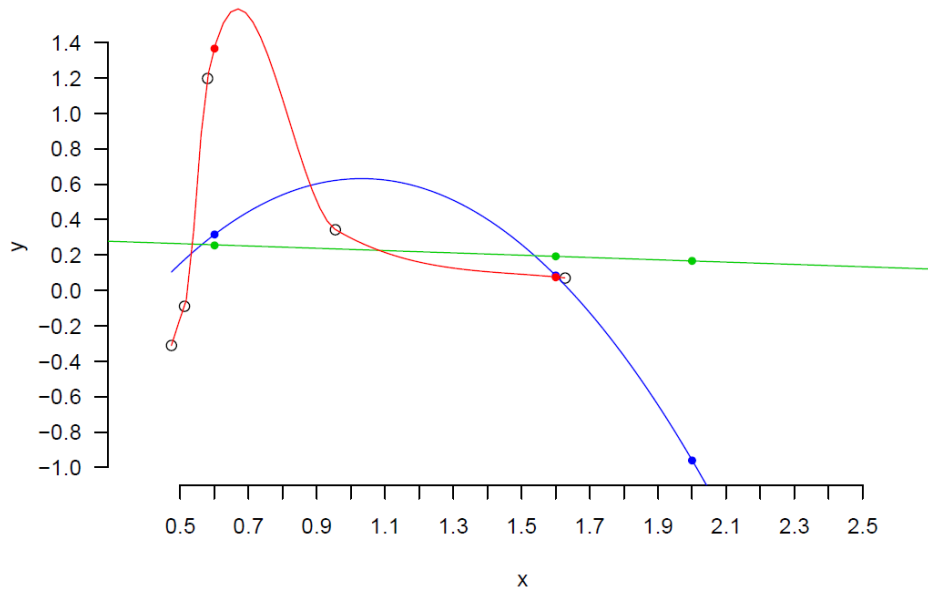
## Nonparametric (loess):

*$y_i$  is predicted from a “local” regression within a window defined by its nearest neighbors.*

(by default: local quadratic regression and neighbors forming 75% of the data)



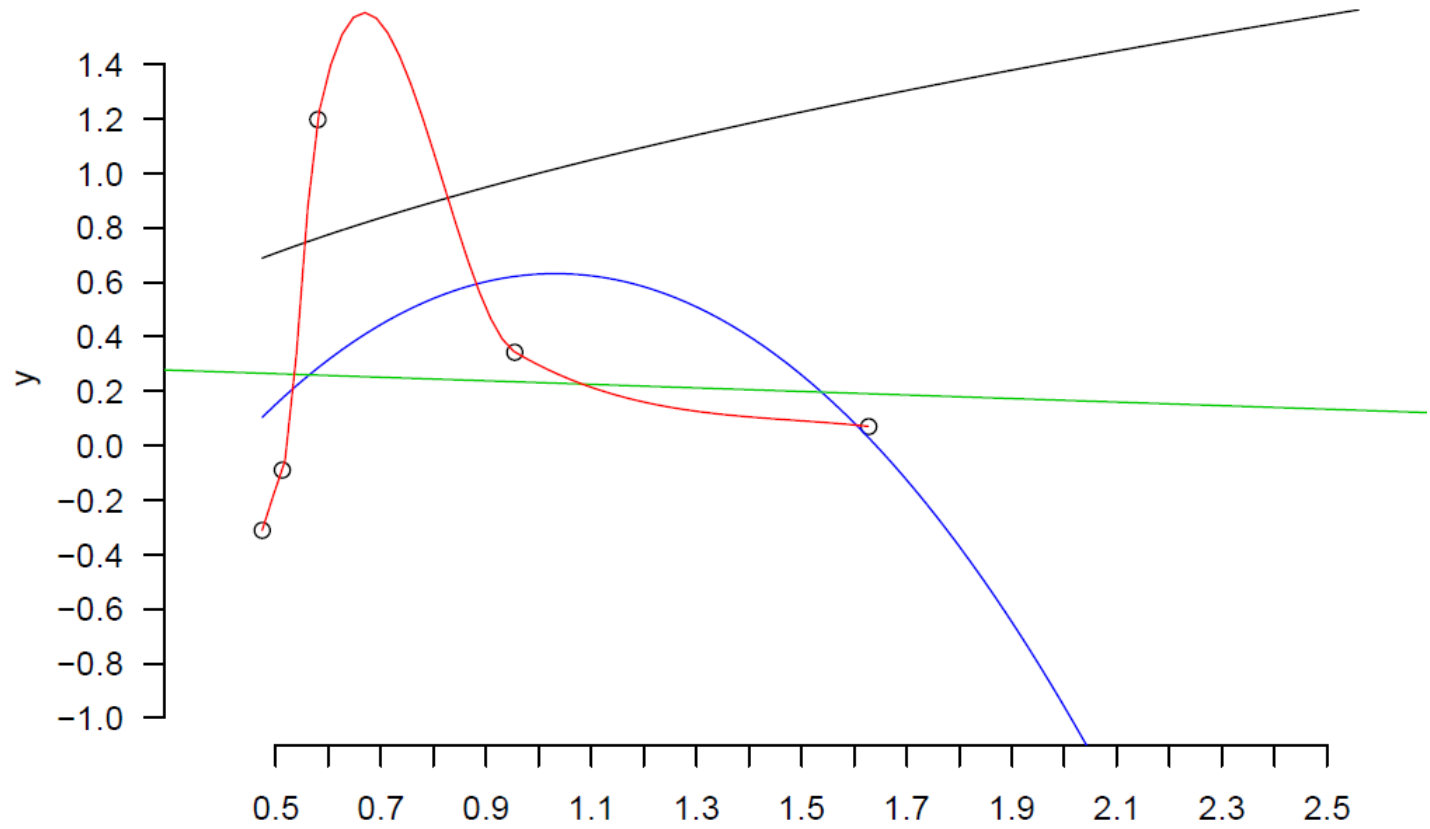


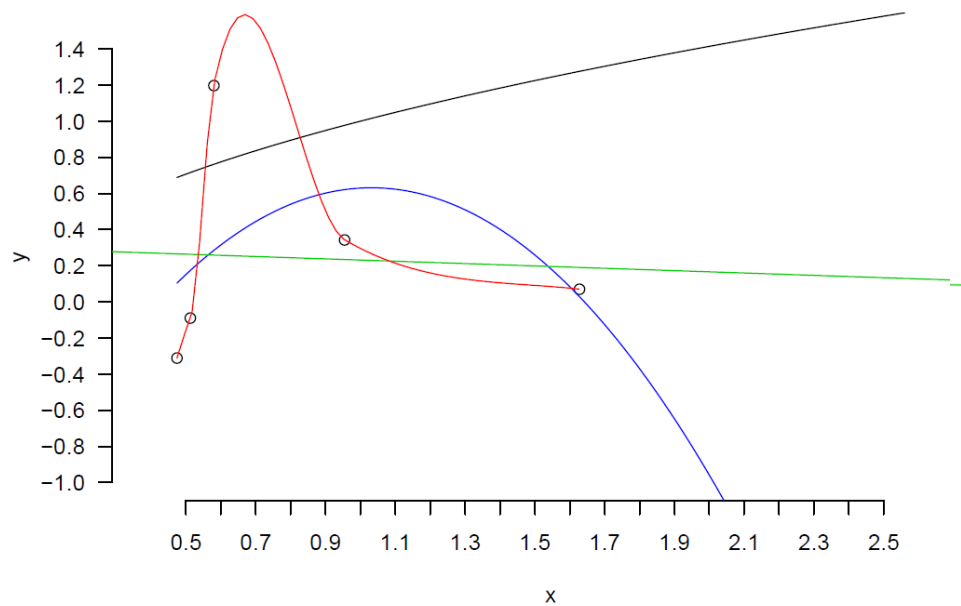


Model	$f(0.6)$	$f(1.6)$	$f(2.0)$
Eyeballing	?	?	?
Linear regression	0.257	0.192	0.166
Linear regression w/ quadratic	0.315	0.084	-0.959
Nonparametric	1.368	0.076	-



The truth (normally we don't know this)






Model	$f(0.6)$	$f(1.6)$	$f(2.0)$
Eyeballing	?	?	?
Linear regression	0.257	0.192	0.166
Linear regression w/ quadratic	0.315	0.084	-0.959
Nonparametric	1.368	0.076	-
<b>Truth</b>	<b>0.775</b>	<b>1.265</b>	<b>1.414</b>



# Model accuracy

- 
- The predictions  $\hat{y}$  differ from the true  $y$ ;
  - We can evaluate how much this happens “on average”.

# A few model evaluation metrics

---

- Mean squared error (MSE): 
$$\text{MSE} = n^{-1} \sum_{i=1}^n (y - \hat{y})^2$$
- Root mean squared error RMSE  $\sqrt{\text{MSE}}$
- Mean absolute error (MAE): 
$$\text{MAE} = n^{-1} \sum_{i=1}^n |y - \hat{y}|$$
- Median absolute error (mAE):  $\text{mAE} = \text{median}|y - \hat{y}|$
- Proportion of variance explained:
- Etc..



## ...And the winner is...

Model	<i>MSE</i>	MSE (interpolation only)
Eyeballing	?	?
Linear regression	0.992	0.709
Linear regression w/ quadratic	2.410	0.883
Nonparametric	-	0.883

True  $f(x)$ :  $y = \sqrt{x} + \epsilon$   
with  $\epsilon \sim \text{Normal}(0; 1)$


# What happened?

---

- There were few observations, relative to the complexity of most models (except linear regression);
- The observed data were a random sample from the true “data-generating process”  $f(x) + \epsilon$ ;

BUT

- By chance, some patterns appeared that are not in the true  $f(x)$ ;
- The more flexible models  $f(x)$  **overfitted** these patterns.

- 
- Imagine we had sampled another 5 observations, re-fitted all of the models, and predicted again. Each time we remember the predictions given. We do this a large number of times, and then take the average for the predictions over all samples.
  - Which model(s) would, on average, give the prediction corresponding exactly to  $f(x) = v(x)$ ?
  - Which models' predictions would vary the most?
  - Which model would you guess (!) to have the lowest MSE, on average?



## **Unbiased:**

*Model that gives the correct prediction, on average over samples from the target population*

- Unbiased in this case: nonparametric, square-root
- Biased in this case: all others

## **High variance:**

*Model that easily overfits accidental patterns.*

- High variance in this case:
- Low variance in this case:

# Bias-variance tradeoff

---

- Flexibility -> lower bias
- Flexibility -> higher variance

**Bias and variance are implicitly linked because they are both affected by model complexity.**

# Possible definitions of “complexity”

---

- Amount of information in data absorbed into model;
- Amount of compression performed on data by model;
- Number of effective parameters, relative to effective degrees of freedom in data.

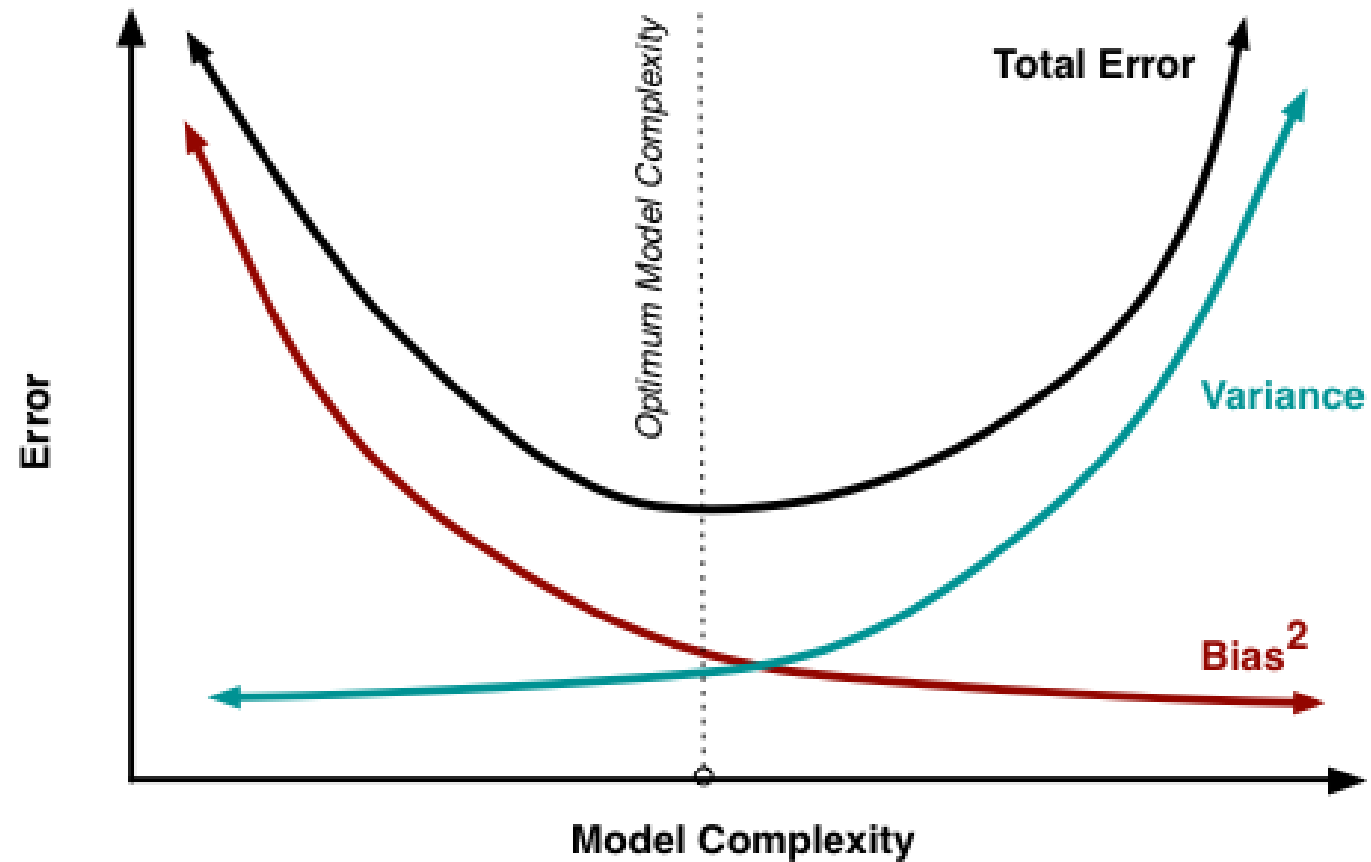
For example:

- More predictors, more complexity;
- Higher-order polynomial, more complexity ( $x$ ,  $x^2$ ,  $x^3$ ,  $x_1 * x_2$ , etc.);
- Smaller “neighborhood” in KNN, more complexity
- ...



Note: bias variance tradeoff occurs just as much with  $n = 1.000.000$  as it does with  $n = 5$ !

# MSE contains both bias and variance





# MSE contains both bias and variance

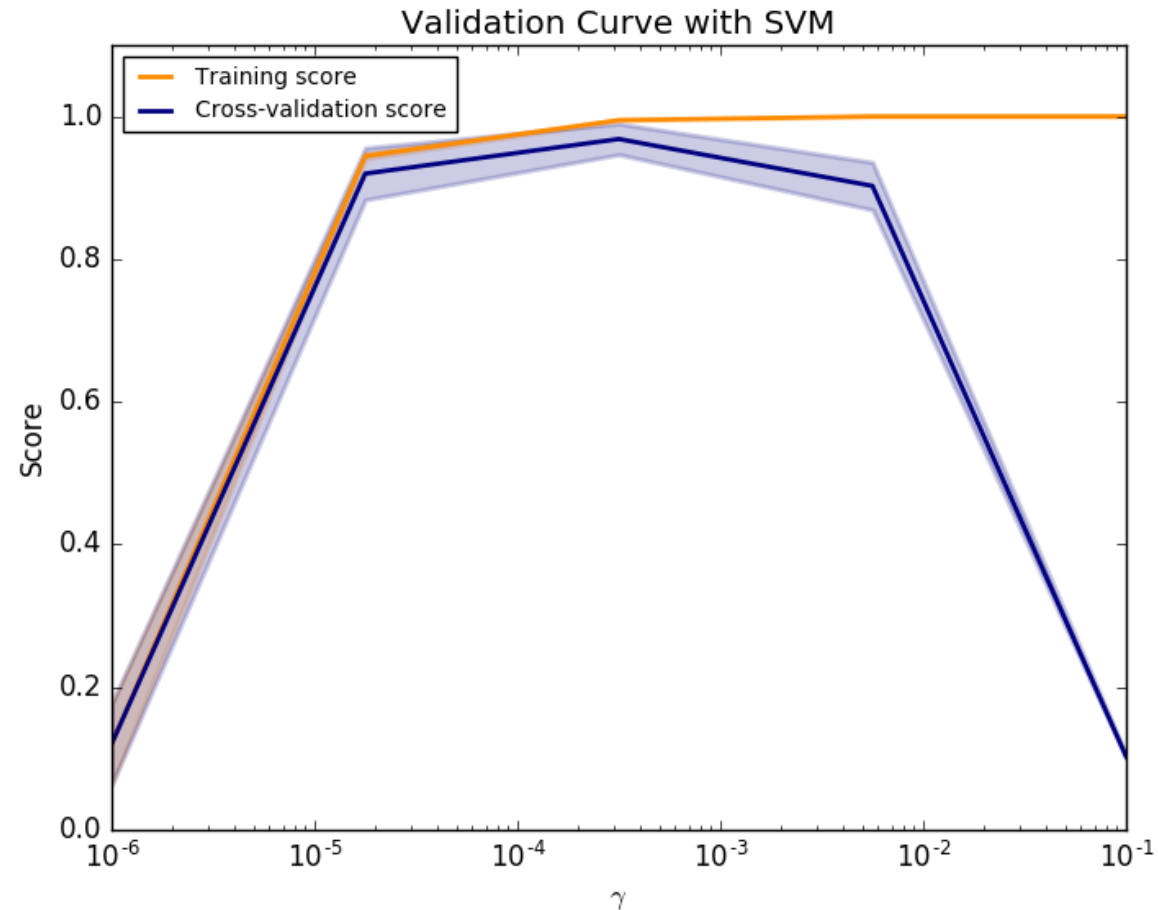
---

$$E(\text{MSE}) = \text{Bias}^2 + \text{Variance} + \sigma^2$$

*Population mean squared error is squared bias PLUS model variance PLUS irreducible variance.*

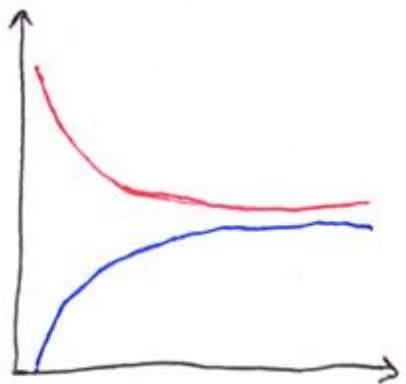
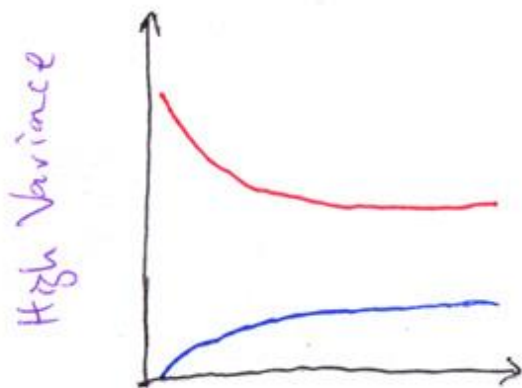
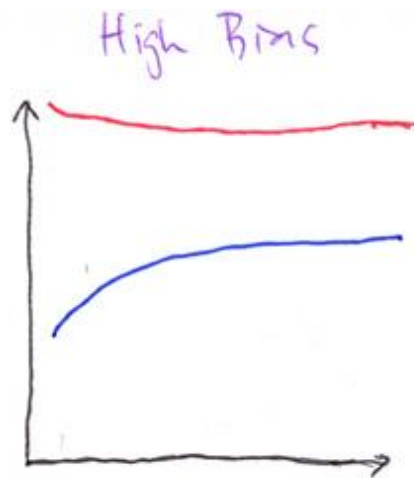
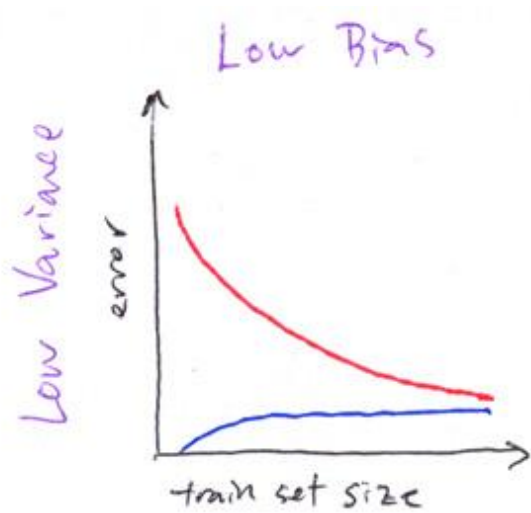
(The  $E(\cdot)$  means “on average over samples from the target population”).

# Observed training and test error in a flexible model





# Learning curves



- Which curves are training and which are test error?

# What this means in practice

---

Sometimes a wrong model is better than a true model (on average etc);

If you do not believe in true models: sometimes a simple model is better than a more complex one.

These factors **together** determine what works best:

- How close the functional form of  $f(x)$  is to the true  $f(x)$ ;
- The amount of irreducible variance;
- The sample size ( $n$ );
- The complexity of the model ( $p/df$  or equivalent).



# Training-validation-test paradigm



So far, I have **cheated**;

I **knew** the true  $f(x)$  so you could calculate exactly what  $E(\text{MSE})$  was;

This is sometimes called the “Bayes error”.

In practice we do not know the truth;

- **How can we estimate  $E(\text{MSE})$ ?**

# Train/dev/test

---

## **Training data:**

Observations used to fit  $f(x)$

## **Validation data** (or “dev” data):

New observations from the same source as training data  
*(Used several times to select model complexity)*

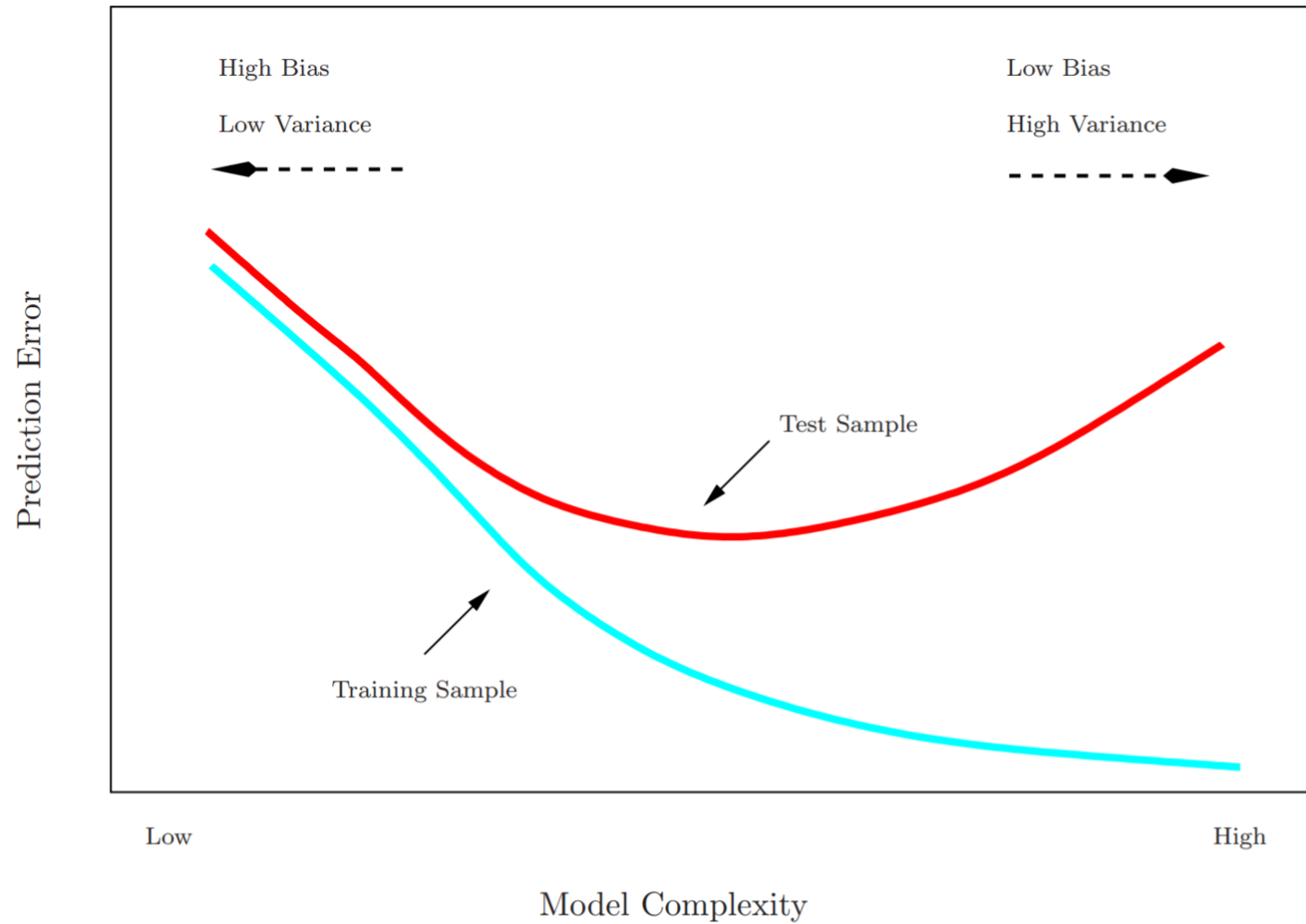
## **Test data:**

New observations from the intended prediction situation

**Question:** Why don't these give the same average MSE?



# Train/dev/test



# Train/dev/test

---

- The idea is that the average squared error in the test set  $MSE_{\text{test}}$  is a good estimate of the Bayes error  $E(\text{MSE})$
- This only holds when the test set is “like” the intended prediction situation!

# Drawbacks of train/dev/test

---

- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to overestimate the test error for the model fit on the entire data set.

From <https://lagunita.stanford.edu/c4x/HumanitiesScience/StatLearning>

# K-fold crossvalidation

---

# Conclusion

---

- Bias and variance trade off, in theory;
- Bias and variance trade off, in practice;
- We try to estimate this using train/dev/test paradigm;
- It is important to be ruthless when applying this setup;
- Getting good test data is **difficult, unsolved problem**;
- Cross-validation is a useful alternative to separate dev set;
- Beware that **any procedure that makes decisions based on the data** requires validation!