Data Science 1

STAT/CS 287

Jim Bagrow, UVM Dept of Math and Statistics

LECTURE 18

Today

More on predictive models (*supervised learning*)

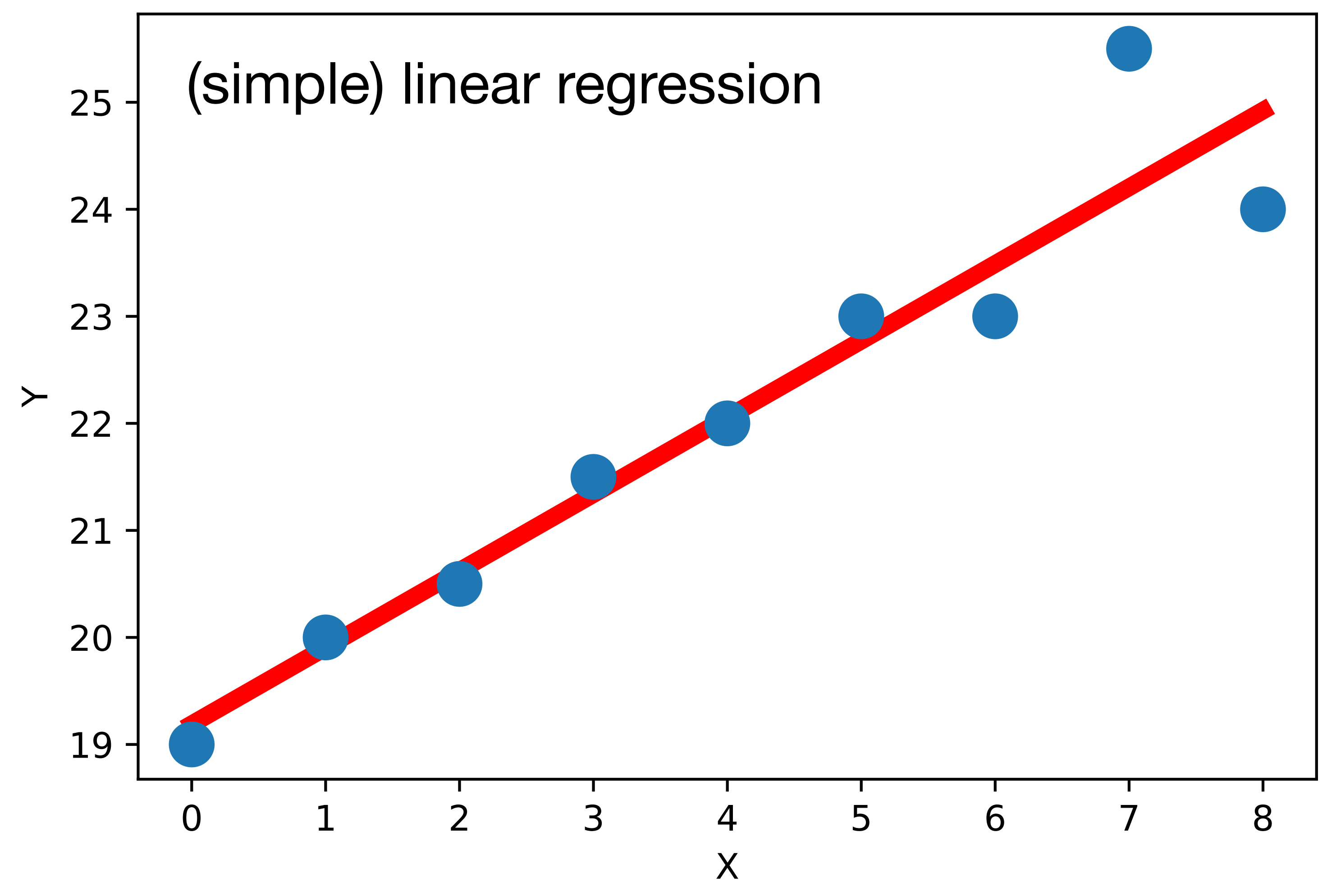
Today

More on predictive models (*supervised learning*)

Prediction vs. Inference → Linear Regression

Prediction

new x comes in, predict y using $y = f(x) = \beta_0 + \beta_1 x$



Today

More on predictive models (*supervised learning*)

Prediction vs. Inference → Linear Regression

Prediction

new x comes in, predict y using $y = f(x) = \beta_0 + \beta_1 x$

Inference

learn how changing x changes y by examining β's

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 5.849e+04
Date:                Thu, 31 Oct 2019   Prob (F-statistic):          3.44e-150
Time:                        10:25:51   Log-Likelihood:                 153.19
No. Observations:                 100   AIC:                            -300.4
Df Residuals:                      97   BIC:                            -292.6
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.9926      0.016     60.925      0.000       0.960       1.025
x1             0.0562      2.517      0.022      0.982      -4.939       5.051
x2             0.5034      0.198      2.539      0.013       0.110       0.897
==============================================================================
Omnibus:                        4.957   Durbin-Watson:                   1.903
Prob(Omnibus):                  0.084   Jarque-Bera (JB):                2.704
Skew:                           0.153   Prob(JB):                        0.259
Kurtosis:                       2.255   Cond. No.                     3.53e+03
==============================================================================
```

Today

More on predictive models (*supervised learning*)

Prediction vs. Inference → Linear Regression

Prediction

new x comes in, predict y using $y = f(x) = \beta_0 + \beta_1 x$

Inference

learn how changing x changes y by examining β's

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 5.849e+04
Date:                Thu, 31 Oct 2019   Prob (F-statistic):          3.44e-150
Time:                        10:25:51   Log-Likelihood:                 153.19
No. Observations:                 100   AIC:                            -300.4
Df Residuals:                      97   BIC:                            -292.6
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.9926      0.016     60.925      0.000       0.960       1.025
x1             0.0562      2.517      0.022      0.982      -4.939       5.051
x2             0.5034      0.198      2.539      0.013       0.110       0.897
==============================================================================
Omnibus:                        4.957   Durbin-Watson:                   1.903
Prob(Omnibus):                  0.084   Jarque-Bera (JB):                2.704
Skew:                           0.153   Prob(JB):                        0.259
Kurtosis:                       2.255   Cond. No.                     3.53e+03
==============================================================================
```

Recall

Natural Language Processing Tasks & Semantic Similarity

Supervised Learning — Classifiers

Documents and labels:

$$\begin{array}{c c} & \text{— \# unique words (types) —} \\ \text{— \# documents —} & \underbrace{\begin{bmatrix} 0\ 0\ 0\ 7\ 3\ 0\ 0\ 1 \cdots 0\ 5\ 0\ 0\ 0 \\ \vdots \quad \ddots \qquad\qquad \vdots \\ \\ \\ \end{bmatrix}}_{X} \end{array} \qquad \underbrace{\begin{bmatrix} \text{spam} \\ \text{spam} \\ \vdots \\ \text{spam} \\ \text{ham} \\ \vdots \\ \text{ham} \end{bmatrix}}_{y}$$

— Labels vector

— Training data

$$y = f(X)$$

A new, unlabeled document comes in:

built a machine $\hat{f}$ to predict label given count vector —

$$\hat{f}([\,0\ 0\ 0\ 2\ 6\ 3\ 0\ 5 \cdots 0\ 0\ 11\ 0\ 0\,]) =$$

$$\arg\max\,[\Pr(\text{spam}), \Pr(\text{ham})] =$$

$$\arg\max\,[\,0.6, 0.4\,] =$$

spam

(for example)

$$y = \hat{f}(X)$$

Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

Examples

	X	y
KIDS dataset:	features of hospitalizations	outcome of hospital stay
Image classification:	(raw) image features	label of subject of image
Finance:	values of stocks, bonds, foreign exchange	tomorrow's stock price of $IBM

Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

Examples

literal predictions

	X	y
KIDS dataset:	features of hospitalizations	outcome of hospital stay
Image classification:	(raw) image features	label of subject of image
Finance:	values of stocks, bonds, foreign exchange	tomorrow's stock price of $IBM

Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

Examples

literal predictions

	X	y
KIDS dataset:	features of hospitalizations	outcome of hospital stay
Image classification:	(raw) image features	label of subject of image
Finance:	values of stocks, bonds, foreign exchange	tomorrow's stock price of $IBM

Why build a predictive model?

X **- readily available** y **- very hard to come by** **so approximate** y **with** $\hat{y} = \hat{f}(X)$

Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

How to build a predictive model?

Invest in the effort to generate training data, many X, y pairs

Figure out a good $\hat{f}$ by comparing $\hat{f}(X)$ and y when both are known - training or learning

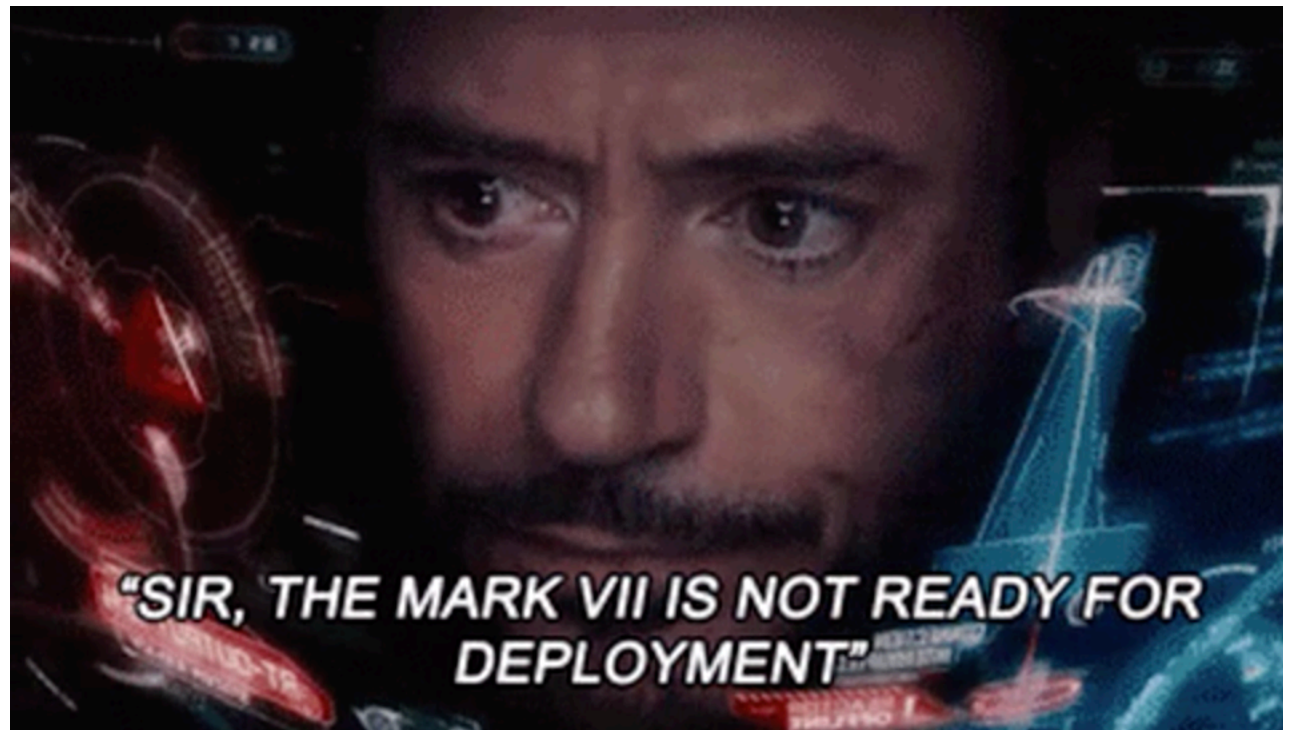
Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

Never forget the **guiding principle** – **deployment**

Invest in the effort to generate training data, many examples X, y pairs	**X - readily available**
Figure out a good $\hat{f}$ by comparing $\hat{f}(X)$ and y when both are known - training or learning	**y - very hard to come by**
	so approximate y with $\hat{y} = \hat{f}(X)$



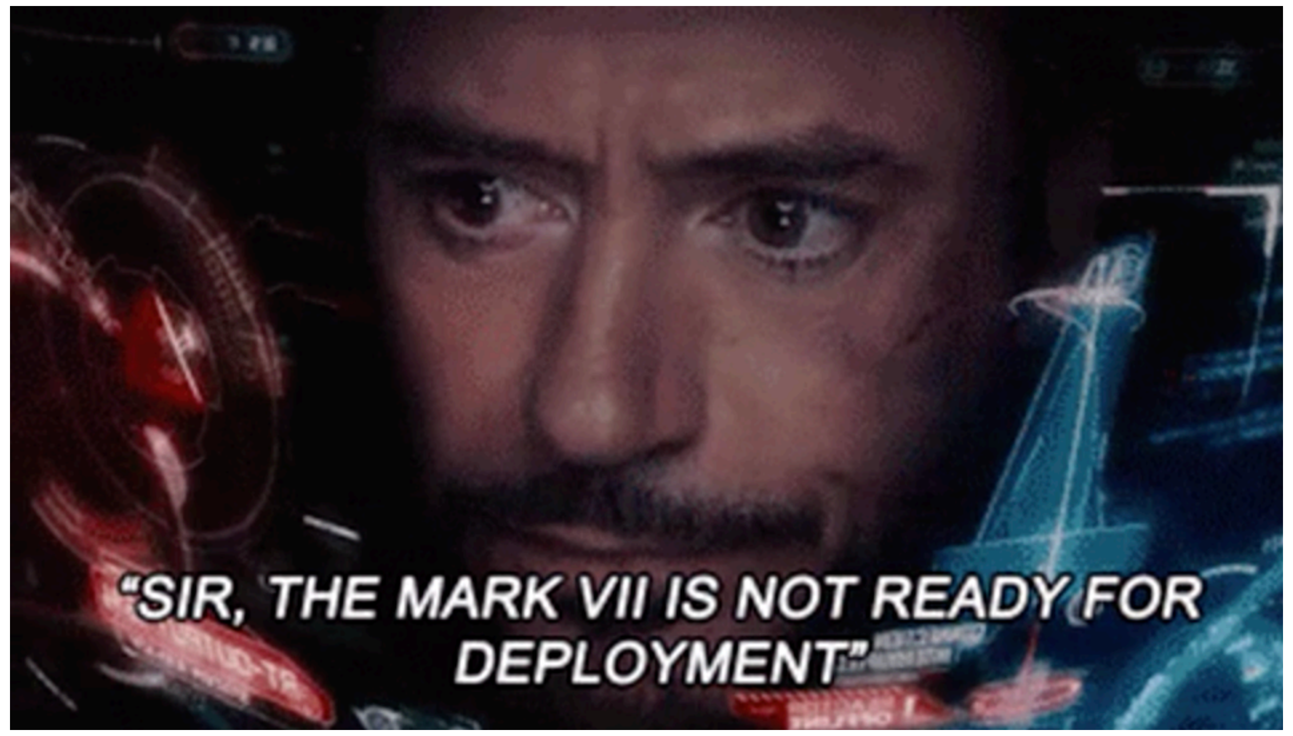
Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

Never forget the **guiding principle** – **deployment**

Invest in the effort to generate training data, many examples X, y pairs	**X - readily available**
Figure out a good $\hat{f}$ by comparing $\hat{f}(X)$ and y when both are known - training or learning	**y - very hard to come by**
	so approximate y with $\hat{y} = \hat{f}(X)$



Predictive Models

$$y \approx \hat{f}(X)$$

Use function $\hat{f}$ to predict an unknown y given a known X

Never forget the **guiding principle** – **deployment**

Invest in the effort to generate training data, many examples X, y pairs

Figure out a good $\hat{f}$ by comparing $\hat{f}(X)$ and y when both are known - training or learning

X - readily available

y - very hard to come by

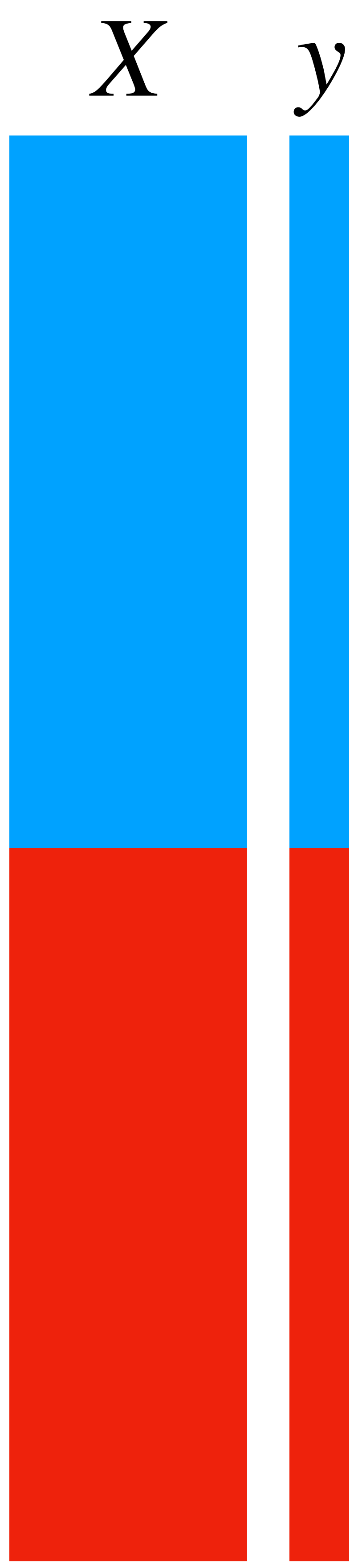
so approximate y with $\hat{y} = \hat{f}(X)$

Very easy, especially for beginners, to get lost fitting $\hat{f}$ to data (error metrics, cross-validation, hyperparameters) but remember: you are building a system that works without knowing y

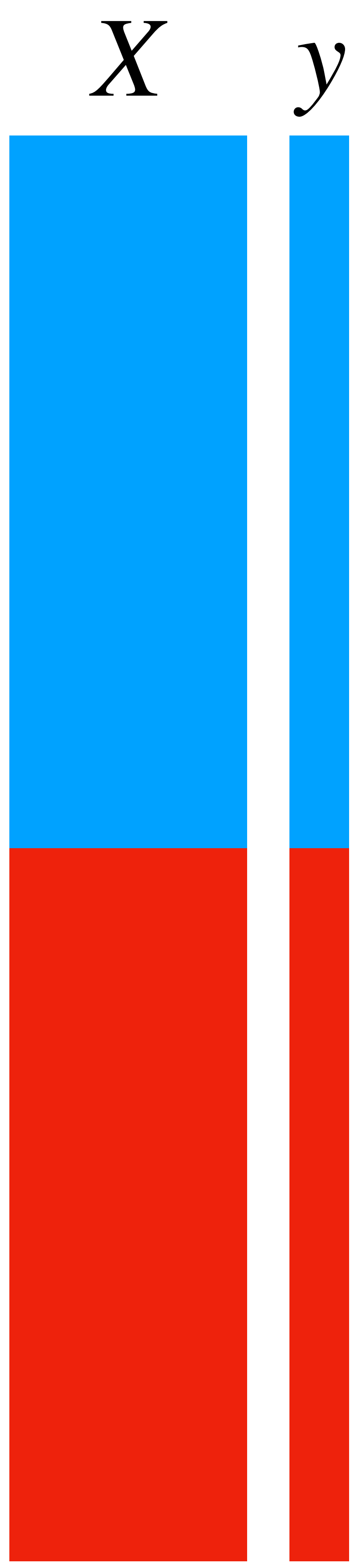
Simulating Deployment: Training / Testing

50 / 50 split

(90/10 also common)



Simulating Deployment: Training / Testing



50 / 50 split

(90/10 also common)

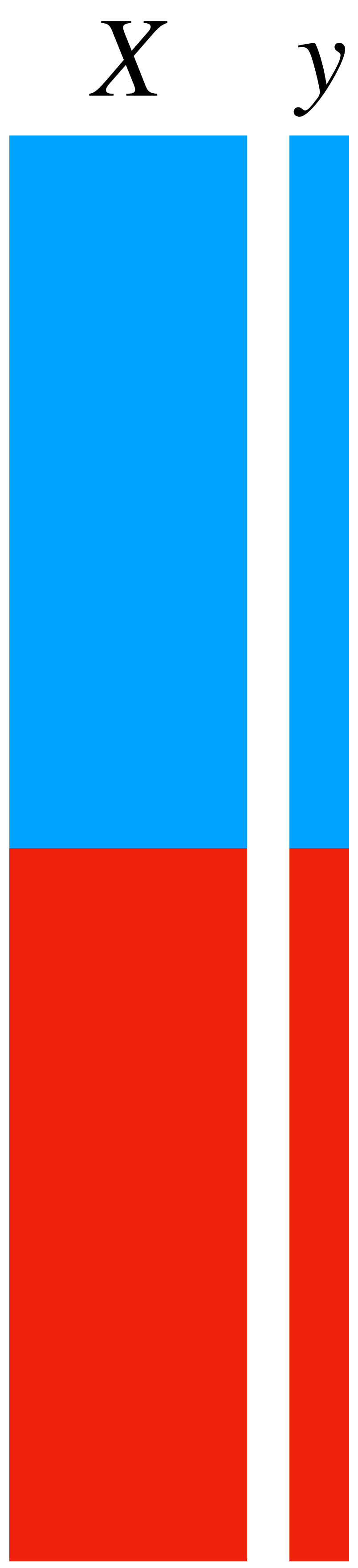
Predictive model is given X_{tr} and y_{tr} to **learn**

$$\hat{f}(X) \approx f(X)$$

Simulating Deployment: Training / Testing

50 / 50 split

(90/10 also common)



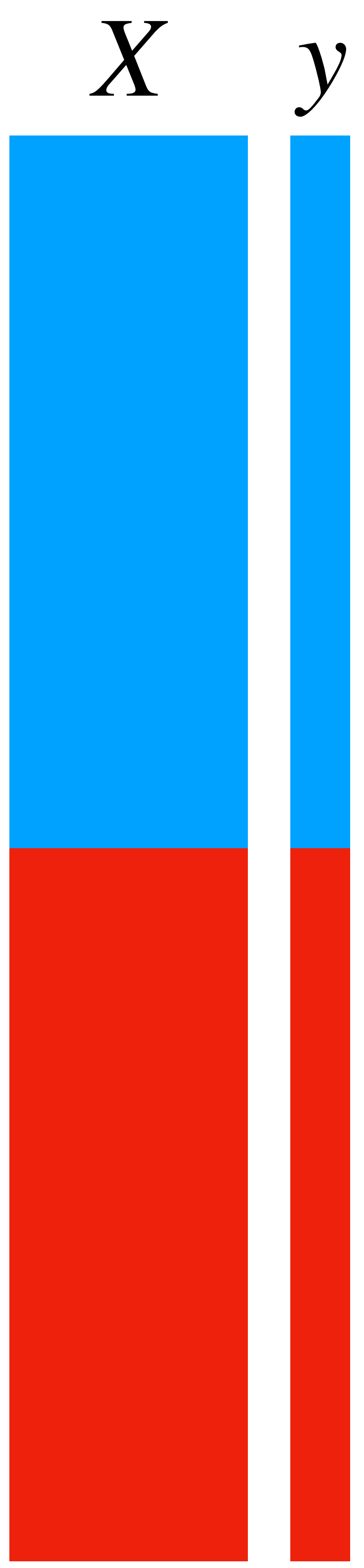
Predictive model is given X_{tr} and y_{tr} to **learn**

$$\hat{f}(X) \approx f(X)$$

Predictive model is **tested** with X_{te} and y_{te} by comparing $\hat{f}(X_{\text{te}})$ against y_{te}

- Example *error metric* (sum of squared errors): $\|\hat{f}(X) - y\|_2^2$

Simulating Deployment: Training / Testing



50 / 50 split

(90/10 also common)

Predictive model is given X_{tr} and y_{tr} to **learn**
$\hat{f}(X) \approx f(X)$

Predictive model is **tested** with X_{te} and y_{te} by comparing $\hat{f}(X_{\text{te}})$ against y_{te}

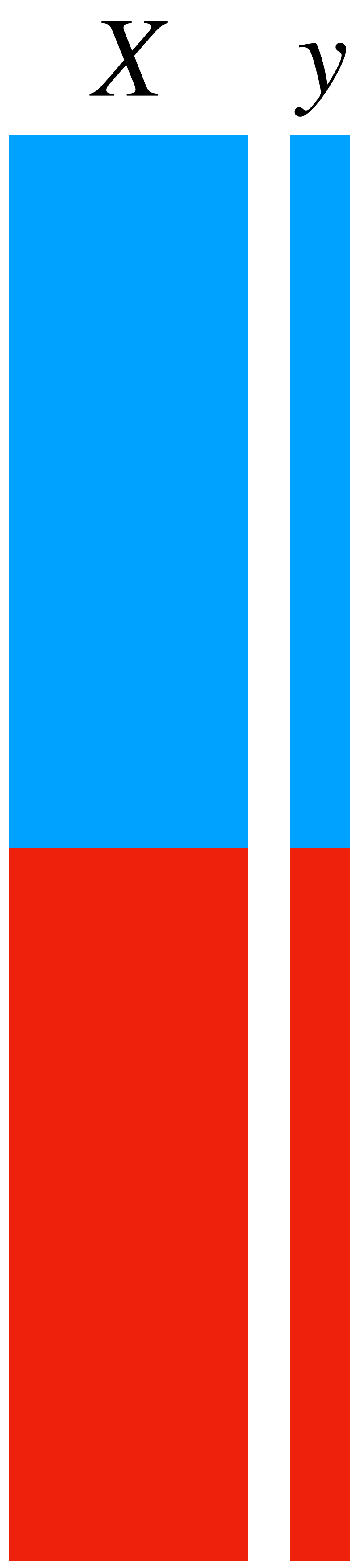
- Example *error metric* (sum of squared errors): $\|\hat{f}(X) - y\|_2^2$

Why not use all the data to learn,
maximize the information we have?

Simulating Deployment: Training / Testing

50 / 50 split

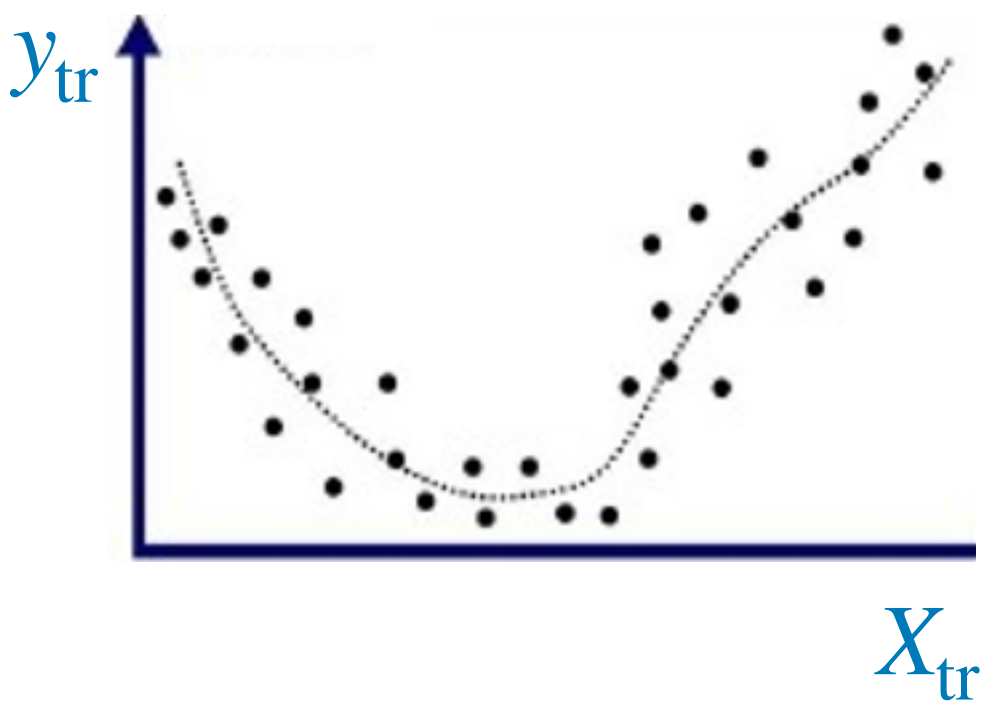
(90/10 also common)



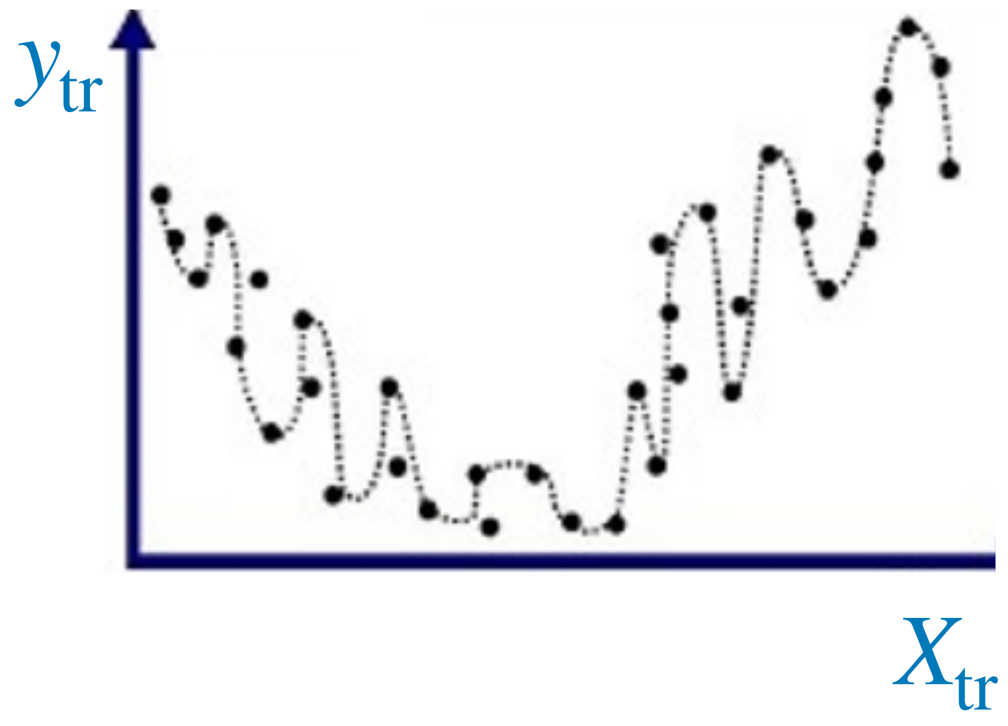
Why not use all the data to learn, maximize the information we have?

To prevent overfitting

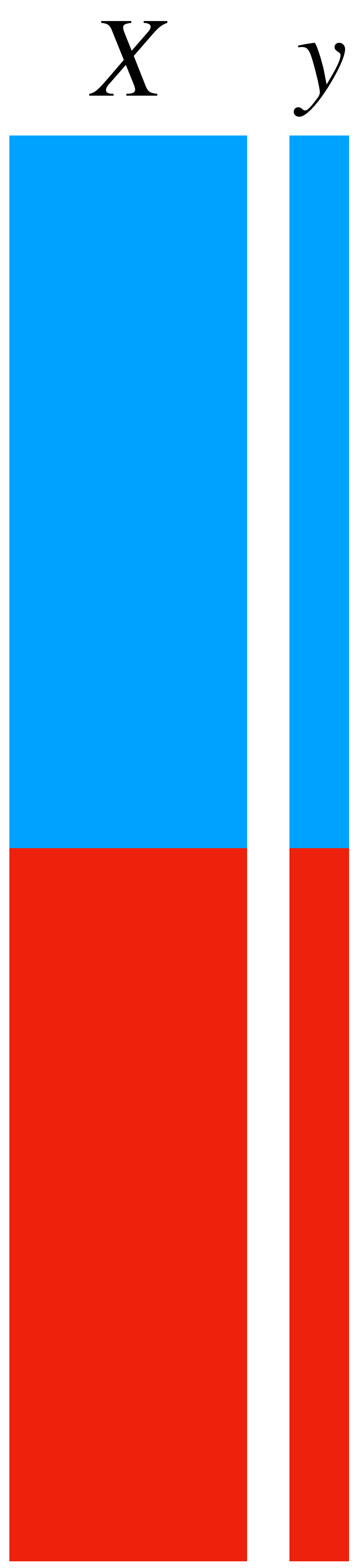
Good fit (?)



Overfit!



Simulating Deployment: Training / Testing

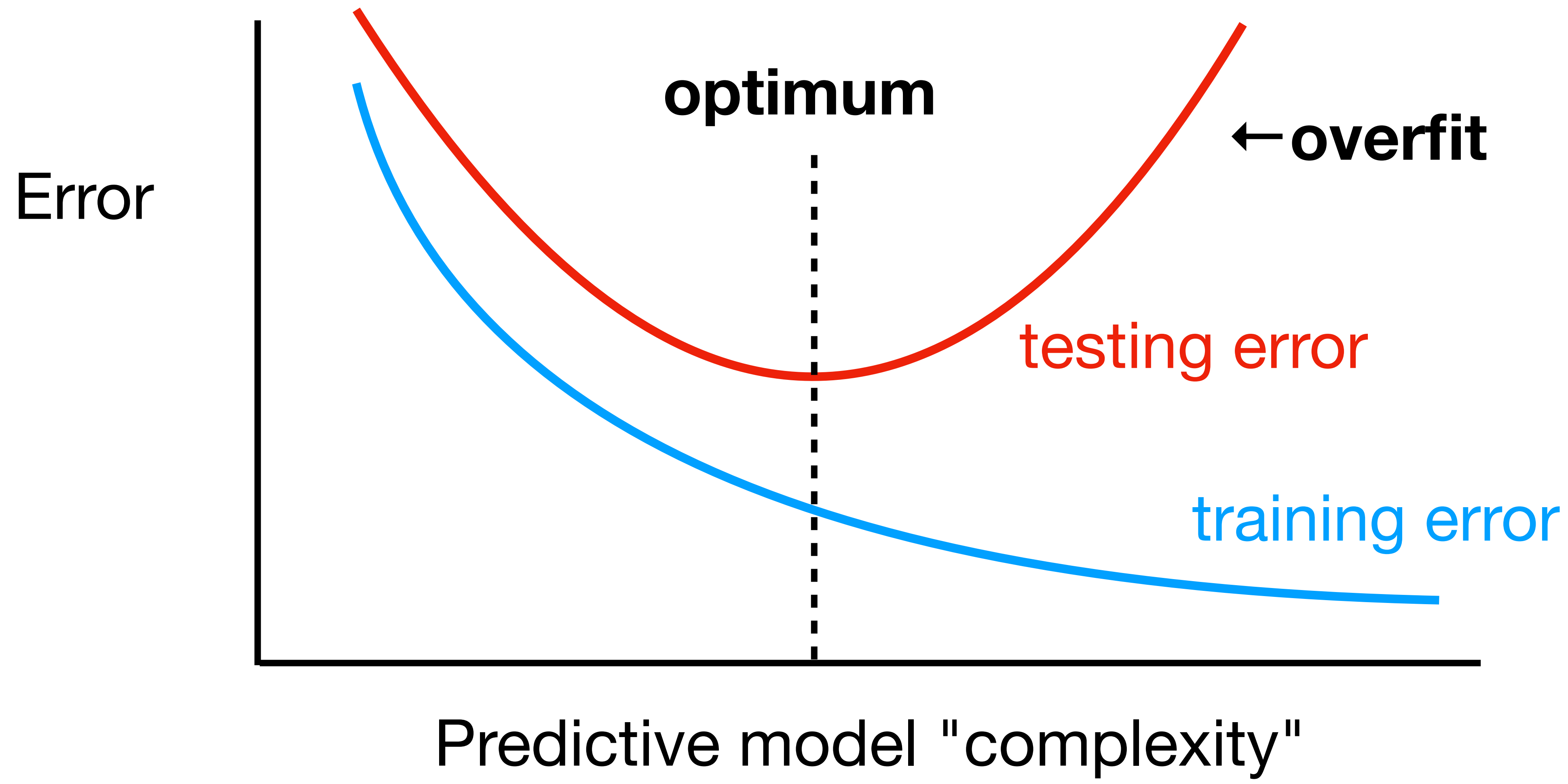


50 / 50 split

(90/10 also common)

Why not use all the data to learn, maximize the information we have?

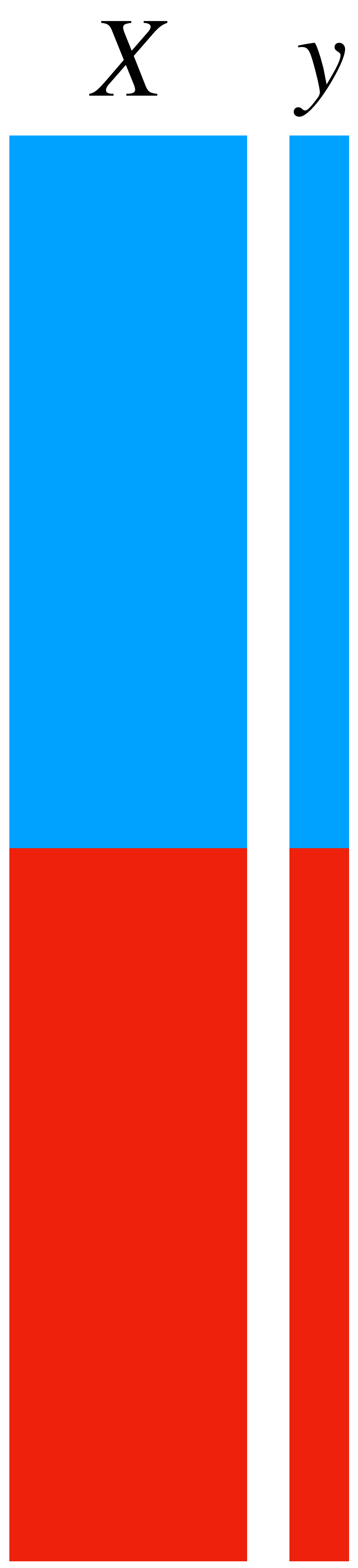
To prevent overfitting



Training / Validation / Testing

50 / 50 split

(90/10 also common)

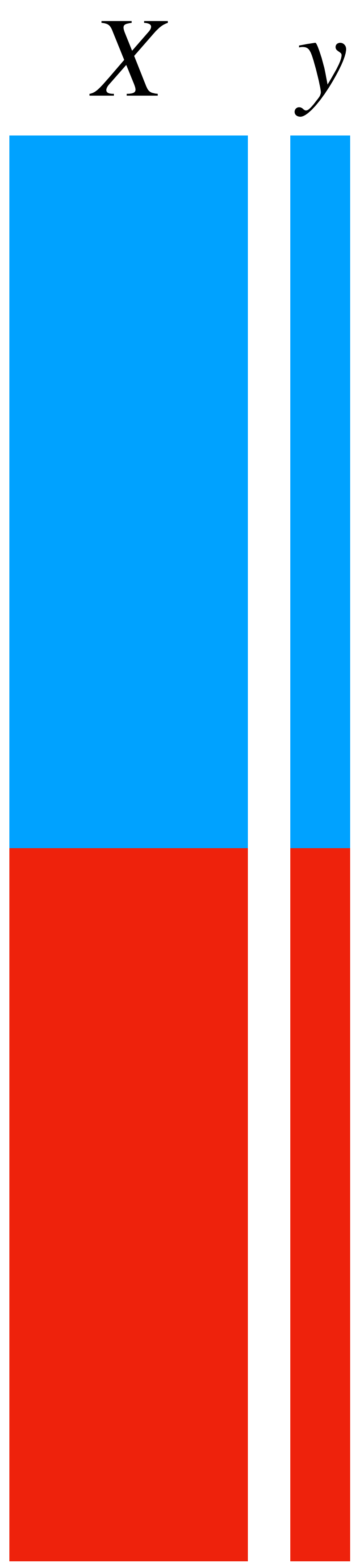


Many predictive models have both *parameters* and *hyperparameters*

Parameters: changed during/due to training

Hyperparameters: chosen <u>before</u> training

Training / Validation / Testing



50 / 50 split

(90/10 also common)

Many predictive models have both *parameters* and *hyperparameters*

Parameters: changed during/due to training

Hyperparameters: chosen <u>before</u> training

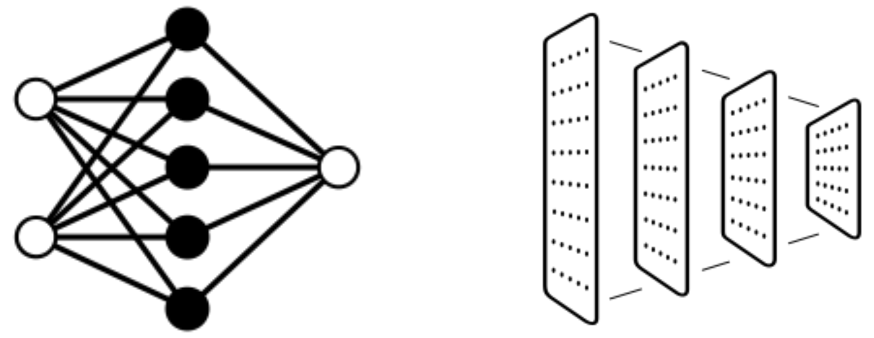
Polynomial regression

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$$

Parameters: coefficients β_i

Hyperparameter: polynomial order d

Neural networks

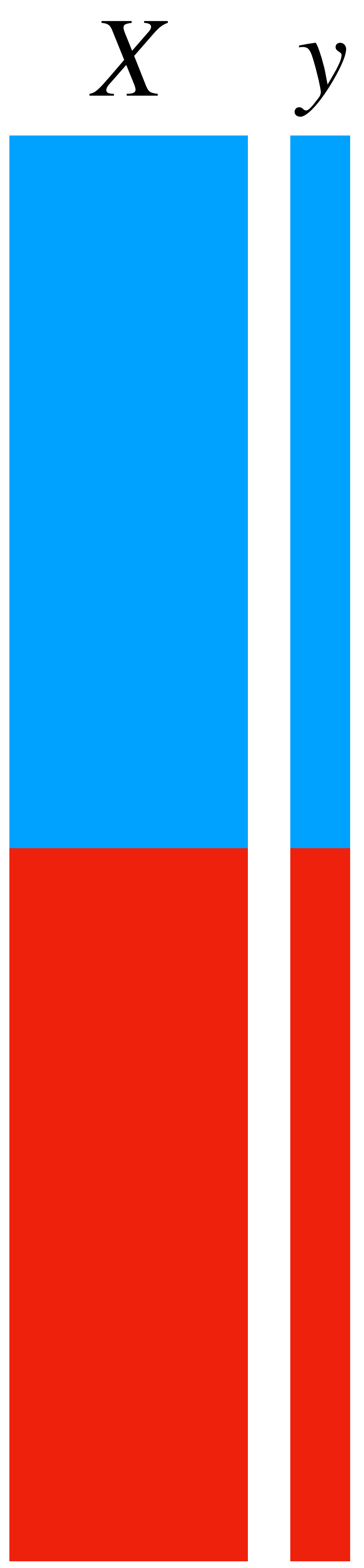


Parameters: weights on links

Hyperparameters:

Network architecture
Choice of activation function
…

Training / Validation / Testing



50 / 50 split

(90/10 also common)

Many predictive models have both *parameters* and *hyperparameters*

Parameters: changed during/due to training

Hyperparameters: chosen <u>before</u> training

You could:

1. Use training data to fit parameters
2. Use testing data to compare different hyperparameters

But:

- Risk overfitting again—all your data went into the model, nothing is left over for testing

Training / Validation / Testing

Many predictive models have both *parameters* and *hyperparameters*

Parameters: changed during/due to training
Hyperparameters: chosen <u>before</u> training

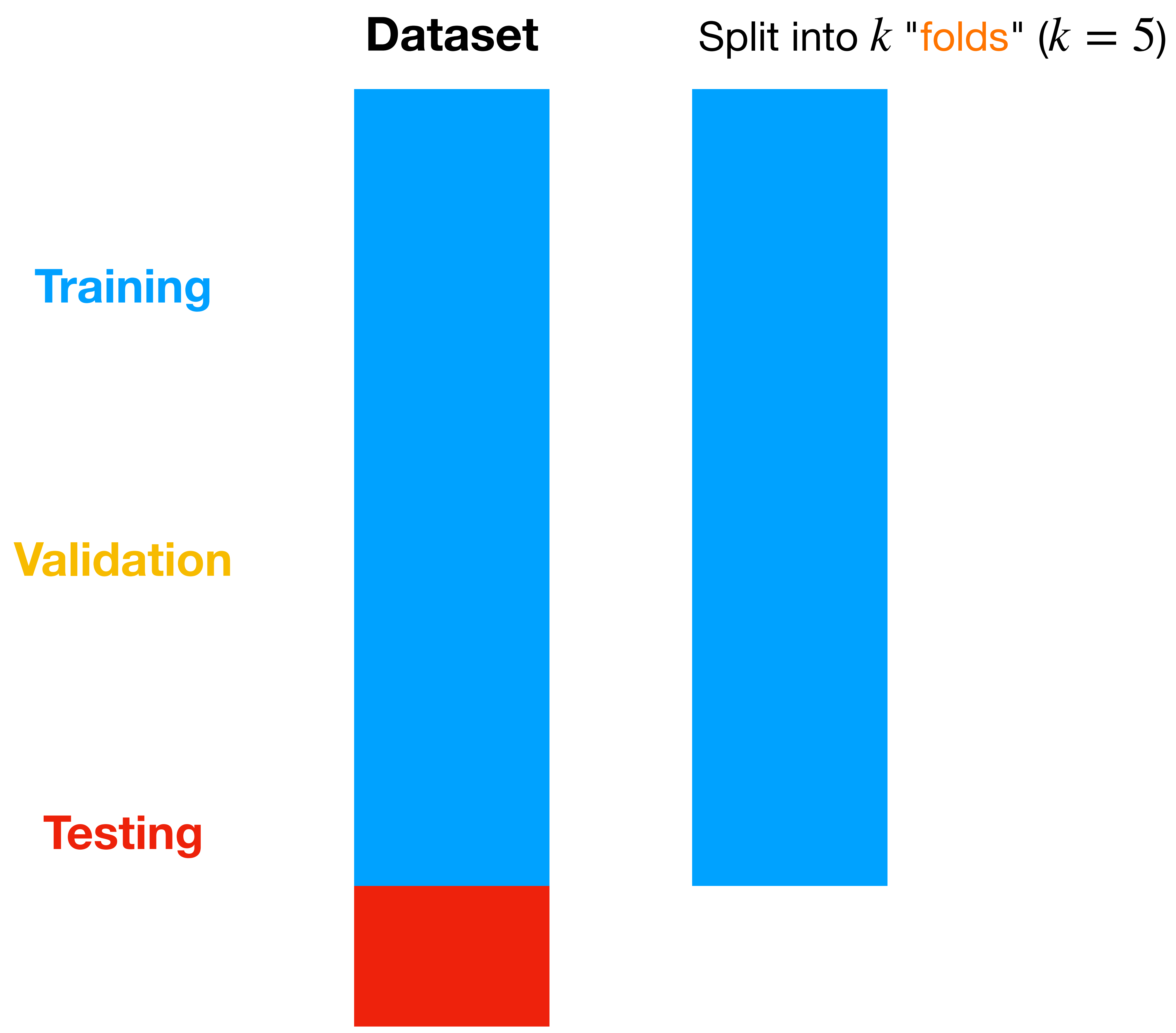
Instead:

1. Use **training data** to fit parameters
2. Use validation data to compare different hyperparameters
3. Use **testing data** to pick best model

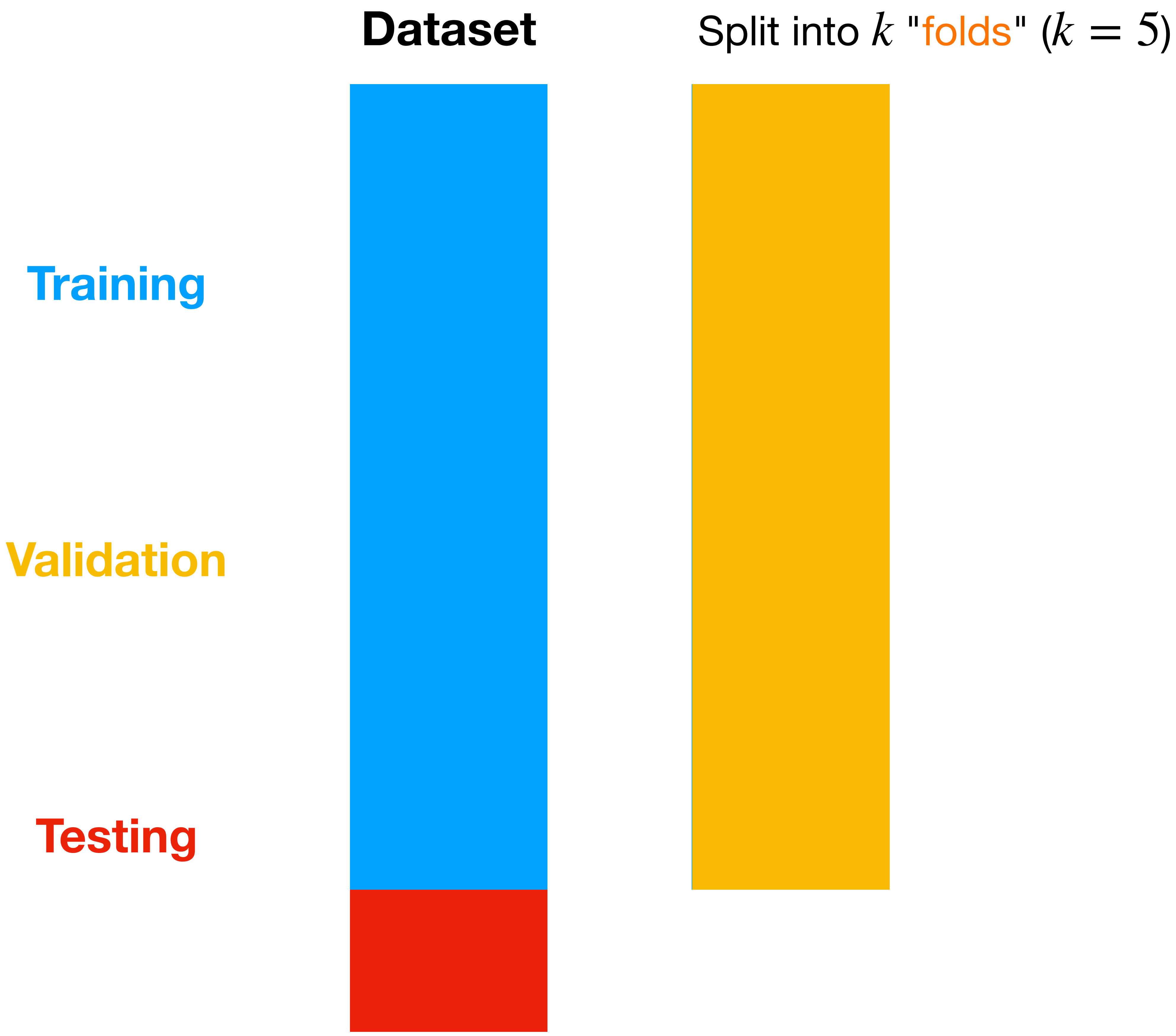
One issue:

May need **multiple** training/validation splits, to get enough statistics (repetitions) to make good comparisons

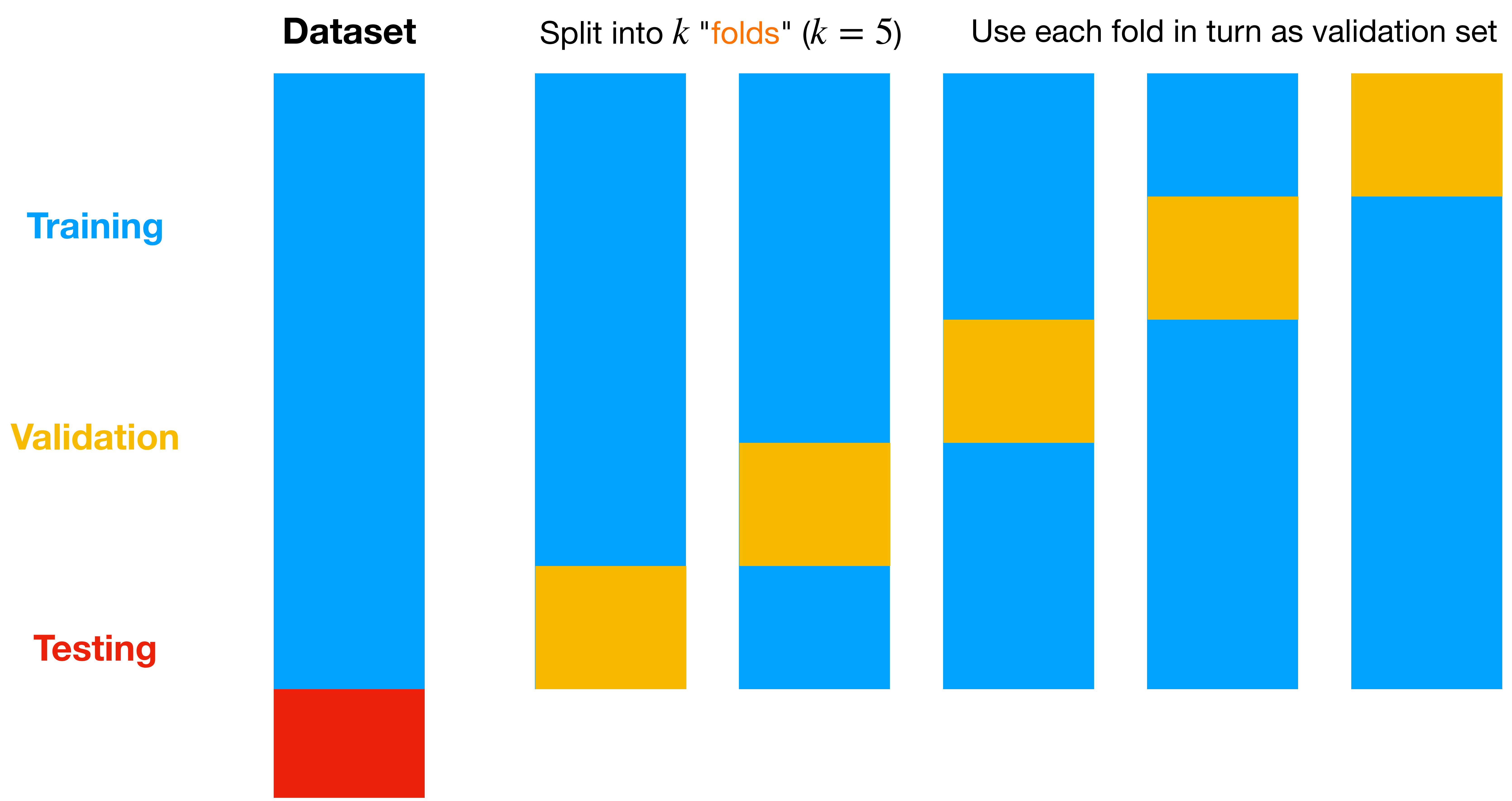
(k-Fold) Cross-validation



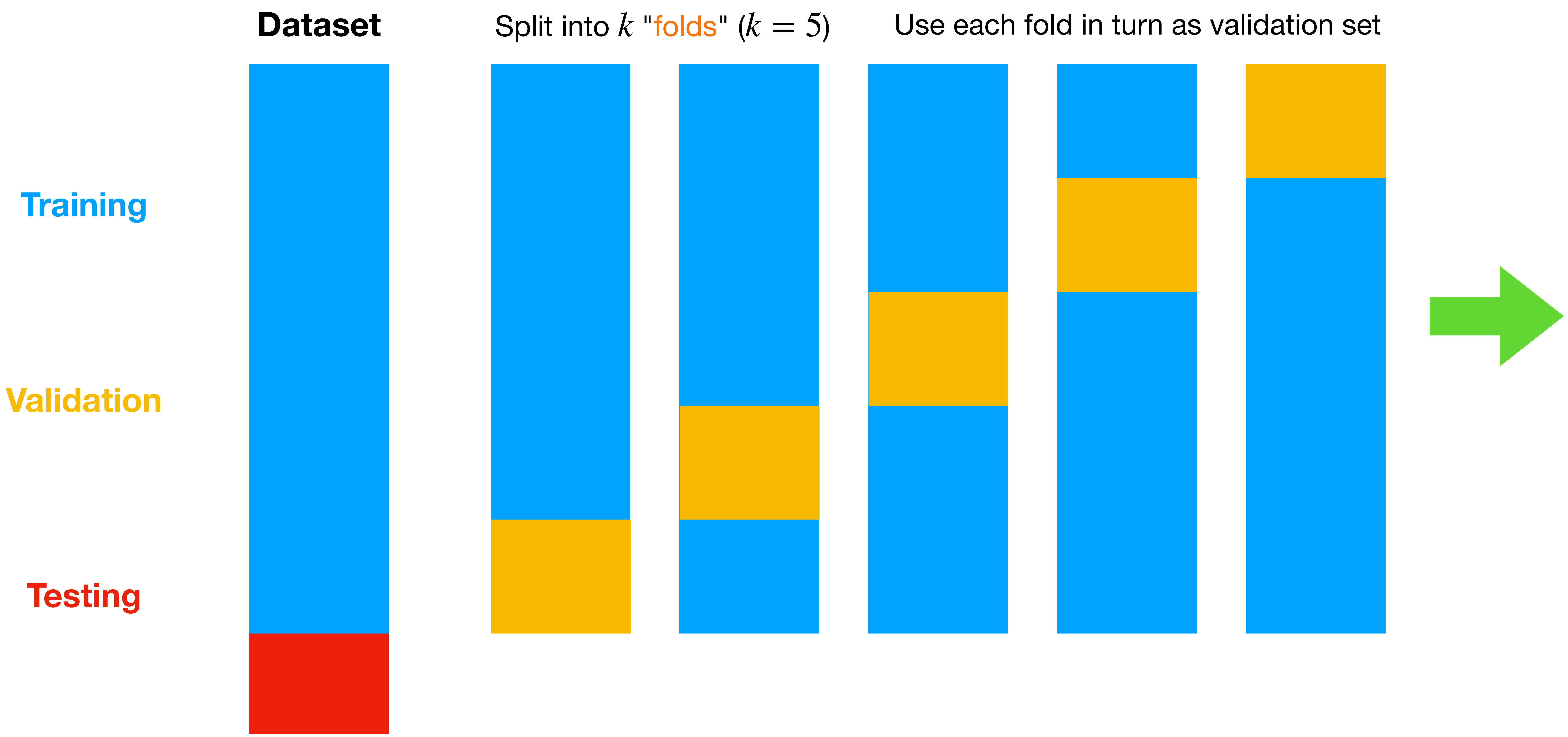
(k-Fold) Cross-validation



(k-Fold) Cross-validation



(k-Fold) Cross-validation



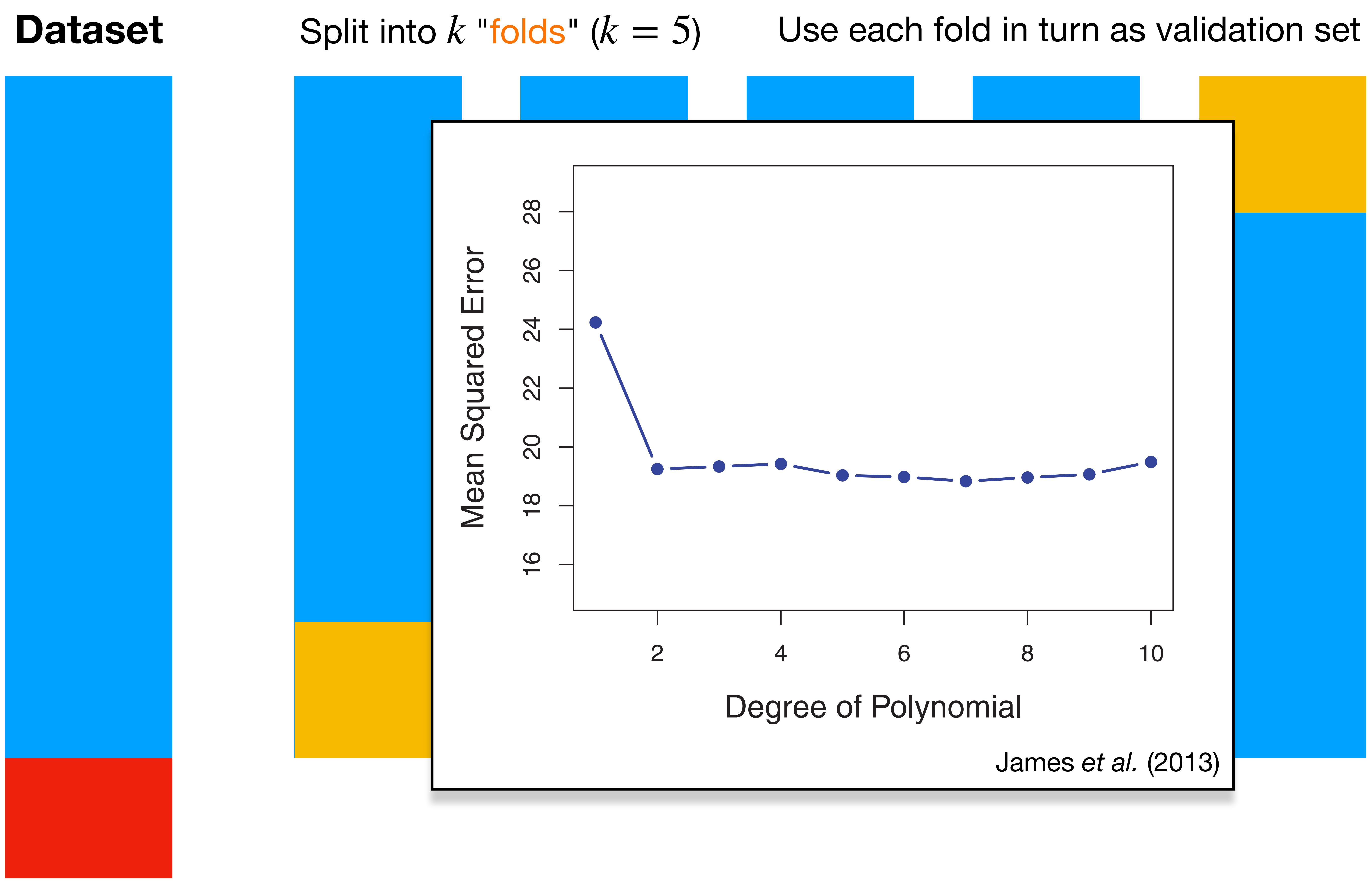
Use measure of performance (error metric?) **averaged over folds** to select best model

(k-Fold) Cross-validation

Training

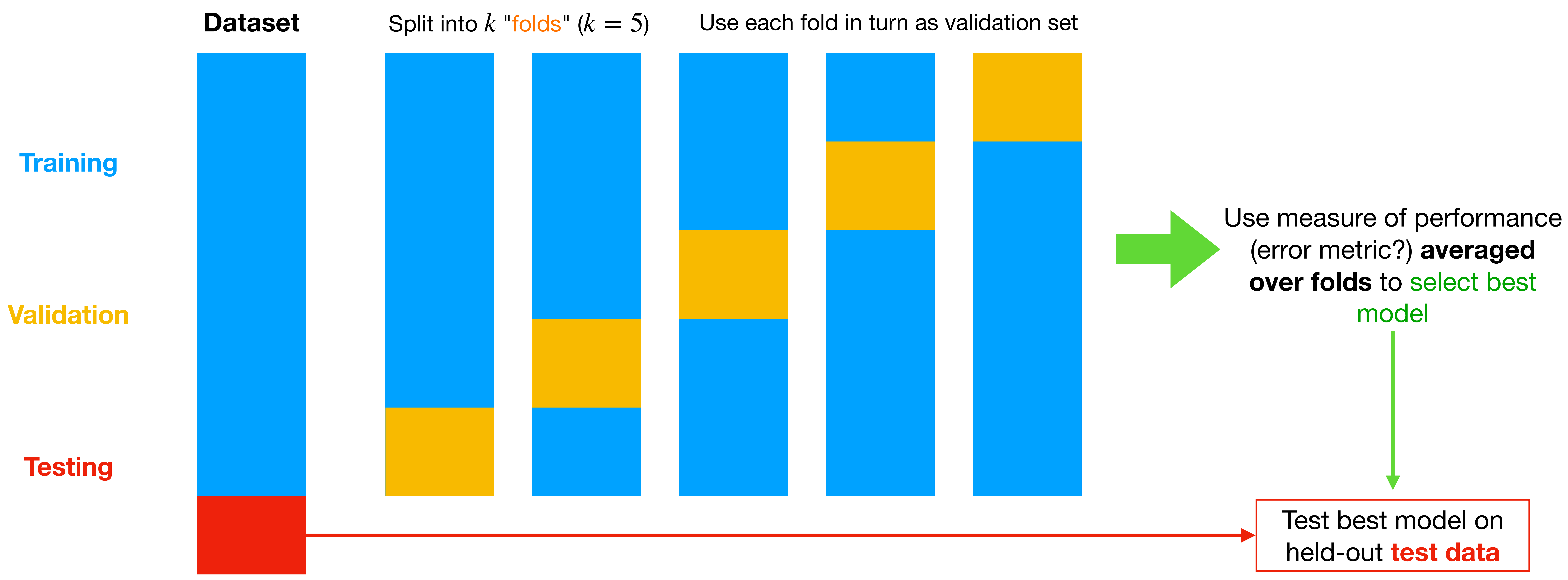
Validation

Testing

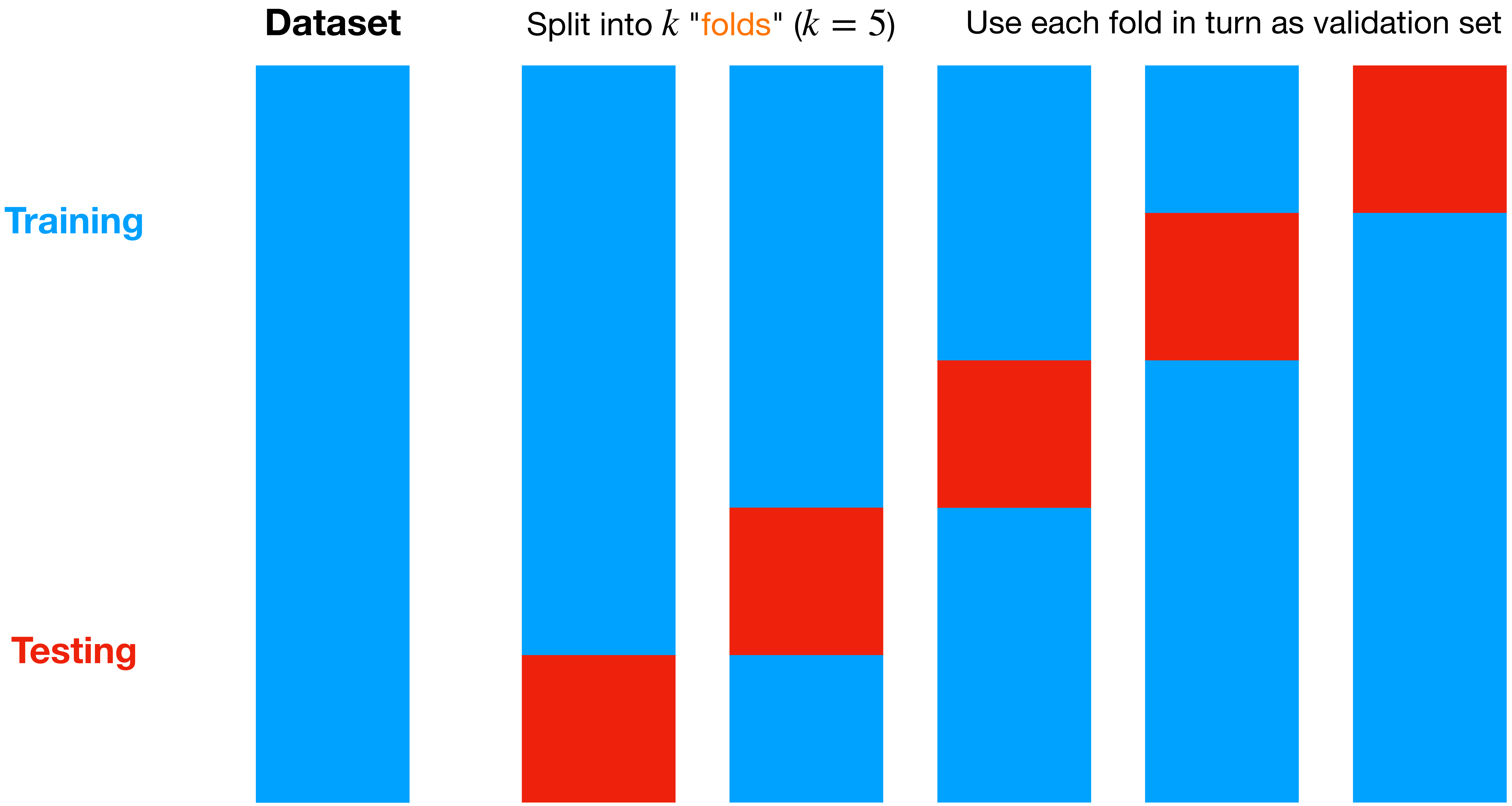


Use measure of performance (error metric?) **averaged over folds** to select best model

(k-Fold) Cross-validation



(k-Fold) Cross-validation



Often common to dispense with the validation split altogether!

(If you don't need to find hyperparameters)

More **robust estimate** of model performance by averaging over **multiple** training/testing splits

Splits/Cross-Validation are biased

Compare

A) Take a single dataset and split it into two datasets

B) Generate a dataset and, later, generate a second dataset

Splits/Cross-Validation are biased

Compare

A) Take a single dataset and split it into two datasets

B) Generate a dataset and, later, generate a second dataset

It's likely there is **unaccounted variation(s)** that is changing the two datasets in option B that is not present in option A

→ Larger difference between the two B datasets than the two A datasets

Splits/Cross-Validation are biased

Compare

A) Take a single dataset and split it into two datasets

B) Generate a dataset and, later, generate a second dataset

It's likely there is **unaccounted variation(s)** that is changing the two datasets in option B that is not present in option A

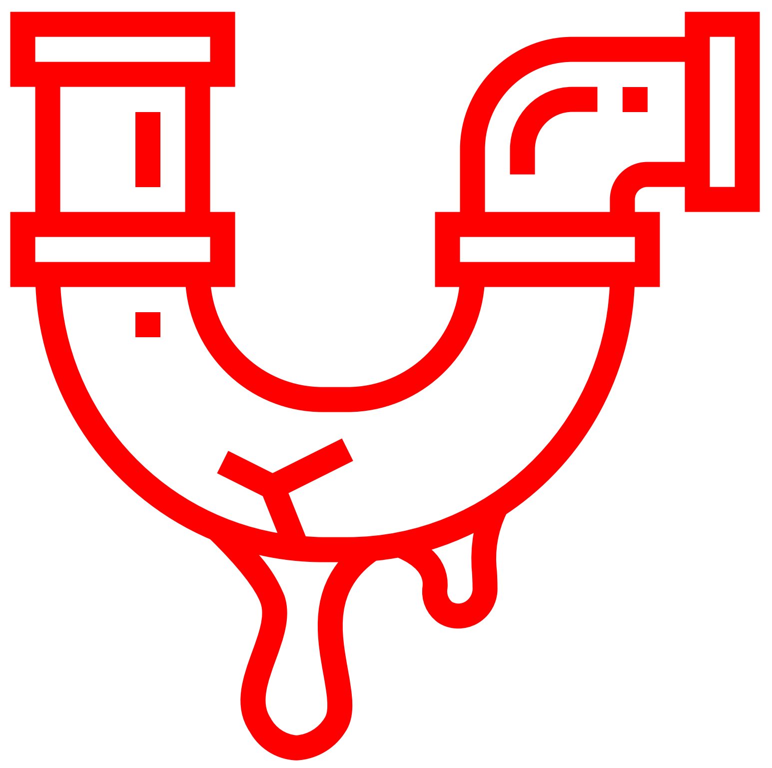
→ Larger difference between the two B datasets than the two A datasets

Therefore, training/testing or cross-validation will likely overestimate the true performance of a predictive model

Underestimates the error of the model

Data Leakage

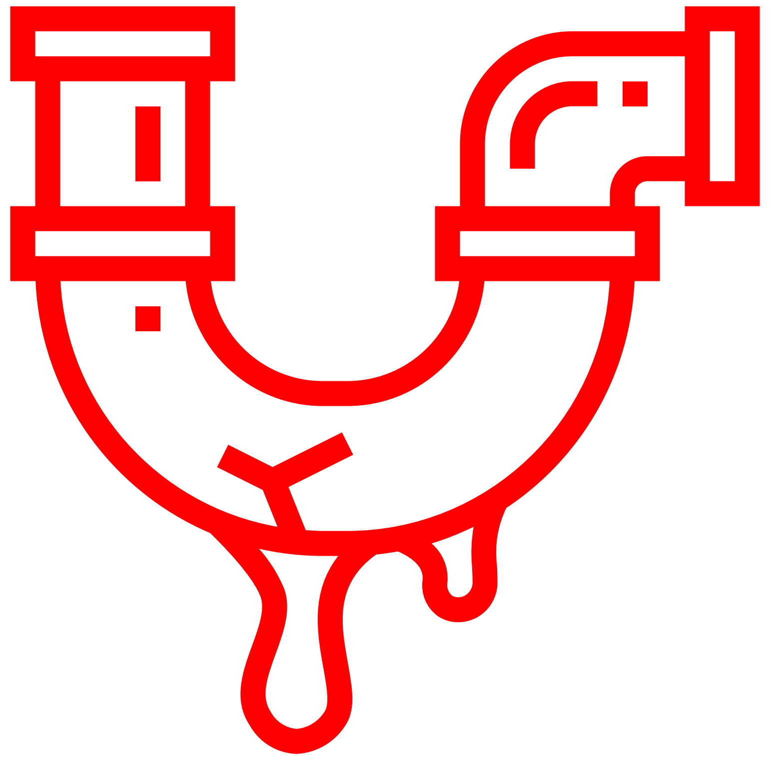
Failure to simulate deployment



Information from the held out test set was used during training

Data Leakage

Failure to simulate deployment



Information from the held out test set was used during training

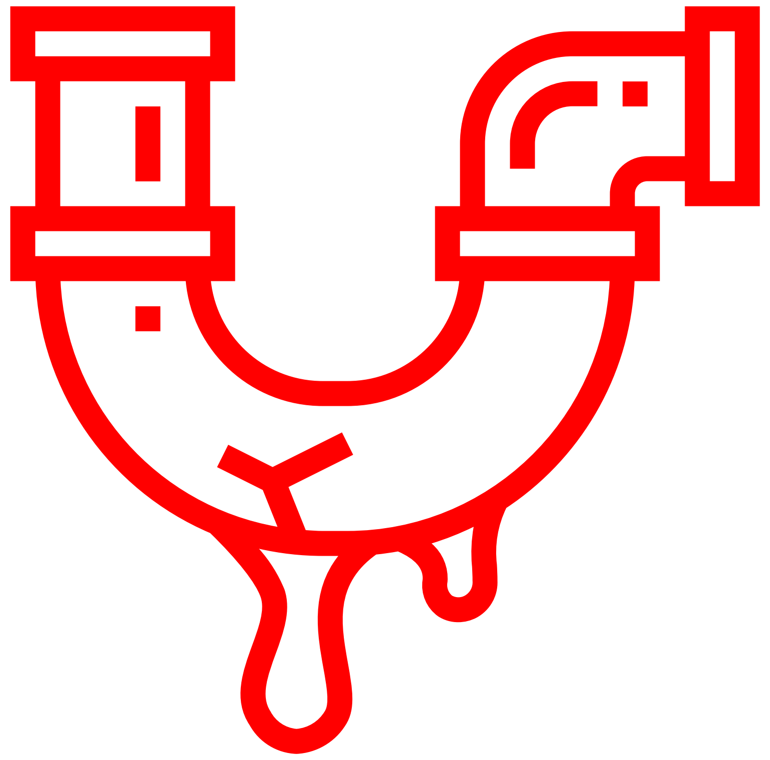
Example

Suppose I want to rescale (one of) my predictive features:

$$X_i \text{ becomes } \frac{X_i - \bar{X}_i}{\sigma_{X_i}}$$

Data Leakage

Failure to simulate deployment

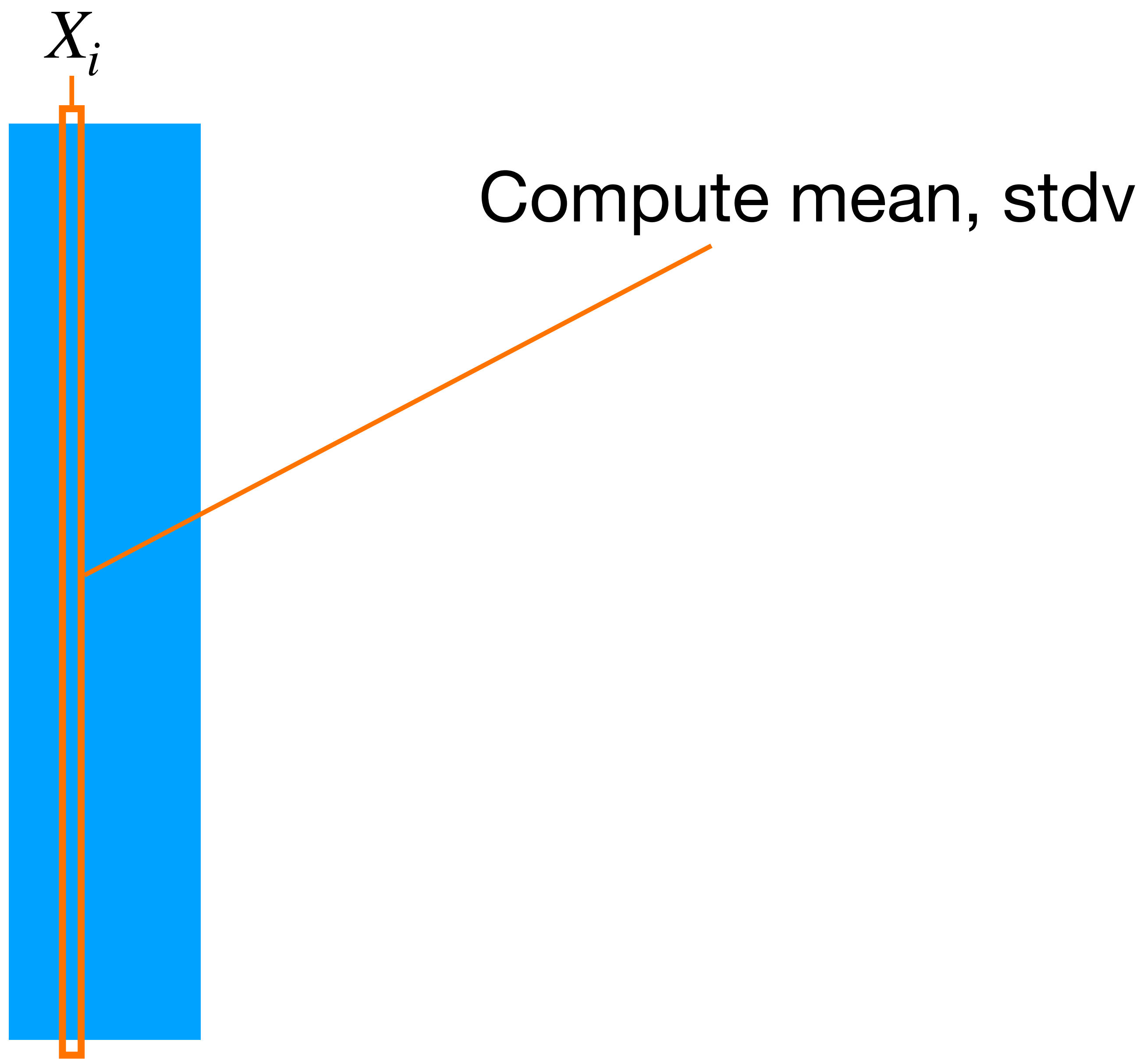


Information from the held out test set was used during training

Example

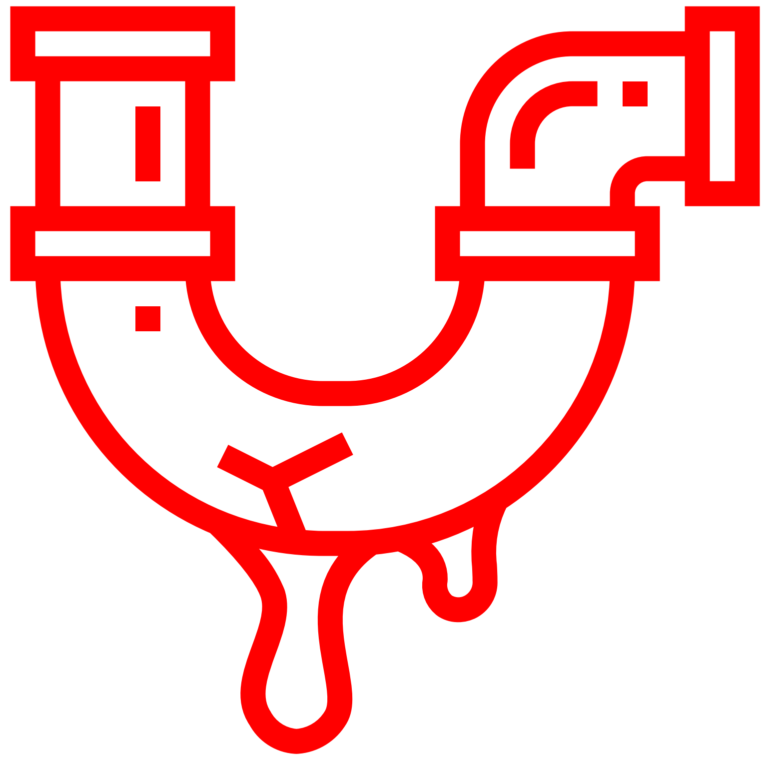
Suppose I want to rescale (one of) my predictive features:

$$X_i \text{ becomes } \frac{X_i - \bar{X}_i}{\sigma_{X_i}}$$



Data Leakage

Failure to simulate deployment

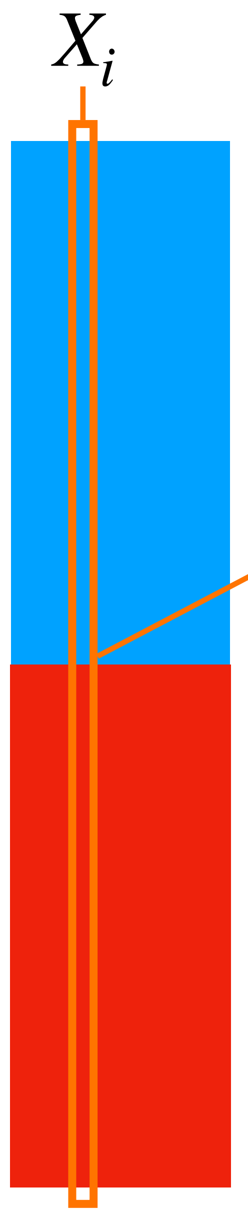


Information from the held out test set was used during training

Example

Suppose I want to rescale (one of) my predictive features:

$$X_i \text{ becomes } \frac{X_i - \bar{X}_i}{\sigma_{X_i}}$$



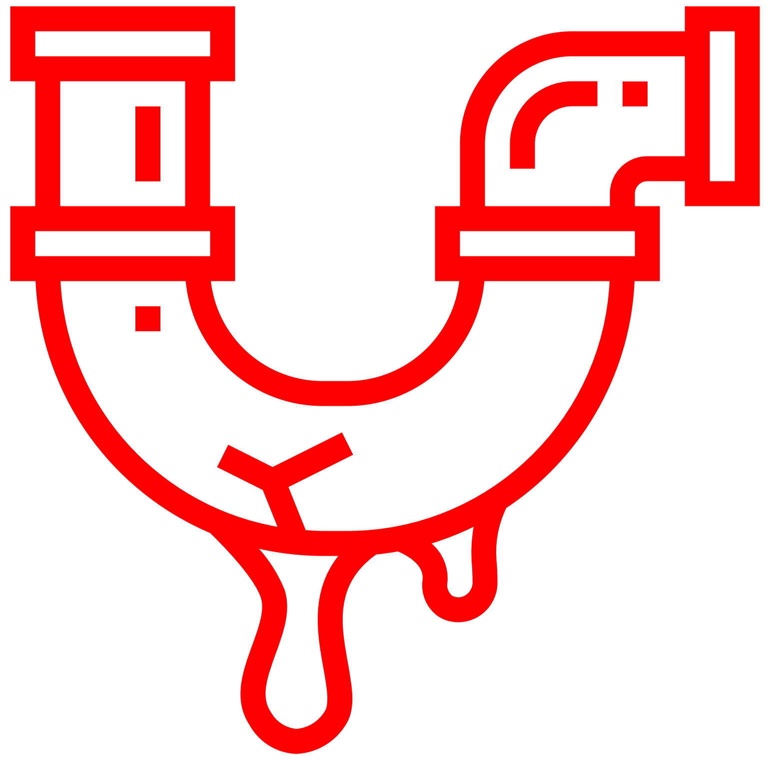
Compute mean, stdv

I haven't split training and testing yet

→ $\bar{X}_i, \sigma_{X_i}$ **used test points**. ***Leakage!***

Data Leakage

Failure to simulate deployment

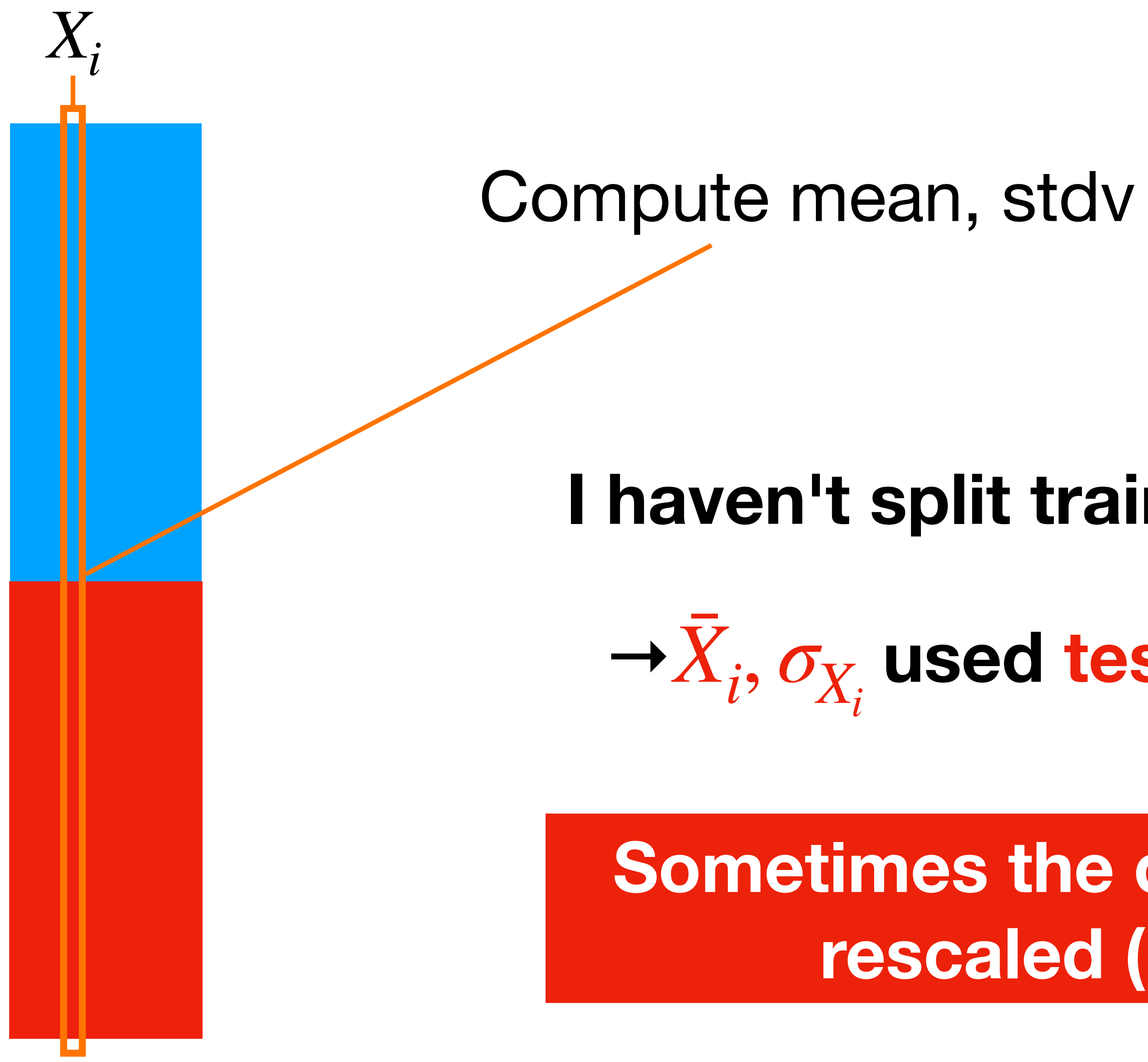


Information from the held out test set was used during training

Example

Suppose I want to rescale (one of) my predictive features:

$$X_i \text{ becomes } \frac{X_i - \bar{X}_i}{\sigma_{X_i}}$$



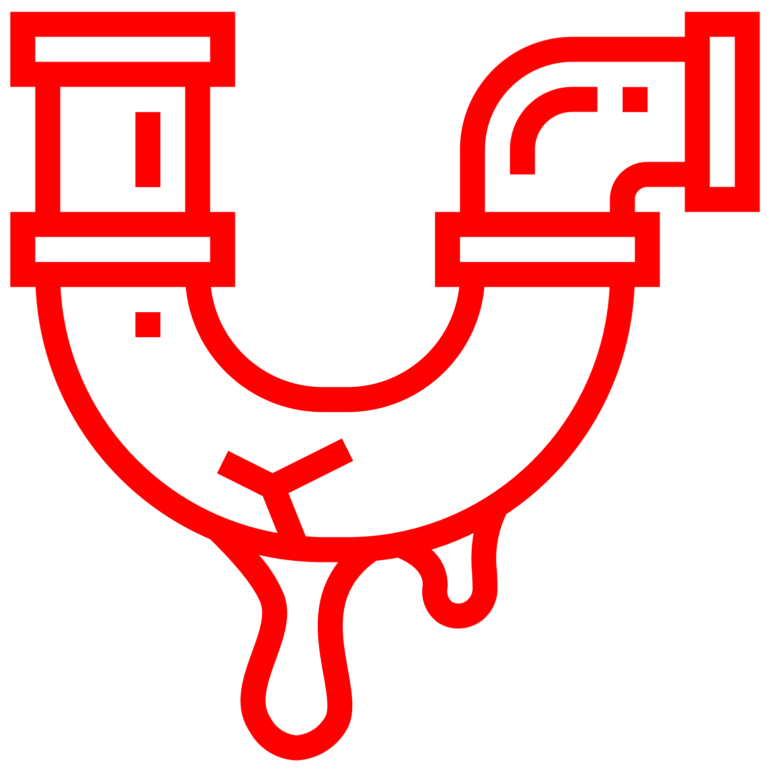
I haven't split training and testing yet

→ $\bar{X}_i, \sigma_{X_i}$ **used test points**. ***Leakage!***

Sometimes the data already come rescaled (pre-leaked!)

Data Leakage

Failure to simulate deployment



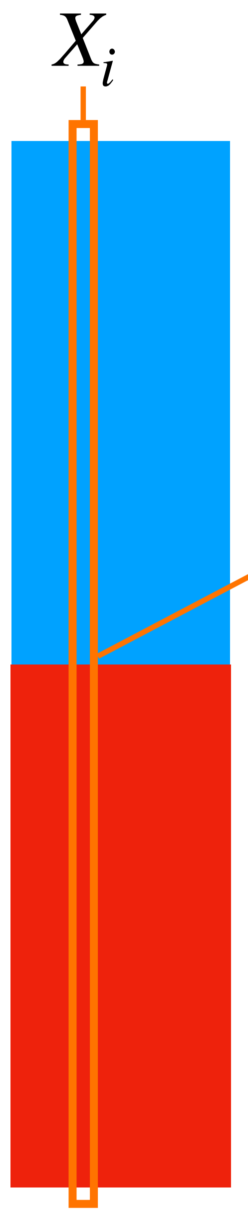
Amazing model performance?
Too-good-to-be-true performance?
Might be leakage!

Information from the held out test set was used during training

Example

Suppose I want to rescale (one of) my predictive features:

$$X_i \text{ becomes } \frac{X_i - \bar{X}_i}{\sigma_{X_i}}$$



Compute mean, stdv

I haven't split training and testing yet

→$\bar{X}_i, \sigma_{X_i}$ used test points. *Leakage!*

Sometimes the data already come rescaled (pre-leaked!)

Summary — Predictive Models

- Prediction vs inference
- Prediction with supervised learning $y = \hat{f}(X)$
  - dominant form of machine learning
  - use pre-existing X, y (*training data*) to figure out $\hat{f}$
- Why build predictive models?
  - X is cheap, y expensive, so use $\hat{f}(X)$ instead of y
- Deployment - how will model work when there is no y?
  - Simulate deployment with cross-validation
  - Take care—data leakage!