Bias/Variance and Cross-Validation

R. Henning's slides
with customization
by F. Burkholder

galvanize

- Review: Linear Regression
- Overfitting and Underfitting
- The Bias/Variance Tradeoff
- Cross-Validation
- K-fold Cross-Validation
- Subset Selection of Predictors

Regression vs. Classification

(in machine learning)

Regression vs. Classification
(in machine learning)

What is regression?

Features (X) predict continuous targets (Y).
E.g. predict future sales/revenue

What is classification?

Features (X) predict categorical targets (Y).
E.g. predict yes/no, male/female, 0-9

Predict placement in horse racing
Regression or classification?

Regression vs. Classification
(in machine learning)

What is regression?

Features (X) predict continuous targets (Y).
E.g. predict future sales/revenue

What is classification?

Features (X) predict categorical targets (Y).
E.g. predict yes/no, male/female, 0-9

In either case, we want to predict a result given new data.

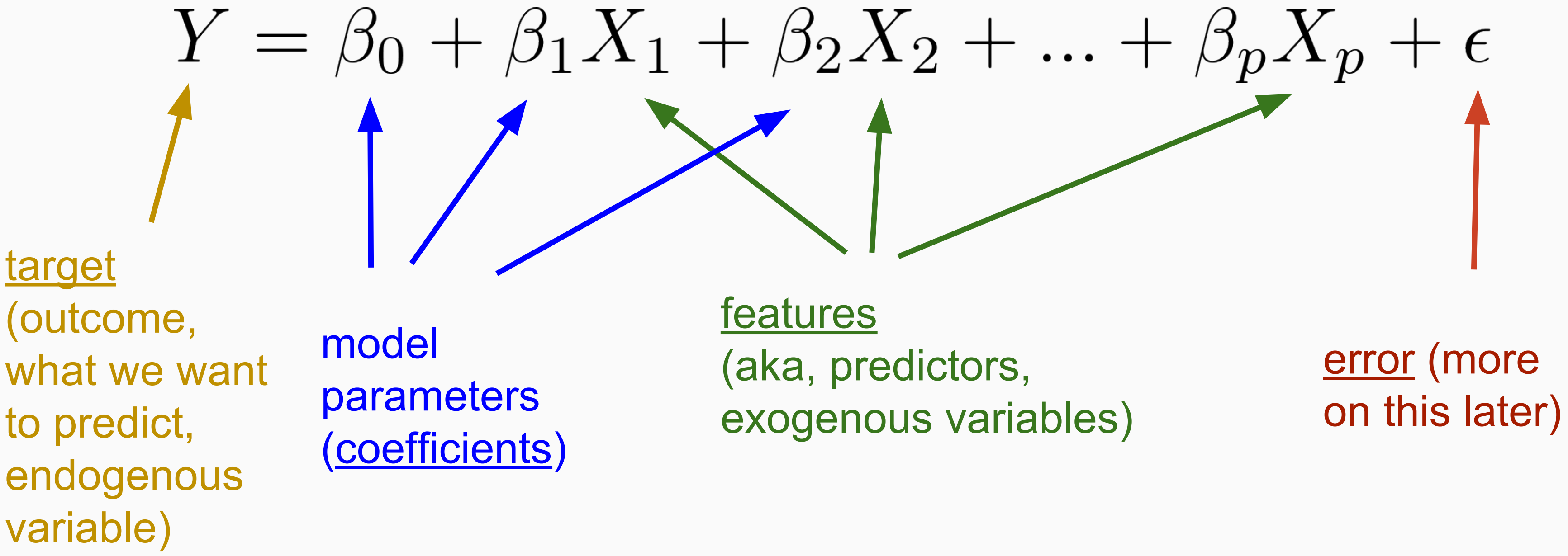
Something we've learned that can make predictions: Linear Regression

We assume the world is built on linear relationships. Under that assumption, we can model the relationship between *features* and a *target* like this:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

X_1	X_2	X_3	X_4	Y
0	1.3	2.6	Male	12.1
1	-3.2	-6.4	Female	13.7

Review: Linear Regression



$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

target (outcome, what we want to predict, endogenous variable)

model parameters (coefficients)

features (aka, predictors, exogenous variables)

error (more on this later)

How good is your regression?

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

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

$$\mathrm{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

where

$\hat{y}_i$ is the predicted value for datapoint i

$\bar{y}$ is the mean of all the y_i's

RSS is the *residual sum of squares*.
TSS is the *total sum of squares*, which is the variance of y.

To penalize for model complexity:

$$\text{Adjusted } R^2 = 1 - \frac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}$$

d = number of features kept

We also will sometimes use MSE, which the *mean squared error*.

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

where n is the number of datapoints

RMSE = sqrt(MSE) (Advantage over MSE?)

Other metrics that penalize for model complexity:

$$\mathrm{AIC} = \frac{1}{n}\left(RSS + 2d\hat{\sigma}^2\right)$$

$$\mathrm{BIC} = \frac{1}{n}\left(RSS + \log(n)d\hat{\sigma}^2\right)$$

where

σ^2 = variance of the residuals

d = number of features kept

Is R^2 (or adjusted R^2) all that matters?

We *could* just keep inserting interaction features until R^2 = 1.

Boom. Data science has been solved. Here’s the idea:

```
def train_super_awesome_perfect_model (X, y):
    while True:
        model = LinearRegression()
        model.fit(X, y)
        if calculate_r2(model, X, y) >= 0.999:
            return model
        else:
            X = insert_random_interaction_feature(X)
```

Well?

Our fit on existing data is not all that matters

We care about how our model performs on new data, right? So do we really care if we have an amazing R^2 or adjusted R^2 on the data we trained our model on?

But for argument's sake, let's say we could get a lot of data from all over the feature space to train our model on. Then it would be safe to train a very complicated model on all that data to get a high R^2 that will surely predict well on new data, right?

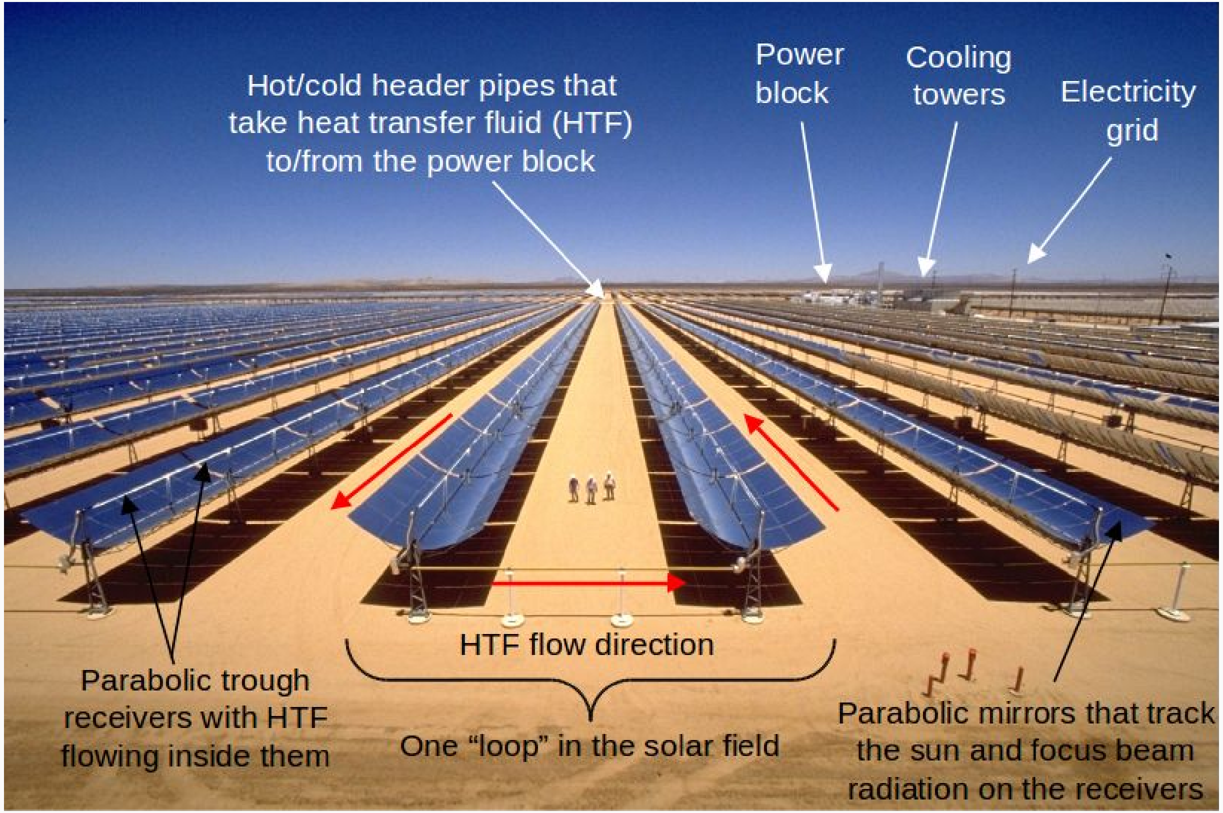
Our fit on existing data is not all that matters

We care about how our model performs on new data, right? So do we really care if we have an amazing R^2 or adjusted R^2 on the data we trained our model on?

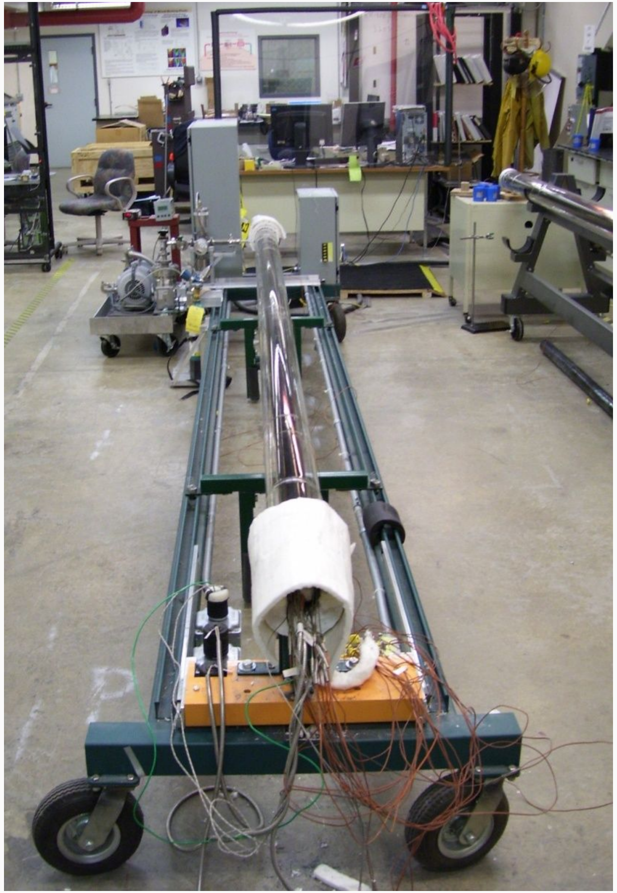
But for argument's sake, let's say we could get a lot of data from all over the feature space to train our model on. Then it would be safe to train a very complicated model on all that data to get a high R^2 that will surely predict well on new data, right?

No, and no, and let me describe via a tangent.

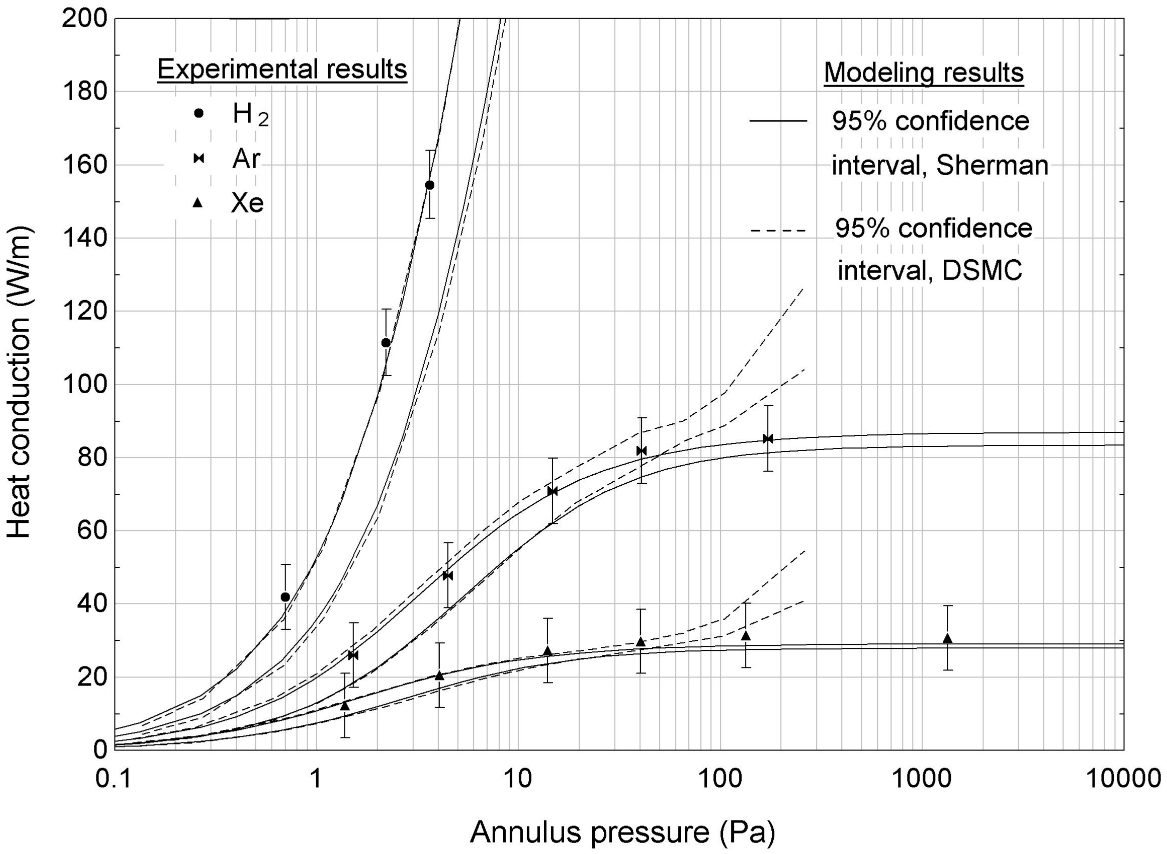
Real life tangent - signal and noise (and my Ph.D.)



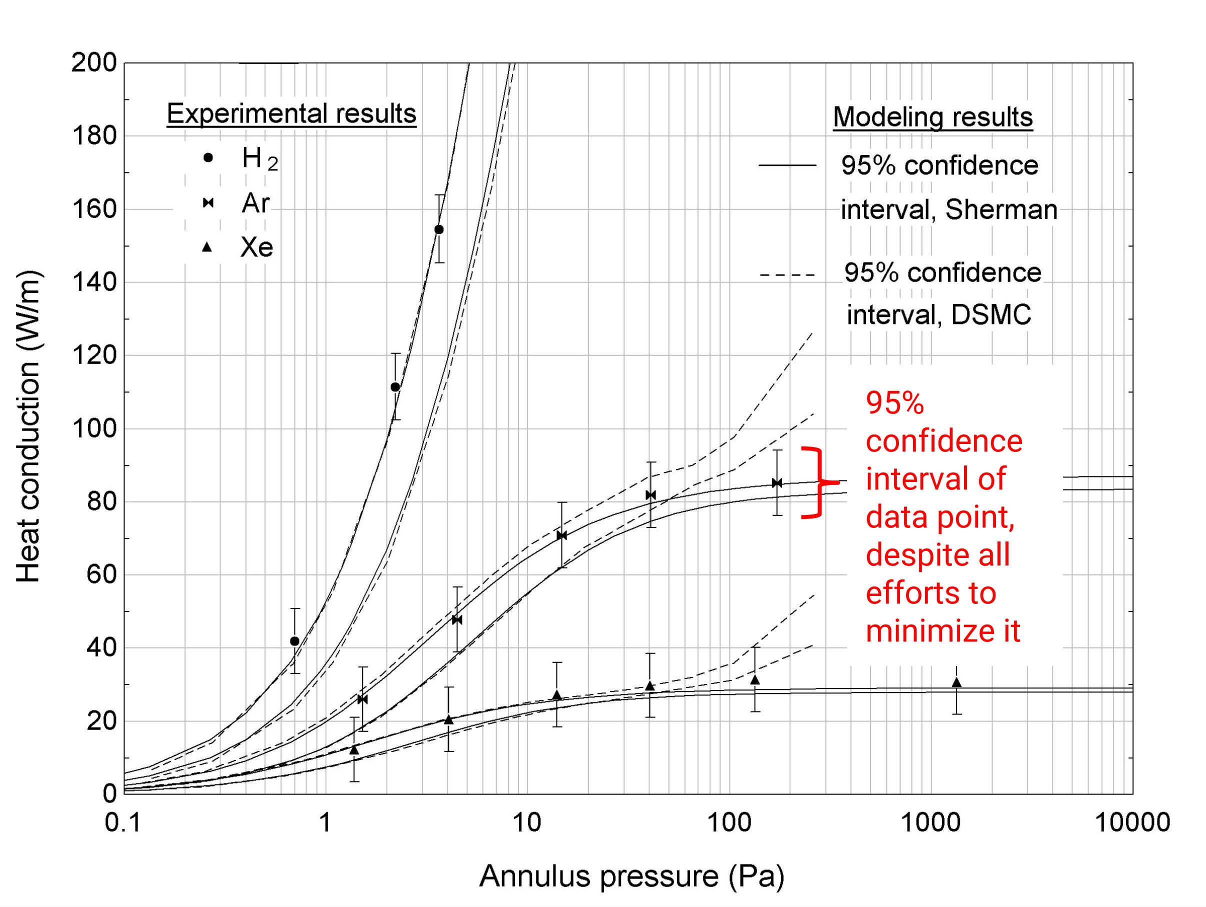
Hot/cold header pipes that take heat transfer fluid (HTF) to/from the power block
Power block
Cooling towers
Electricity grid
HTF flow direction
Parabolic trough receivers with HTF flowing inside them
One “loop” in the solar field
Parabolic mirrors that track the sun and focus beam radiation on the receivers



Real life - signal and noise - results in a controlled environment



Real life - can’t get rid of the noise



Most data is a combination of:

- signal (that we want to model)
- random noise (that we don’t)

BAD IDEA

We *could* just keep inserting interaction features until $R^2 = 1$.

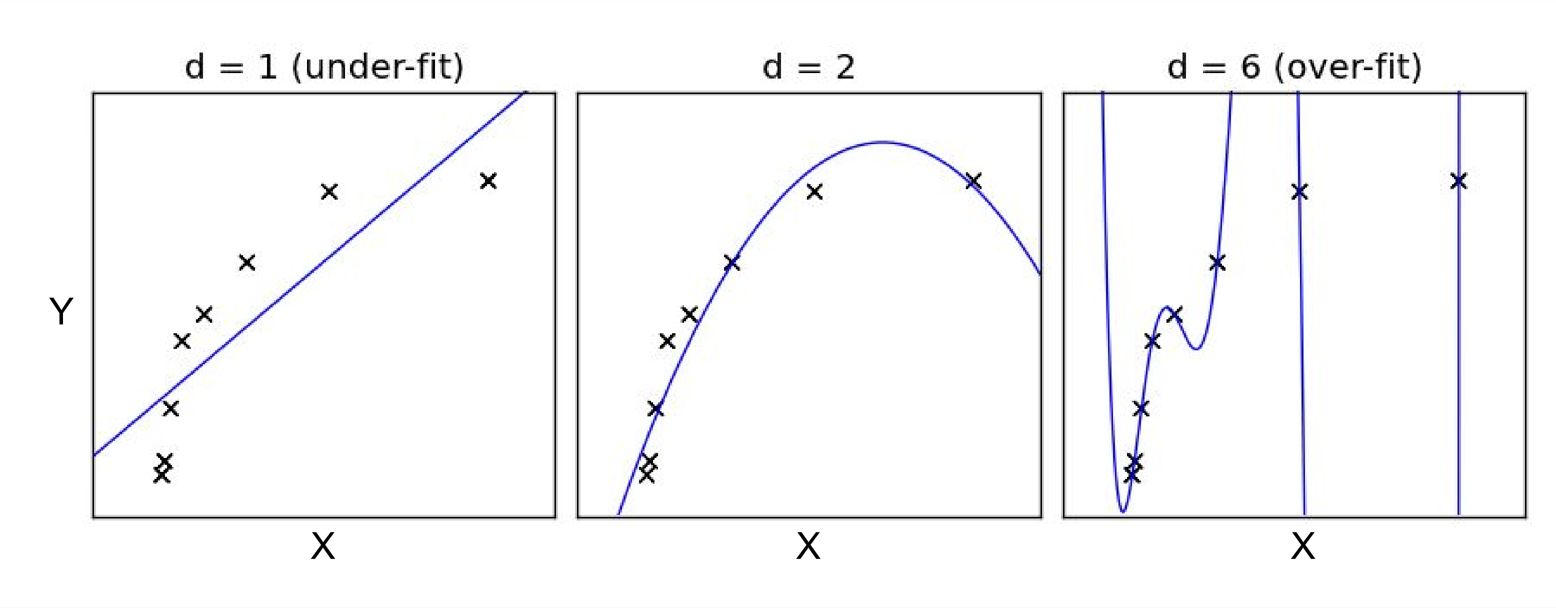
Boom. Data science has been solved. Here’s the idea:

```
def train_super_awesome_perfect_model (X, y):
    while True:
        model = LinearRegression()
        model.fit(X, y)
        if calculate_r2(model, X, y) >= 0.999:
            return model
        else:
            X = insert_random_interaction_feature(X)
```

Bad idea, because:
1) good performance on *new* data is the goal.

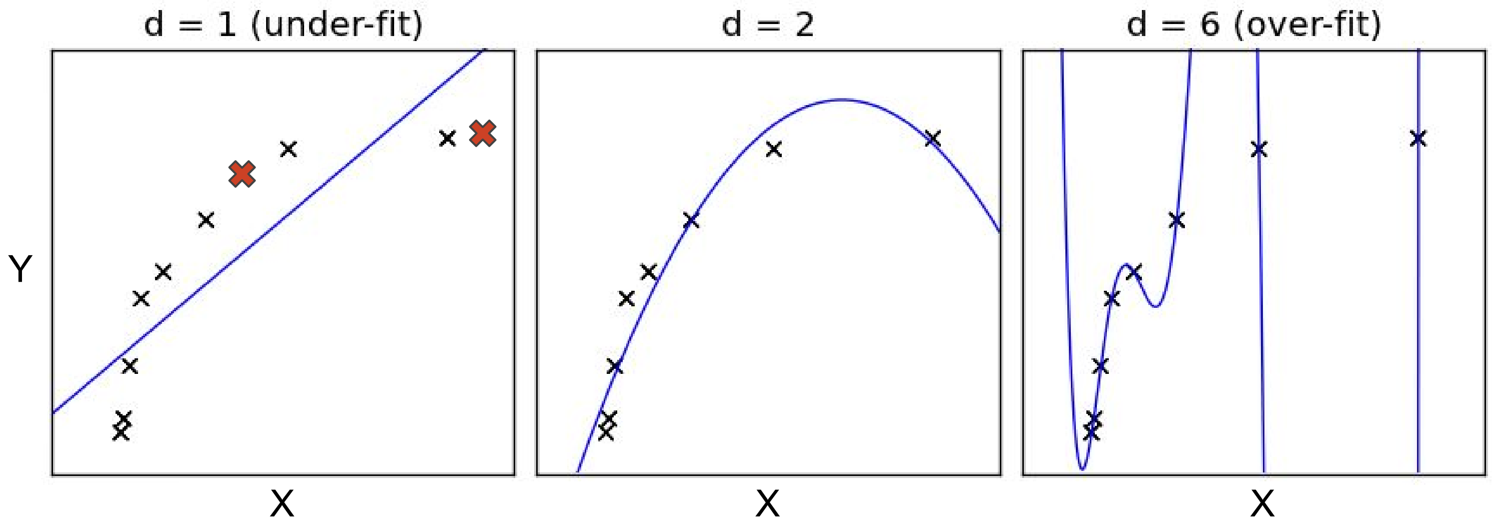
2) to get such great correlation on your training set, you likely fit the signal *and* the noise (you overfit).

The morale of the story: there is signal *and* noise in data, so how should we fit?



What's bad about the first (left) model?

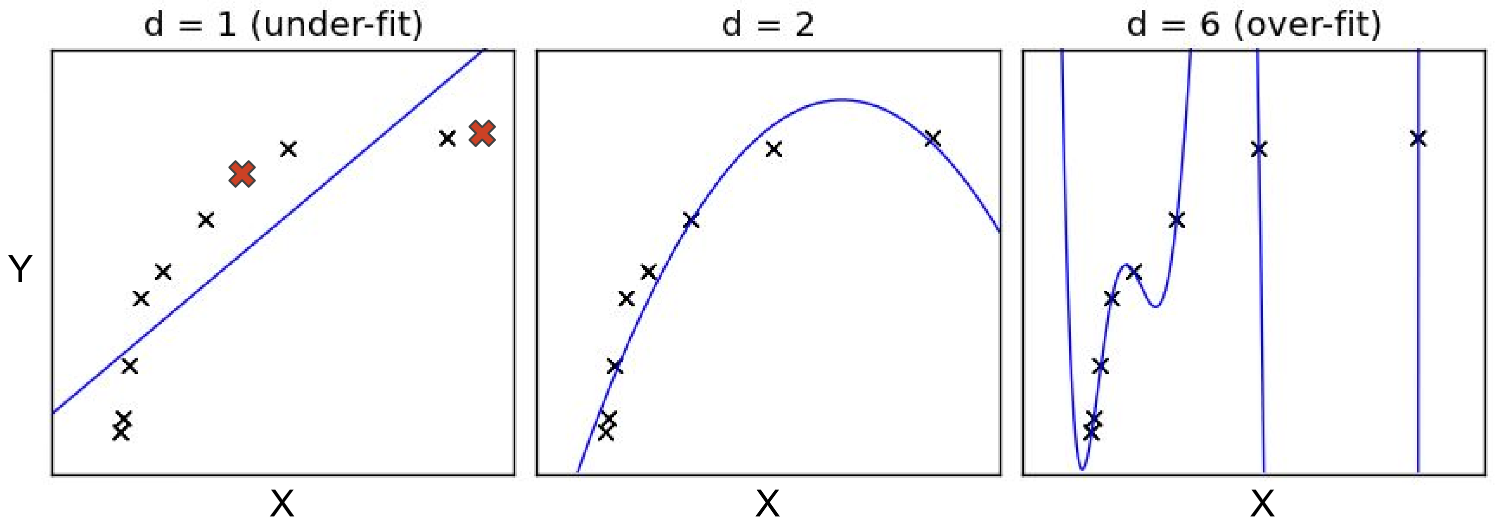
The morale of the story: there is signal *and* noise in data, so how should we fit?



✖ = new data from underlying signal

What's bad about the first (left) model?

The morale of the story: there is signal *and* noise in data, so how should we fit?

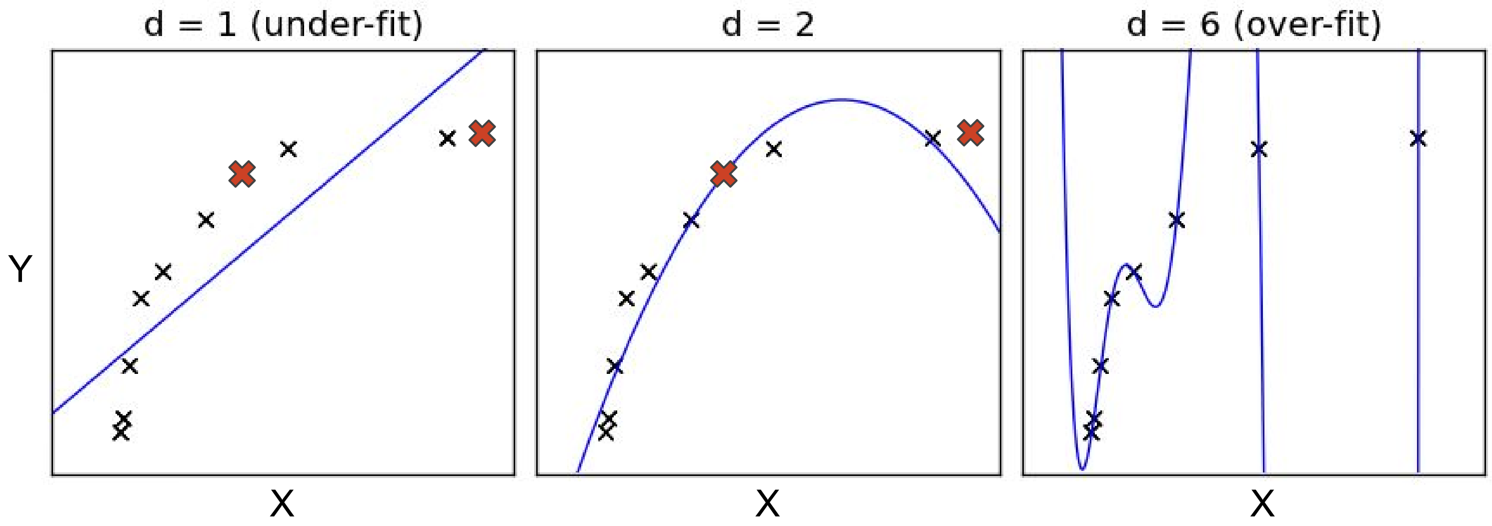


✖ = new data from underlying signal

What's bad about the first (left) model?

What's bad about the second model?

The morale of the story: there is signal *and* noise in data, so how should we fit?

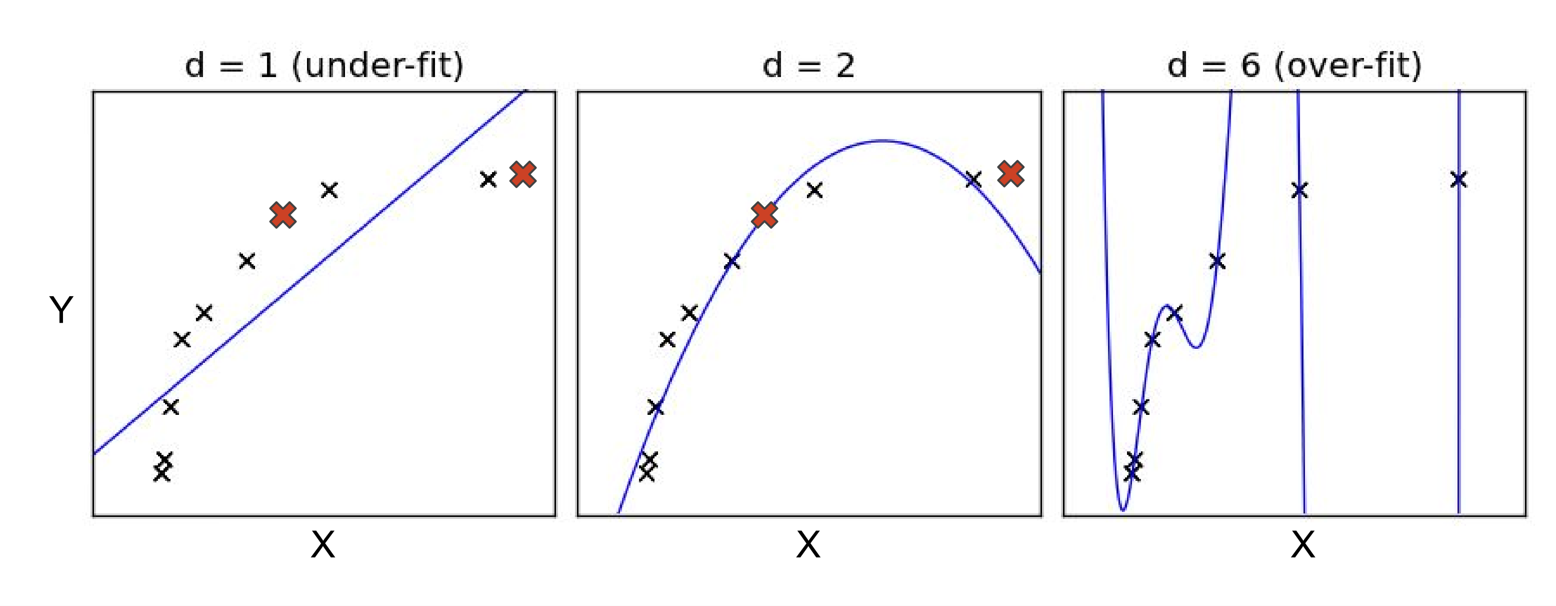


✖ = new data from underlying signal

What's bad about the first (left) model?

What's bad about the second model?

The morale of the story: there is signal *and* noise in data, so how should we fit?

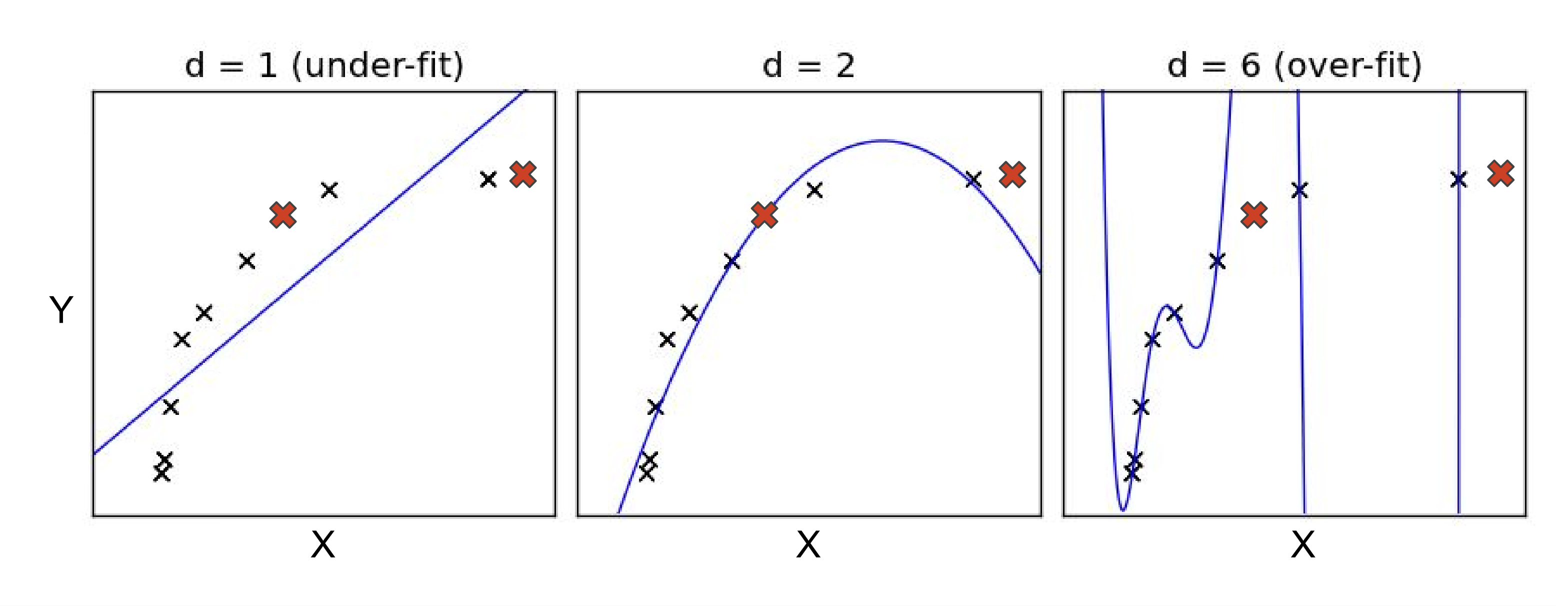


What's bad about the first (left) model?

What's bad about the second model?

What's bad about the third model?

The morale of the story: there is signal *and* noise in data, so how should we fit?



What's bad about the first model?

What's bad about the second model?

What's bad about the third model?

Underfitting and Overfitting

Underfitting: The model doesn't fully capture the relationship between predictors and the target. The model has *not* learned the data's underlying signal.

Overfitting: The model has tried to capture the sampling error. The model has learned the data's signal *and* the noise.

Evidence on the impact of sustained exposure to air pollution on life expectancy from China's Huai River policy

12936–12941 | PNAS | August 6, 2013 | vol. 110 | no. 32

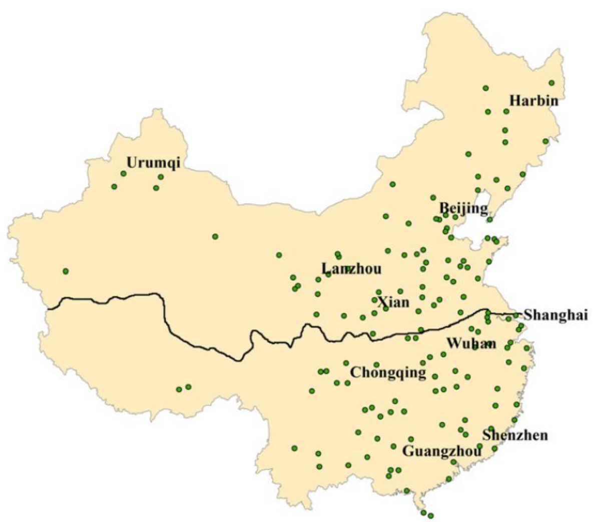
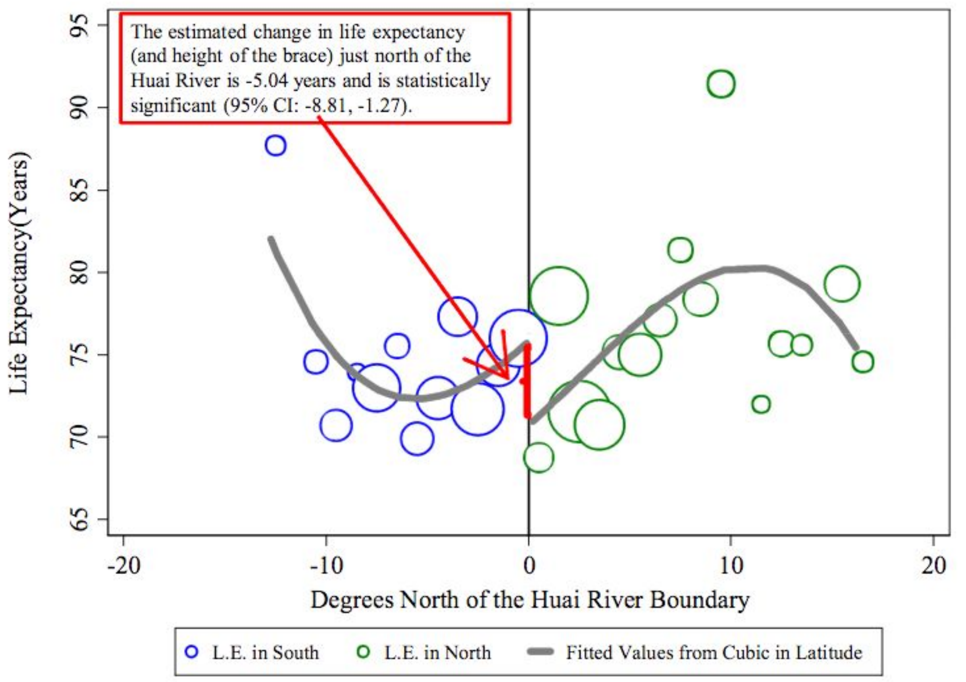


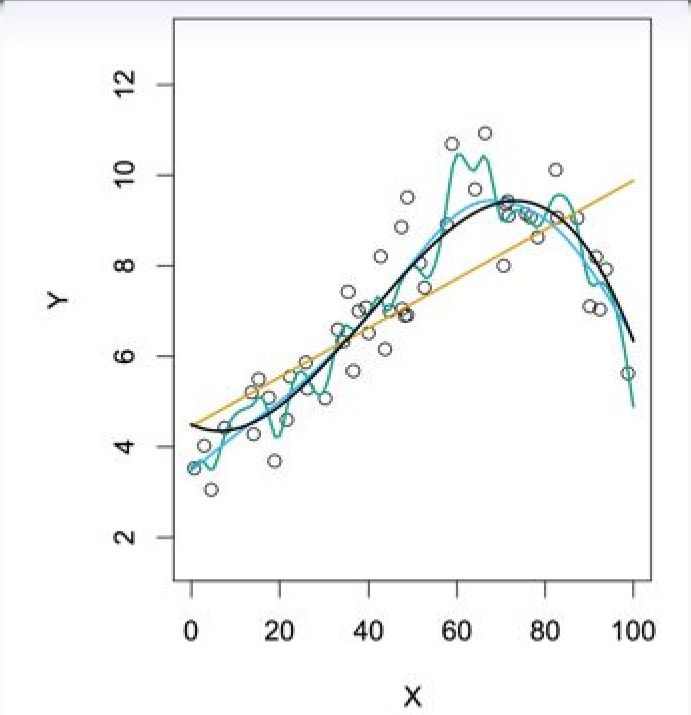
Fig. 1. The cities shown are the locations of the Disease Surveillance Points. Cities north of the solid line were covered by the home heating policy.


The estimated change in life expectancy (and height of the brace) just north of the Huai River is -5.04 years and is statistically significant (95% CI: -8.81, -1.27).

In groups of four at your table, discuss: underfit, properly fit, overfit? Why?

The Bias/Variance Tradeoff

Note: **Bias** and **Variance** are terms you will use A TON as a data scientist! Exciting times!



The Bias/Variance Tradeoff

We assume the true predictor/target relationship is given by an unknown function plus some sampling error:

$$Y = f(X) + \epsilon$$

We estimate the true (unknown) function by fitting a model over the training set.

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

Let's evaluate this model using a test observation **(x_0, y_0)** drawn from the population. What is the model's expected squared prediction error on this test observation?

$$E[(y_o - \hat{f}(x_0))^2] = \ldots$$

The Bias/Variance Tradeoff

Our model's expected squared prediction error will depend on (1) the variability of **y_0** and (2) the variability of the training set used to train our model. We can break this into three pieces:

$$E[(y_o - \hat{f}(x_0))^2] = ... = \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\epsilon)$$

The variance of our model's prediction of **y_0** over all possible training sets

The difference between the true target and our model's average prediction over all possible training sets

The variance of the irreducible error.

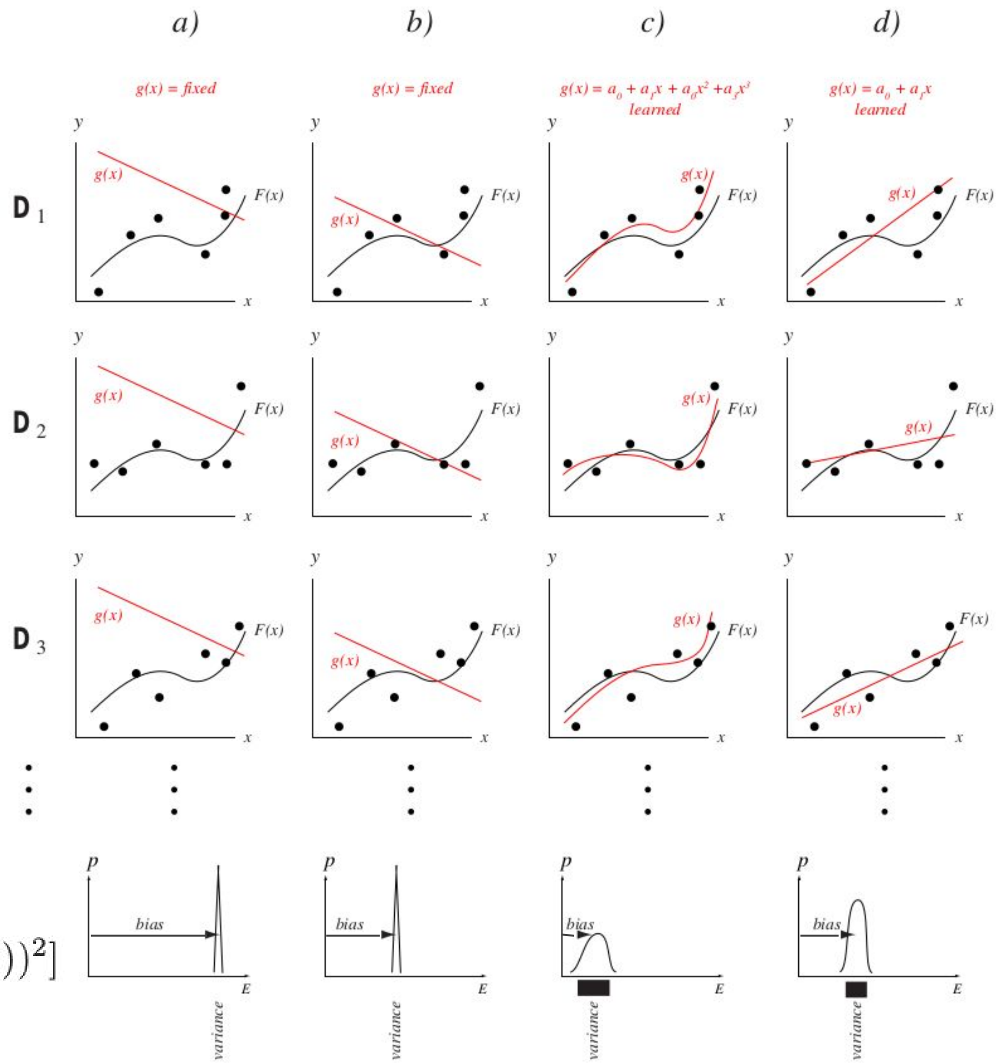
$$\mathrm{Bias}(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$$

Graphical explanation of high/low bias & high/low variance & trade-off

g(x) is the fit

F(x) is the true signal (that we don't know)

The data points are drawn from F(x) and include noise.



MSE histograms

$$E \equiv \mathcal{E}_{\mathcal{D}}[(g(x) - F(x))^2]$$

Verbalizing bias, variance, and bias - variance trade-off

A low bias model accurately predicts the population’s underlying signal (and vice - versa).

A low variance model’s predictions doesn’t change much when it is fit on different data from the underlying population (and vice-versa).

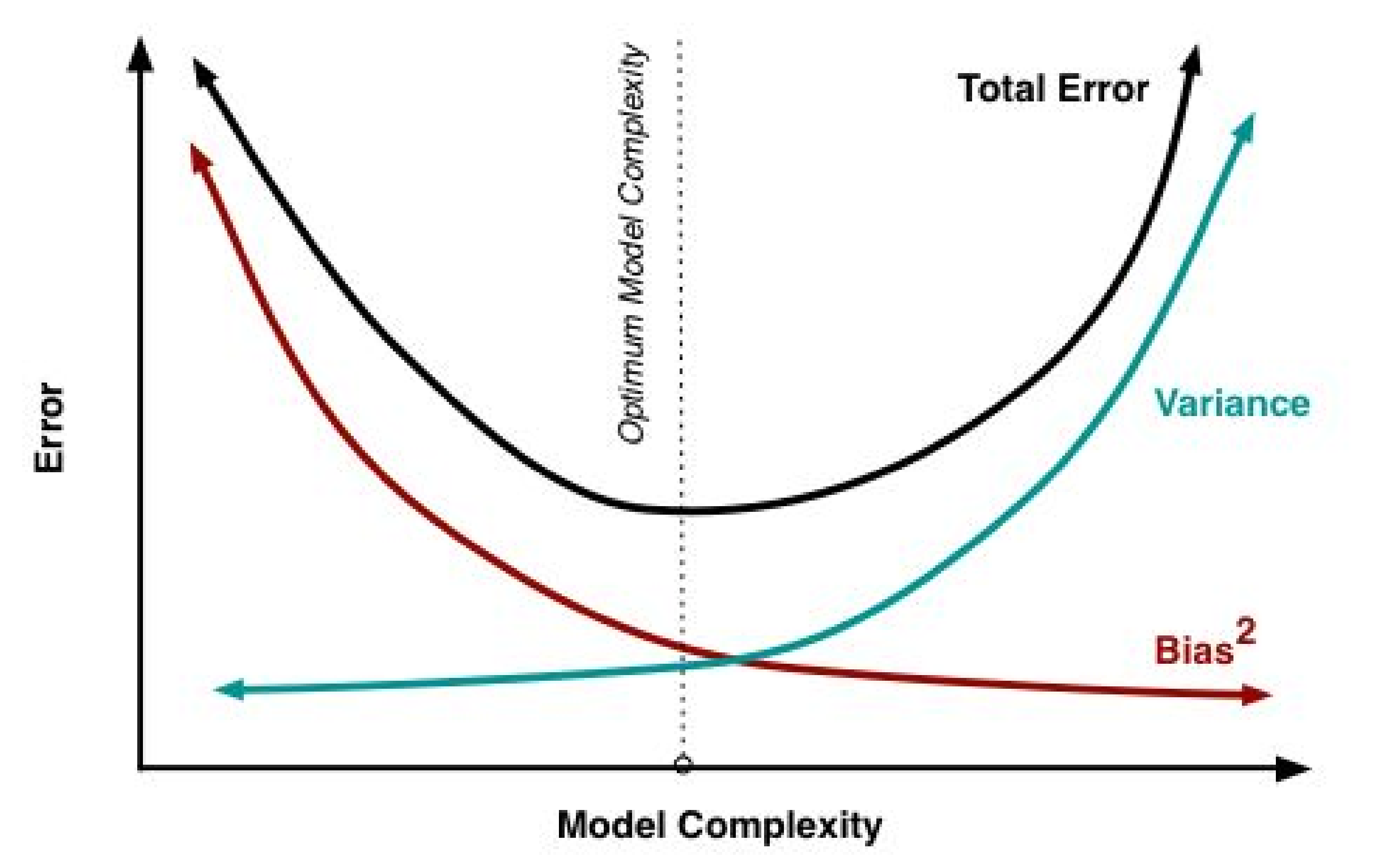
A trade-off often exists between bias and variance because some amount of model complexity is often required to match the underlying population signal, but this same complexity also makes the fit more sensitive to variations in the data the model is fit on. So as bias decreases, variance often increases (and vice-versa).

Bias-variance trade-off questions

1) As makers of predictive models, what is most often our goal?
   a) High bias - high variance models
   b) Low bias - low variance models
   c) High bias - low variance models
   d) Low bias - high variance models
2) A classmate comes up to you and says "A high variance model is simply overfitting, and a high bias model is just underfit." What do you think?
3) Explain, in your own words and without looking back a slide, why there is a bias-variance trade-off.

Think about these for a couple of minutes, pick a partner, and then exchange Slack messages answering them.

Problem: With Bias-Variance tradeoff, how to pick best model?



How can we find the best tradeoff point?

i.e. The optimum model complexity

Cross-Validation

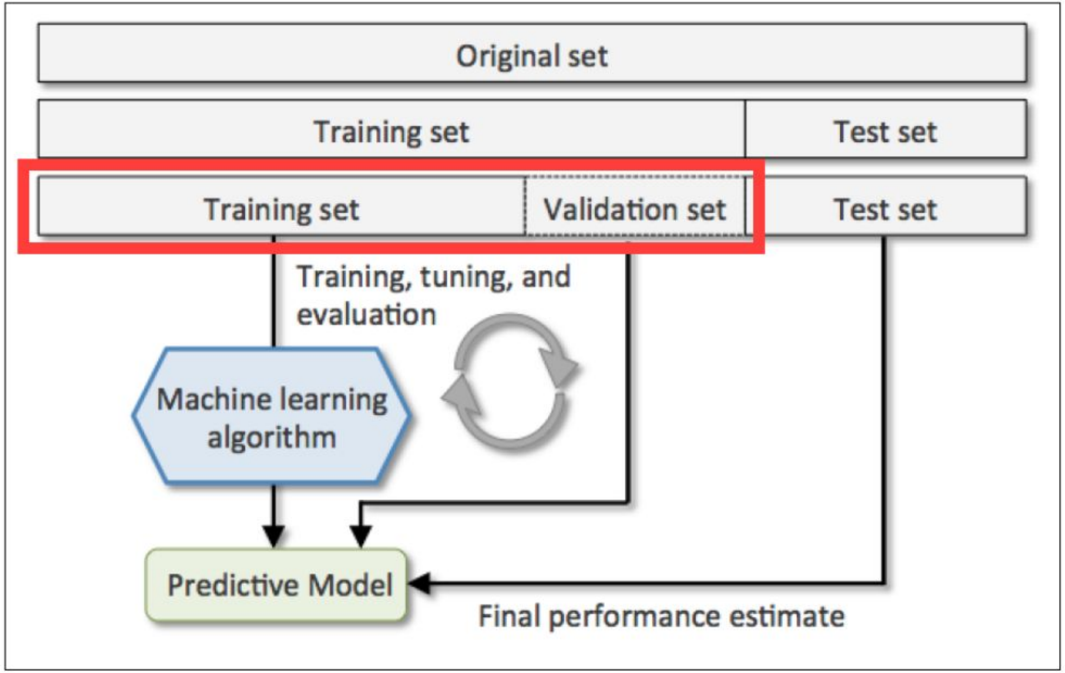
“Cross-validation ... is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.”
- Wikipedia

We use cross-validation for two things:

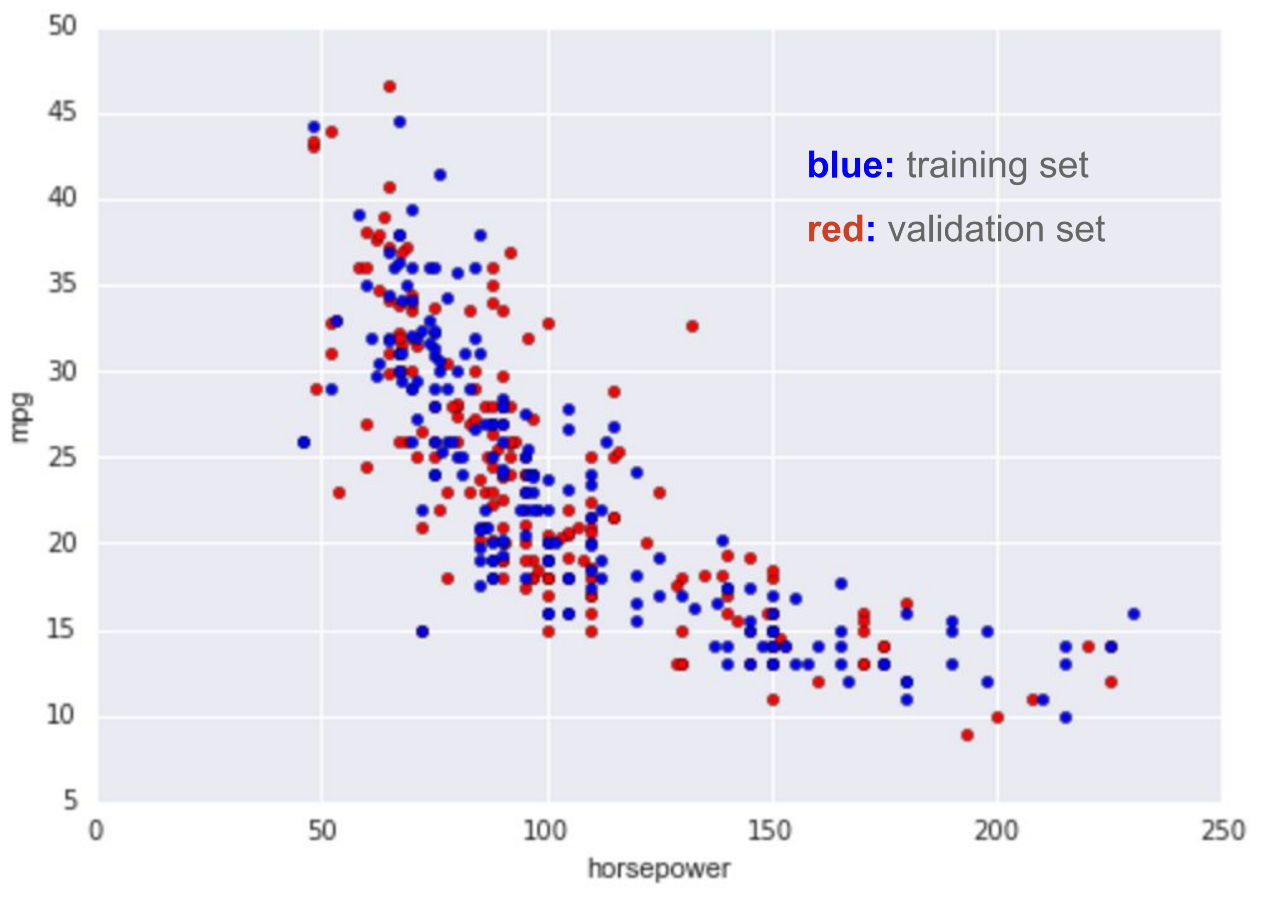
1) Attempting to quantify how well a model (of some given complexity) will predict on an unseen data set
2) Tuning hyperparameters of models to get best predictions.

Scikit-learn tangent (hyperparameters?)

Cross-Validation


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

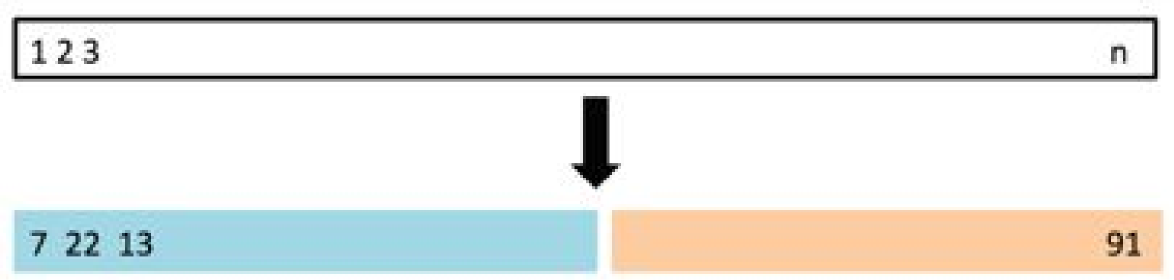
Let’s predict MPG from horsepower



Cross-Validation

Main idea: **Don't use all your data for training.**

Instead: **Split your data into a "training set" and a "validation set".**

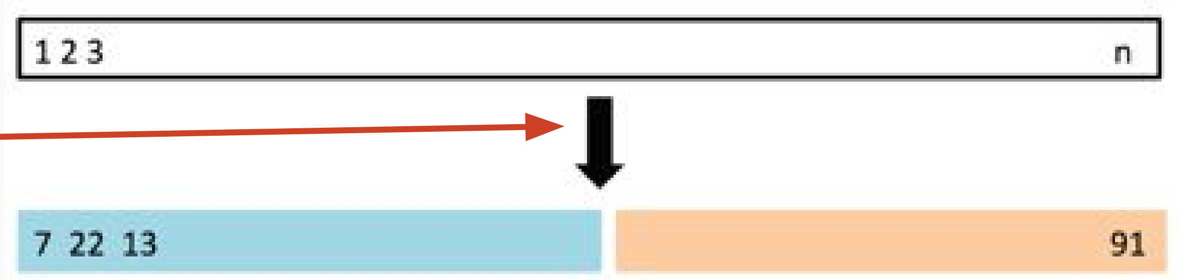


galvanize

Cross-Validation

Main idea: **Don’t use all your data for training.**

Instead: **Split your data into a “training set” and a “validation set”.**



randomize!
(why?)

training set

where to split?

validation set

Cross-Validation

1. Split your data into training/validation sets.

    70/30 or 90/10 splits are commonly used

2. Use the training set to train several models of varying complexity.

    e.g. linear regression (w/ and w/out interaction features), neural nets, decision trees, etc.

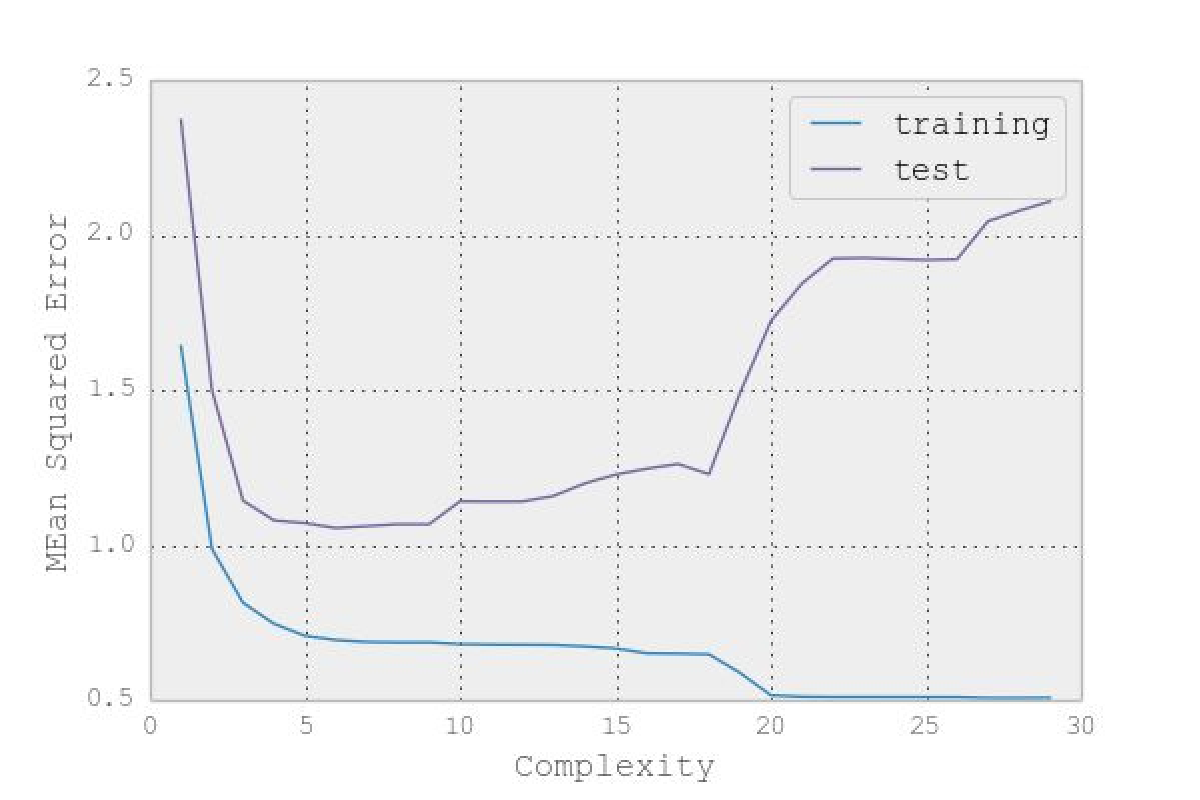
3. Evaluate each model using the validation set.

    calculate R^2, MSE, accuracy, or whatever you think is best

4. Keep the model that performs best over the **validation** set.

Cross-Validation Example

galvanize



Number of features, interaction between features, order of features

Which error is most important to minimize? Why?

What model complexity is best? How did you decide?

At the optimum model complexity, what is the bias? What is the variance?

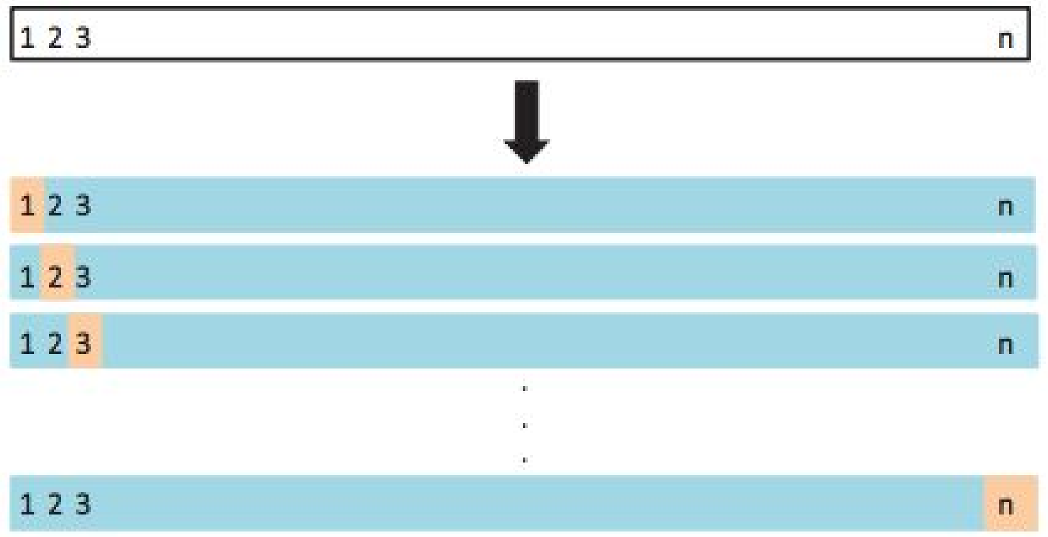
Is it possible for the test error to be lower than the training error?

Types of CV: Leave-one-out Cross-Validation

Assume we have *n* training examples.

A special case of k-fold CV is when k=n. This is called *leave-one-out cross-validation*.

Useful (only) if you have a tiny dataset where you can't afford a large validation set.

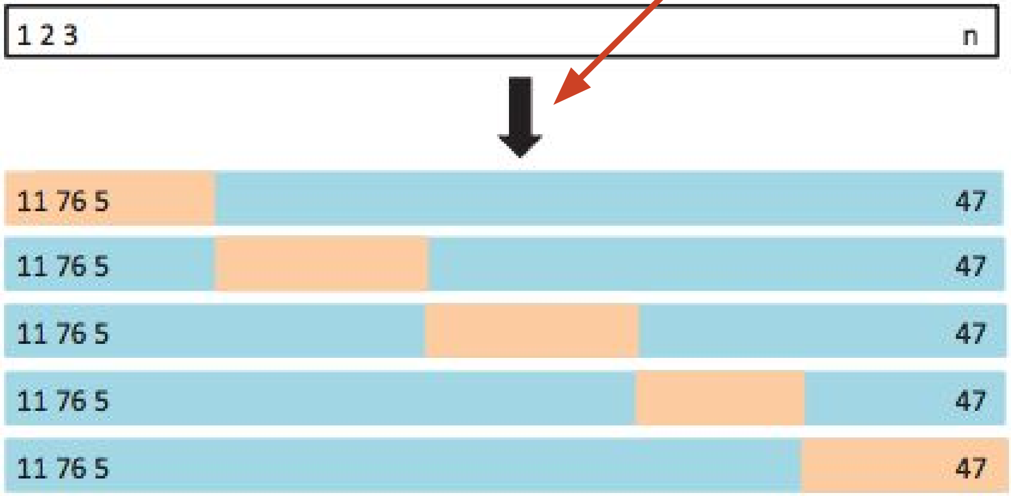


Types of CV: k-Fold Cross-Validation

Commonly, k=5 or k=10

randomize!

1. Split the dataset into k "folds".
2. Train using (k-1) folds. Validate using the one "leave out" fold. Record a validation metric such as RSS or accuracy.
3. Train *k* models, leaving out a different fold for each one.
4. Average the validation results.

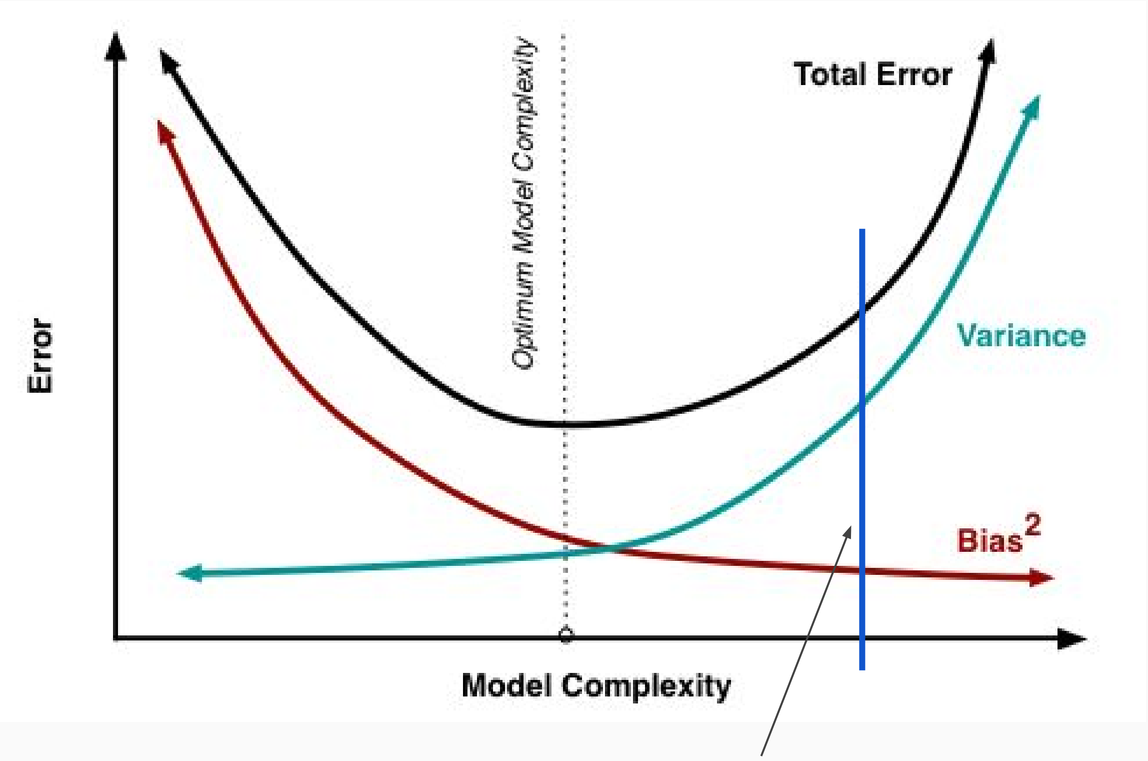


Overfitting in high dimensions is easy, even with simple models.

If our data has high dimensionality (many many predictors), then it becomes easy to overfit the data.

This is one result of the so-called **Curse of Dimensionality** (this afternoon).

Even linear regression might be too complex of a model for high dimensional data (and the smaller the dataset, the worse this problem is).



Linear regression in high dimensions

“HELP, my model is overfitting!”

You have a few options.

1. Get more data… (not usually possible/practical)
2. **Subset Selection:** keep only a subset of your predictors (i.e, dimensions)
3. **Regularization:** restrict your model’s parameter space (this afternoon)
4. **Dimensionality Reduction:** project the data into a lower dimensional space (later in course)

Subset Selection

Best subset: Try every model. Every possible combination of *p* predictors

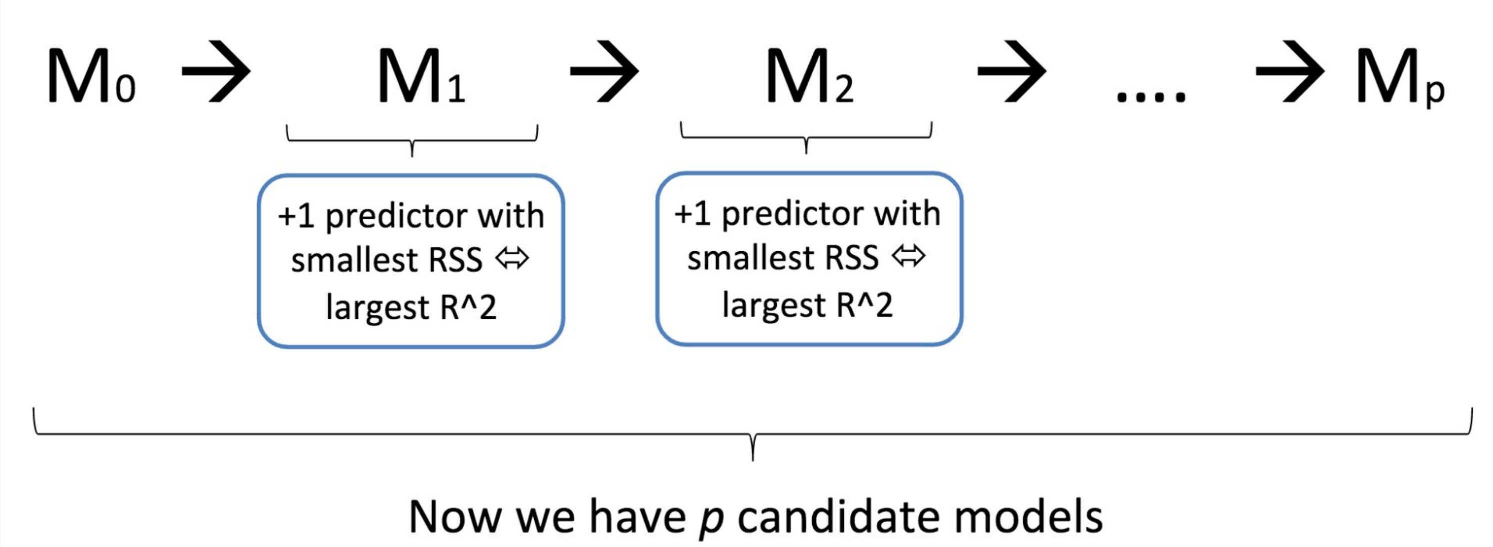
- Computationally intensive. 2^p possible subsets of *p* predictors
- High chance of finding a “good” model by random chance.

  ... A sort-of monkeys-Shakespeare situation ...

Stepwise: Iteratively pick predictors to be in/out of the final model.

- Forward, backward, forward-backward strategies

Forward Stepwise Selection



$M_0 \rightarrow M_1 \rightarrow M_2 \rightarrow \ldots. \rightarrow M_p$

+1 predictor with smallest RSS ⇔ largest R^2

+1 predictor with smallest RSS ⇔ largest R^2

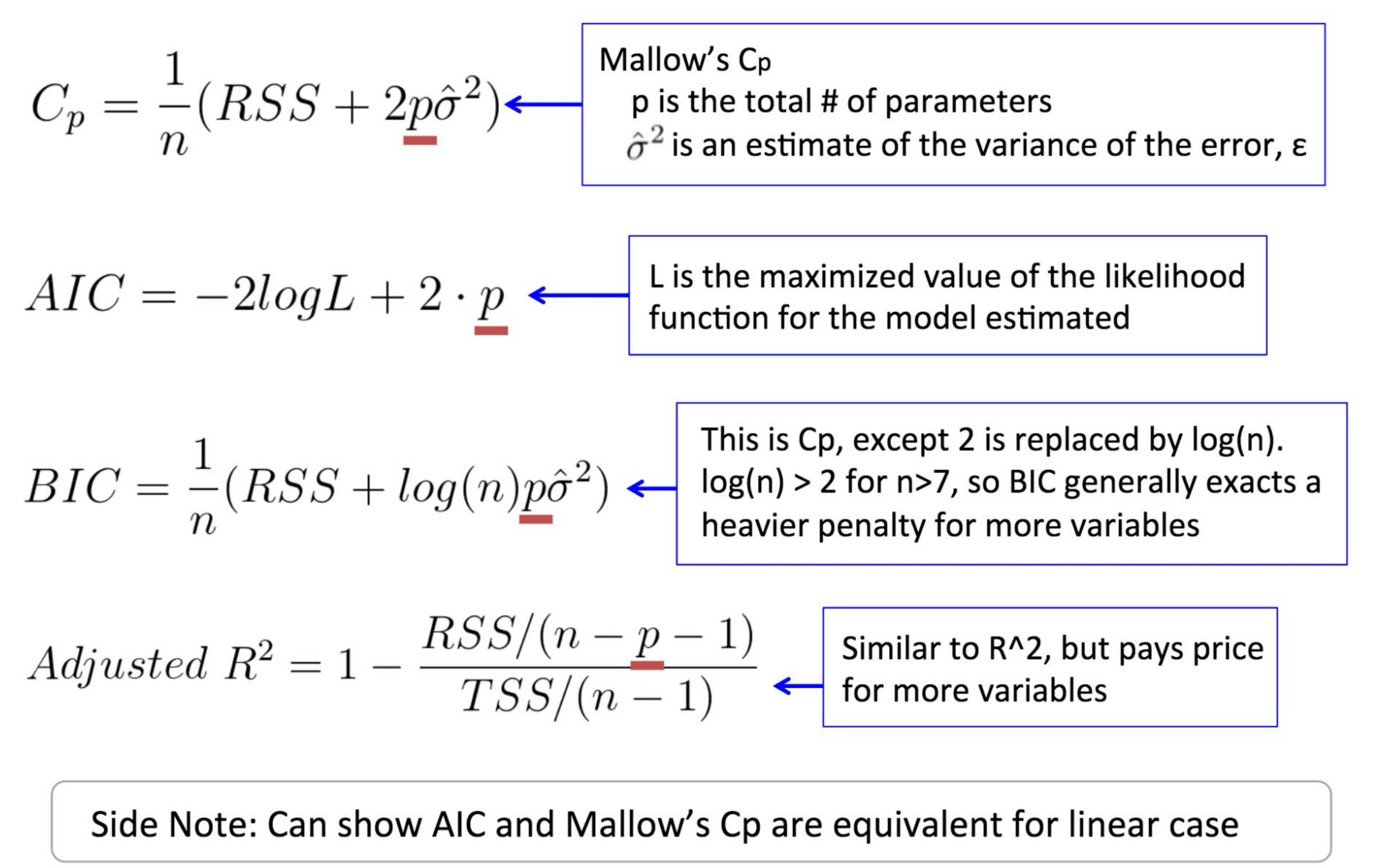
Now we have *p* candidate models

Are RSS and R^2 good ways to decide amongst the resulting *(p+1)* candidates?

Answer: Don't use RSS or R^2 for this part.
Use Mallow's C_p, or AIC, or BIC, or Adjusted R^2.

… or just use cross-validation with any error measurement.

Subset Selection: Comparing models of varying number of predictors...



$$C_p = \frac{1}{n}(RSS + 2\underline{p}\hat{\sigma}^2)$$

Mallow’s Cp
p is the total # of parameters
$\hat{\sigma}^2$ is an estimate of the variance of the error, ε

$$AIC = -2logL + 2 \cdot \underline{p}$$

L is the maximized value of the likelihood function for the model estimated

$$BIC = \frac{1}{n}(RSS + log(n)\underline{p}\hat{\sigma}^2)$$

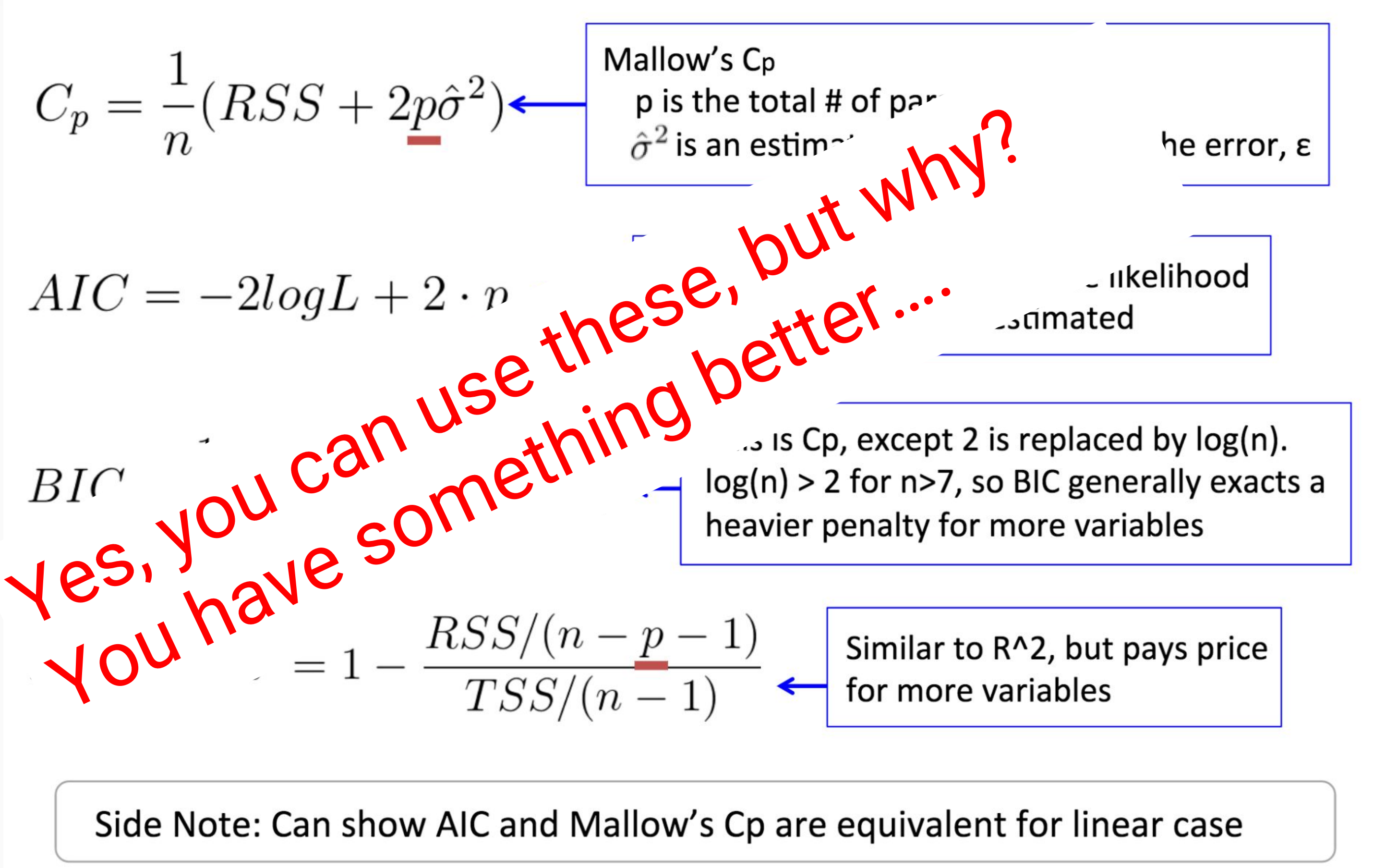
This is Cp, except 2 is replaced by log(n). log(n) > 2 for n>7, so BIC generally exacts a heavier penalty for more variables

$$Adjusted\ R^2 = 1 - \frac{RSS/(n-\underline{p}-1)}{TSS/(n-1)}$$

Similar to R^2, but pays price for more variables

Side Note: Can show AIC and Mallow’s Cp are equivalent for linear case

Subset Selection: Comparing models of varying number of predictors...



$$C_p = \frac{1}{n}(RSS + 2\underline{p}\hat{\sigma}^2)$$

Mallow's Cp

p is the total # of par

$\hat{\sigma}^2$ is an estim he error, ε

$$AIC = -2logL + 2 \cdot p$$

likelihood

stimated

BIC

is Cp, except 2 is replaced by log(n). log(n) > 2 for n>7, so BIC generally exacts a heavier penalty for more variables

$$= 1 - \frac{RSS/(n-\underline{p}-1)}{TSS/(n-1)}$$

Similar to R^2, but pays price for more variables

Yes, you can use these, but why?

You have something better….

Side Note: Can show AIC and Mallow's Cp are equivalent for linear case

Use Cross-Validation