

# Validation Techniques and Learning Curves

# The Story So Far

We've seen several learning models now (KNN, Regression, Logistic Regression, Neural Nets, etc)

You've done your own experiments where you've selected hyperparameters:

- K in K-Nearest Neighbors
- Regularization strength in Regression and Logistic Regression

We've talked about the importance of evaluating a learning model on **unseen validation data**

You've been introduced to the Confusion Matrix and what it can tell you about how your learning algorithm makes mistakes

# The Story So Far

Today we'll look more closely at how we validate learning models

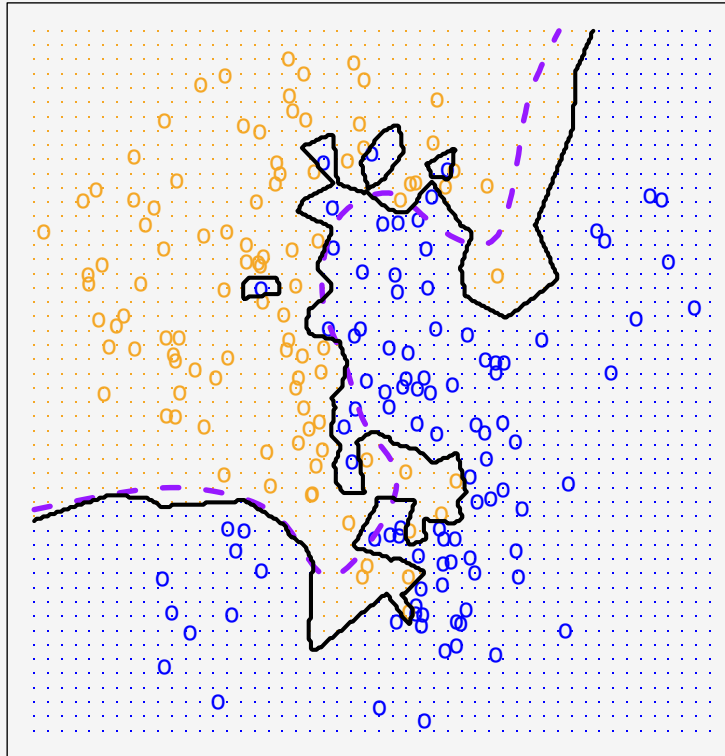
We'll ask questions about whether or not we can believe our validation results

Next time we'll ask questions about whether simple accuracy measures are the best thing to do

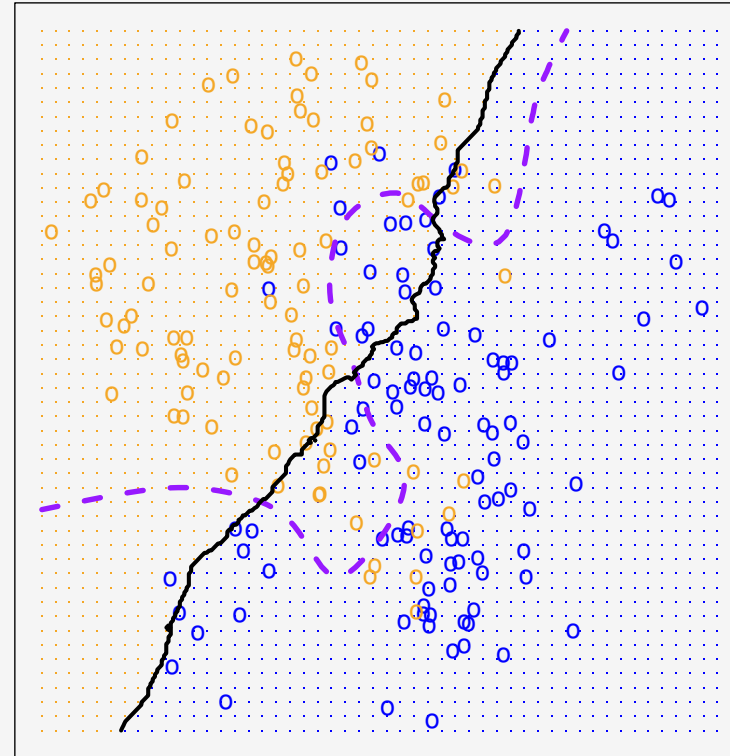
# The Real ML Pipeline (For Reals This Time)

- Overfitting occurs when our model works too hard to fit the training data
- We really care about how the model performs on unseen test data

KNN:  $K=1$

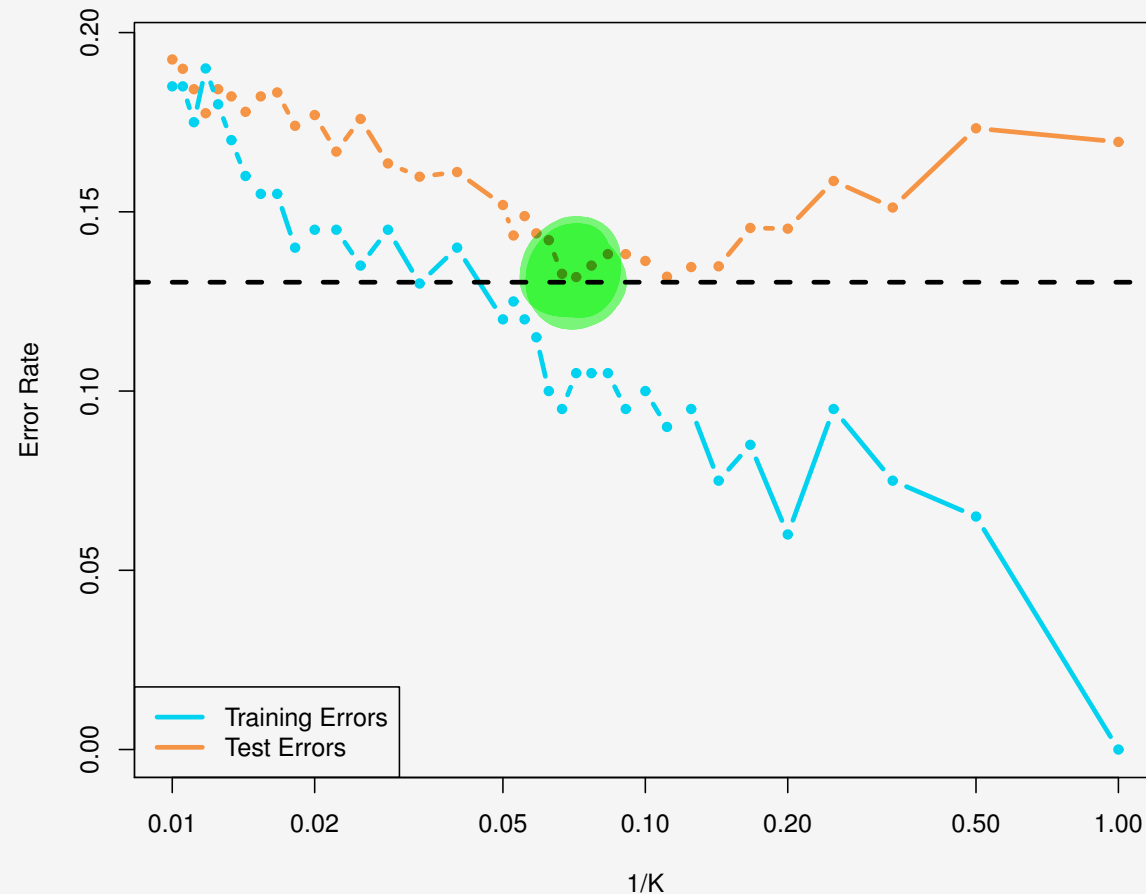


KNN:  $K=100$



# The Real ML Pipeline (For Reals This Time)

- Overfitting occurs when our model works too hard to fit the training data
- We really care about how the model performs on unseen test data



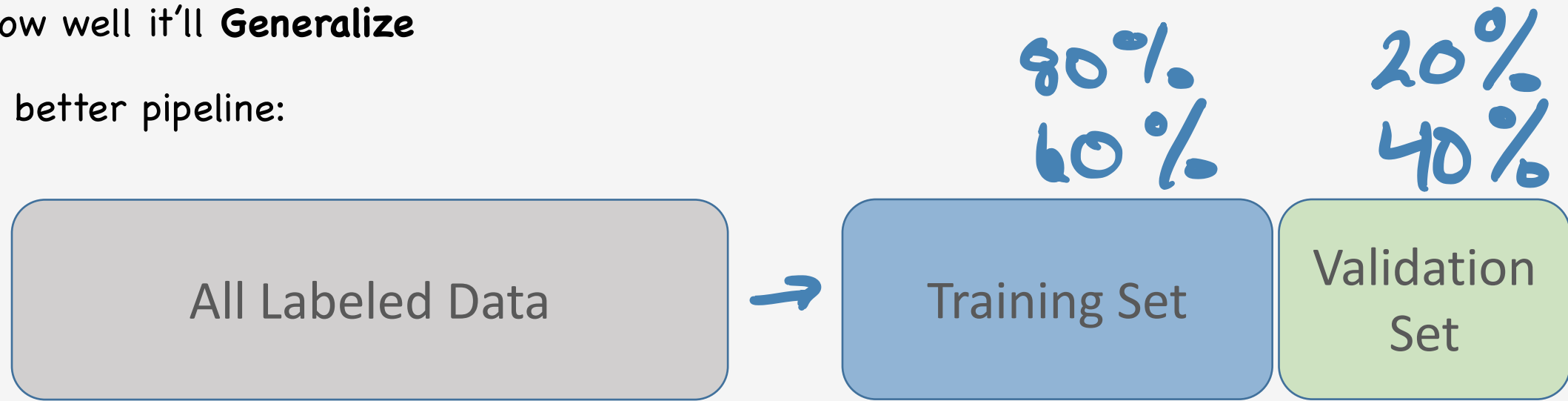
# The Real ML Pipeline (For Reals This Time)

The more flexible (powerful) your model gets, the better it'll do on training set

But that's not useful. We want to know how model will do on new data

How well it'll **Generalize**

A better pipeline:

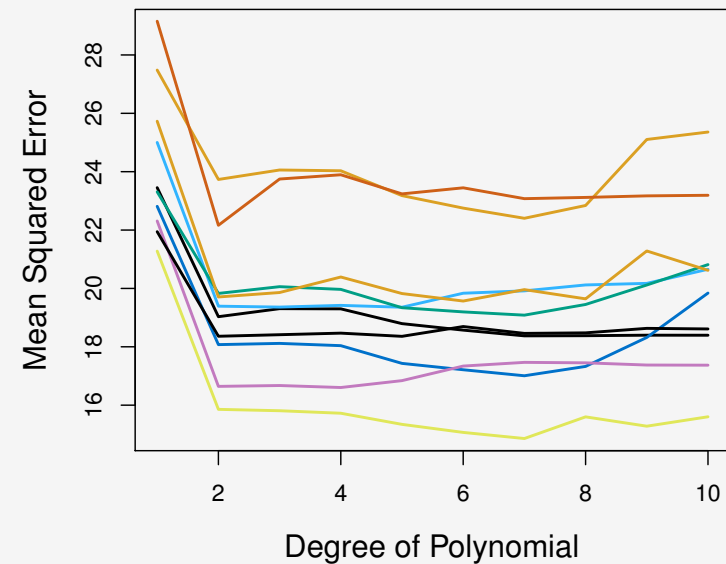
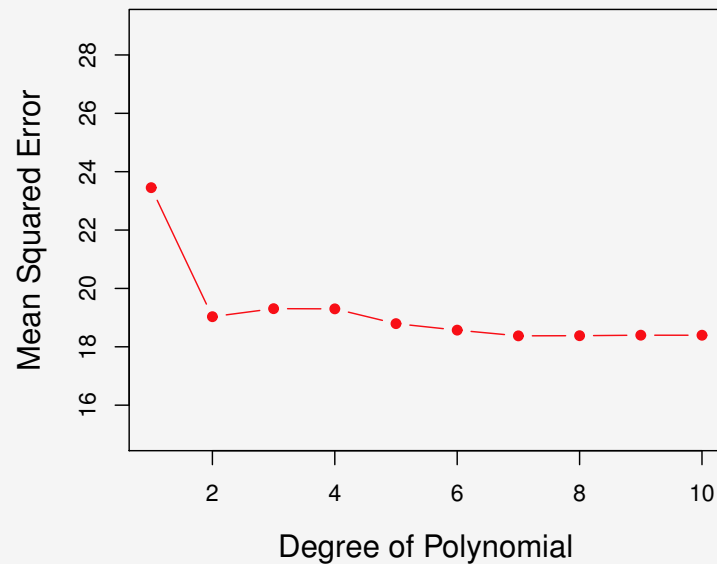


**Train** on training set. **Validate** (or evaluate) on validation set.

# Difficulties of Validation Set Approach

**Example:** Polynomial Regression on automobile features (engine size, etc) to predict gas mileage

Split into training and validation set and evaluate model for different polynomial degrees

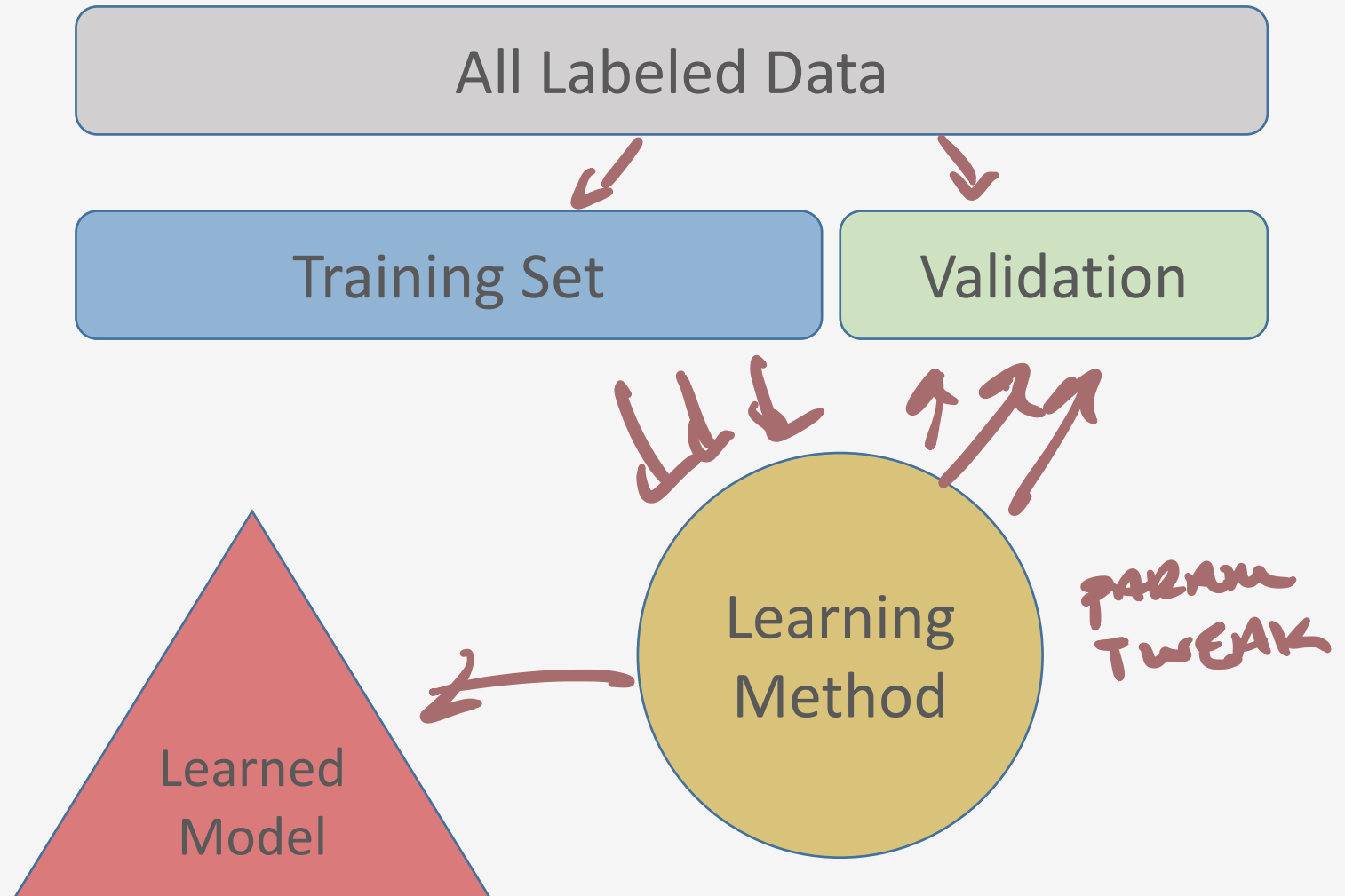


Actual MSE depends quite a bit on particular train-validation split

The fact that we're limiting training set size means we tend to overestimate generalization error

# Learning Algorithm Life-Cycle

- Train on training data
- Test on validation data
- Tweak tuning parameters
- Repeat





# Learning Algorithm Life-Cycle

- Train on training data
- Test on validation data
- Tweak tuning parameters
- Repeat

All Labeled Data

Training Set

Validation

Seems reasonable.

Probably can get accurate models

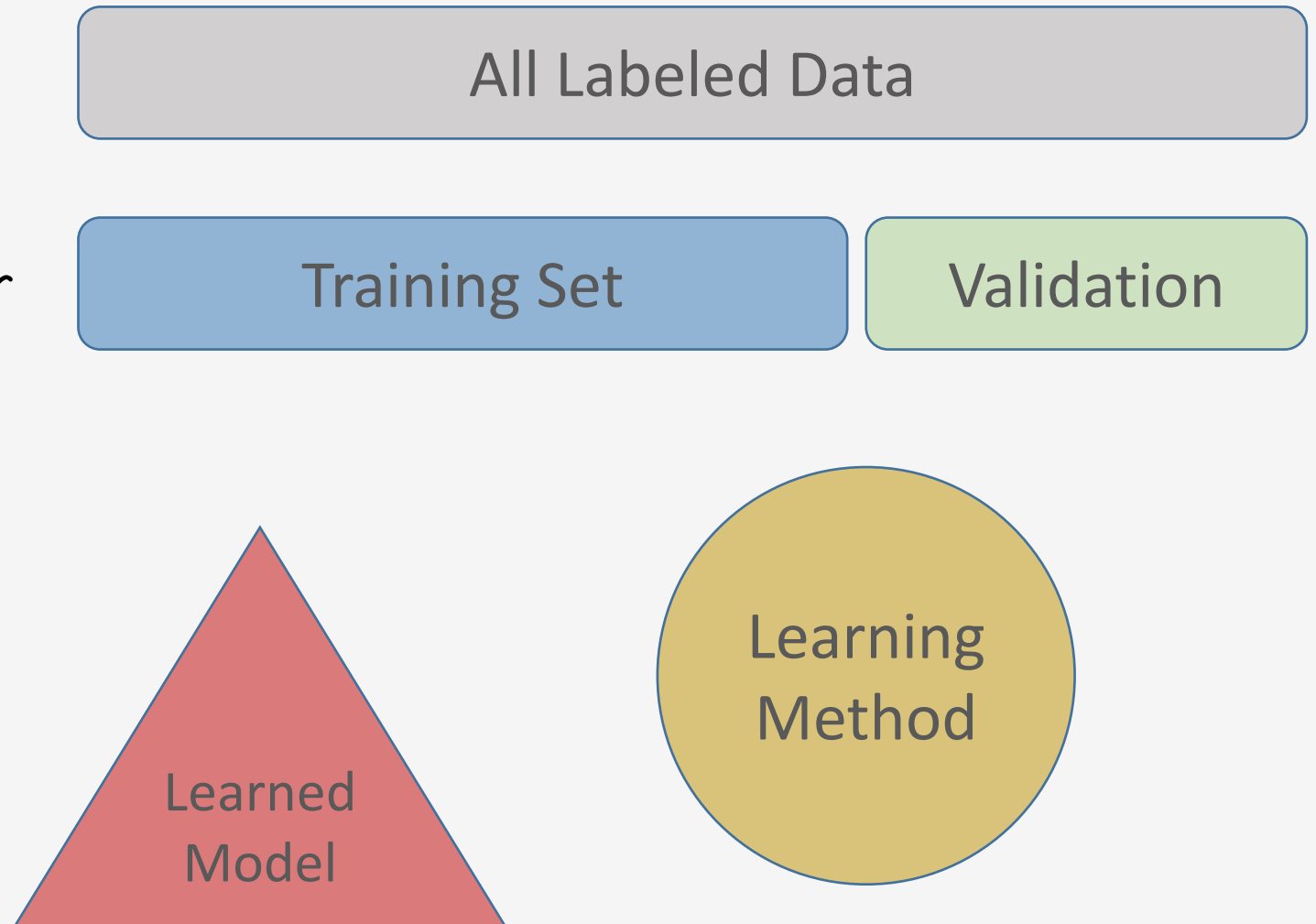
What are **potential pitfalls**?

Learned  
Model

Learning  
Method

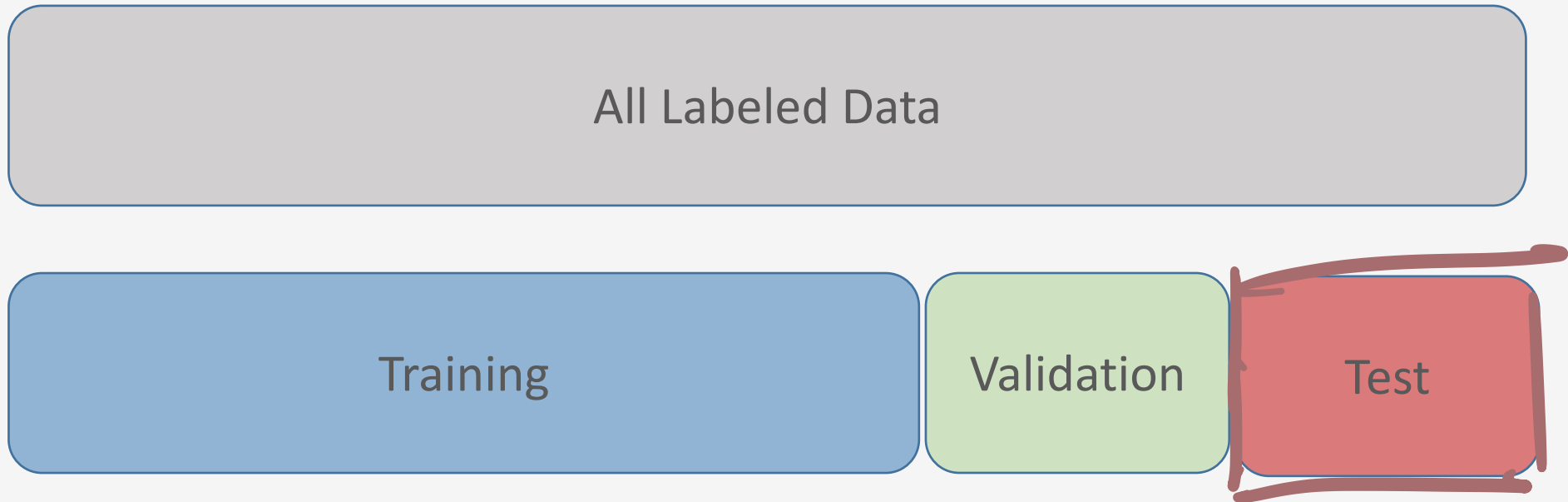
# Learning Algorithm Life-Cycle

- Can actually do **human overfitting** of validation data!
- Validation error no longer accurate indicator of true error
- Need another data partition ...



# The Real ML Pipeline (For Reals This Time)

- Split all labeled data into Training Set / Validation Set / Test Set



- Do feature engineering and parameter tuning while iterating on Validation set
- Do final evaluation of model on Test set **Exactly One Time**
- 60% / 20% / 20% – split is popular, but not a firm rule

# The Real ML Pipeline (For Reals This Time)

OK, but that kinda sucks. We just lost 20% of our training data

What if you just can't spare the training data?

- **Option 1:** Try to get more data
- **Option 2:** Do **Cross-Validation**

The general idea behind **Cross-Validation**:

- Split training data into many partitions
- Repeatedly train on some partitions and test on others
- Aggregate accuracy / error results

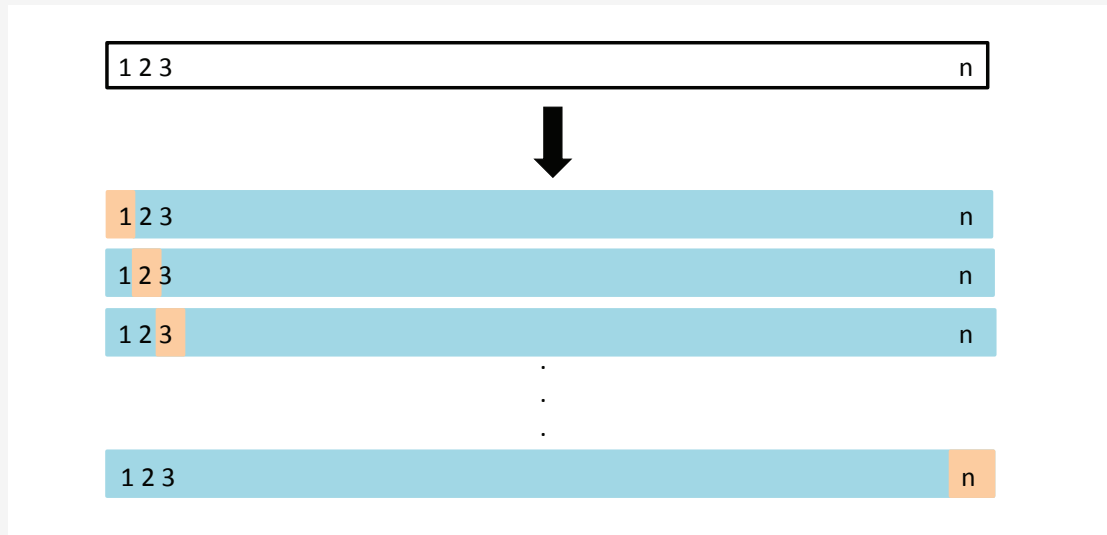
# Leave-One-Out Cross-Validation

- LOOCV is closely related to the validation set approach, but addresses some drawbacks
- Instead of creating two partitions of similar size, use a single observation as validation set
- Train model on training set, evaluate on single-point validation set
- Do this  $n$  times, with each point having a turn as the validation set, then average



# Leave-One-Out Cross-Validation

- LOOCV is closely related to the validation set approach, but addresses some drawbacks
- Instead of creating two partitions of similar size, use a single observation as validation set
- Train model on training set, evaluate on single-point validation set
- Do this  $n$  times, with each point having a turn as the validation set, then average



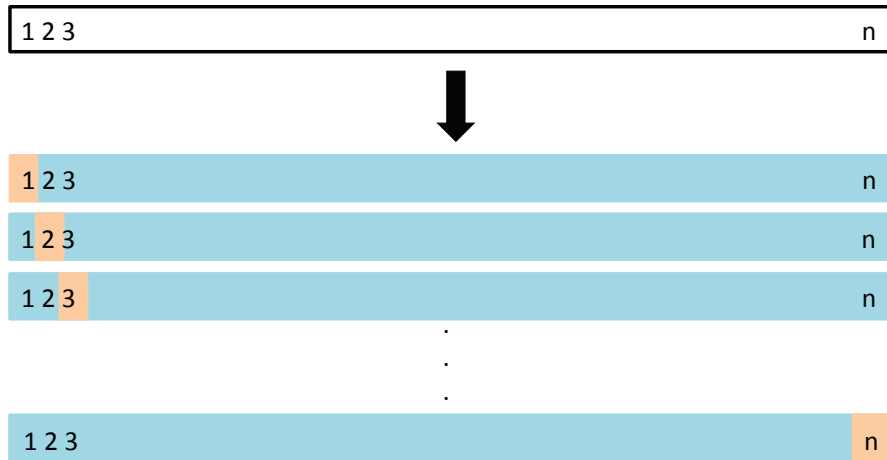
**def:** LOOCV MSE:

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

↑  
p+1 left out

# Leave-One-Out Cross-Validation

- LOOCV is closely related to the validation set approach, but addresses some drawbacks
- Instead of creating two partitions of similar size, use a single observation as validation set
- Train model on training set, evaluate on single-point validation set
- Do this  $n$  times, with each point having a turn as the validation set, then average



**def:** LOOCV Error Rate:

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

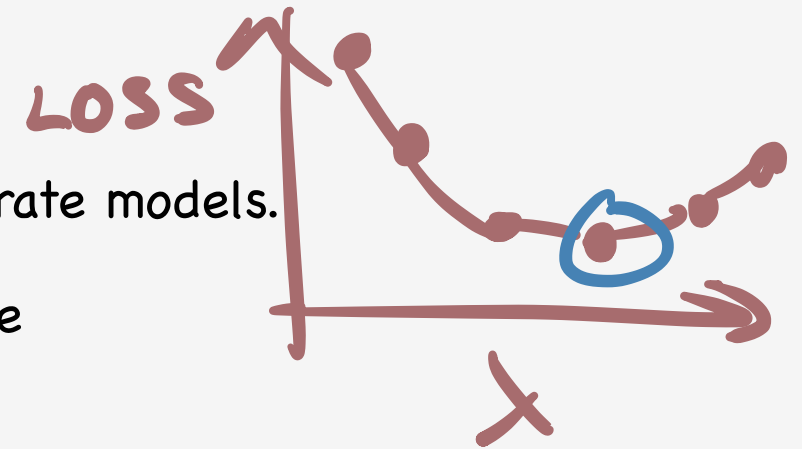
# Leave-One-Out Cross-Validation

## Advantages Over Validation-Set Approach:

- Less bias due to larger training-set size. Doesn't overestimate generalization error as much.
- Unlike validation-set approach, will always yield same result because there's no randomness

## Disadvantages:

- In general, very expensive to compute. Have to train  $n$  separate models.
- Caveat: Some tricks for regression that mitigate this expense



FOR  $i$  IN  $1:n$   
LEAVE OUT  $(x_i, y_i)$

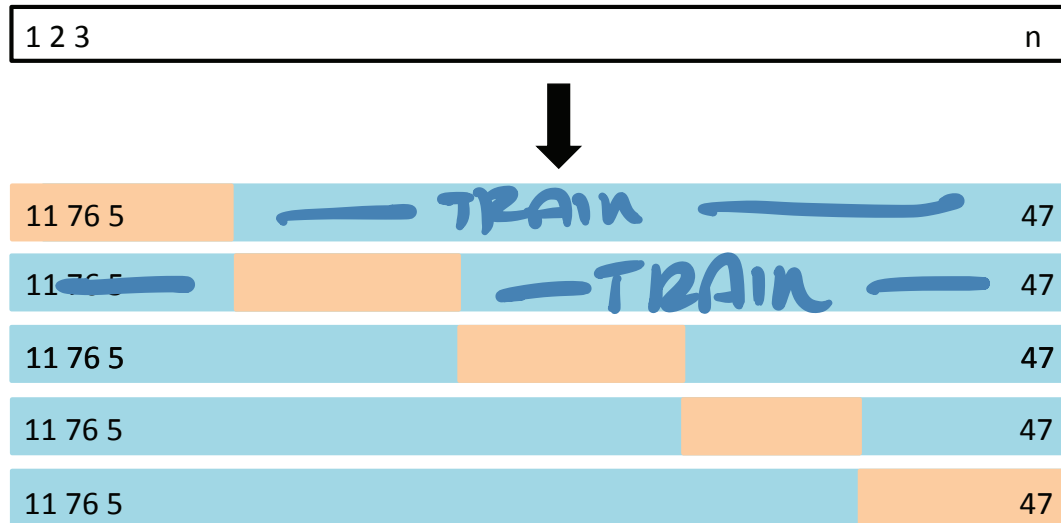
SOLVE (SGD, PLA, KNN)  
EVALUATE  $\rightarrow$  ERR



# k-Fold Cross-Validation

A less-computationally intensive approach is k-Fold Cross-Validation

- Randomly partition labeled data set into k folds (partitions)
- Iteratively use one fold as validation set and train on other k-1 folds. Aggregate results.

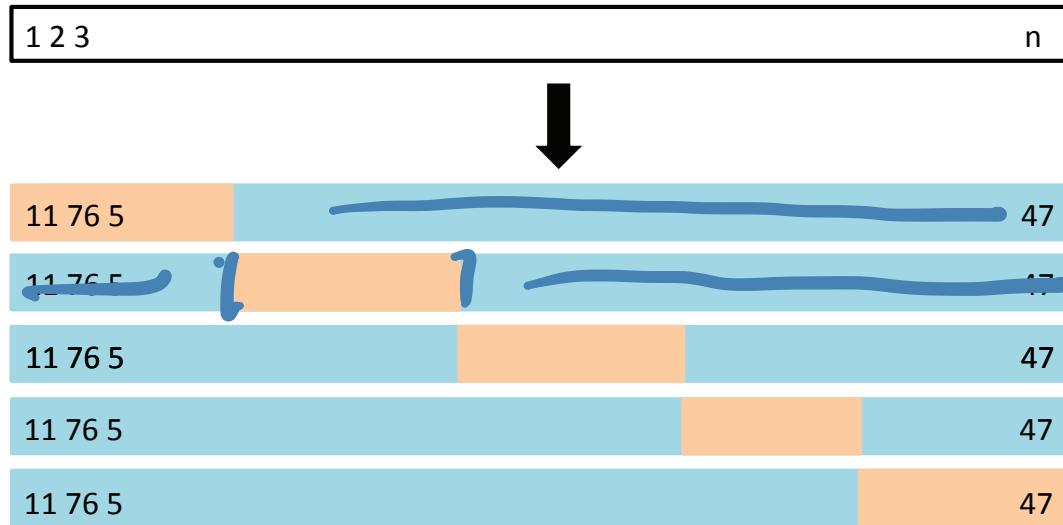


$k=5$   
← LABELLED DATA

# k-Fold Cross-Validation

A less-computationally intensive approach is k-Fold Cross-Validation

- Randomly partition labeled data set into k folds (partitions)
- Iteratively use one fold as validation set and train on other k-1 folds. Aggregate results.




$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k \text{MSE}_i$$

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k \text{Err}_i$$

# k-Fold Cross-Validation

Example: Suppose you perform 5-Fold CV on classification problem with  $n=25$  examples

iteration	train on	test on	error rate
1	$F_1, F_2, F_3, F_4$	$F_5$	9/20
2	$F_1, F_2, F_3, F_5$	$F_4$	3/20 
3	$F_1, F_2, F_4, F_5$	$F_3$	4/20
4	$F_1, F_3, F_4, F_5$	$F_2$	7/20
5	$F_2, F_3, F_4, F_5$	$F_1$	4/20

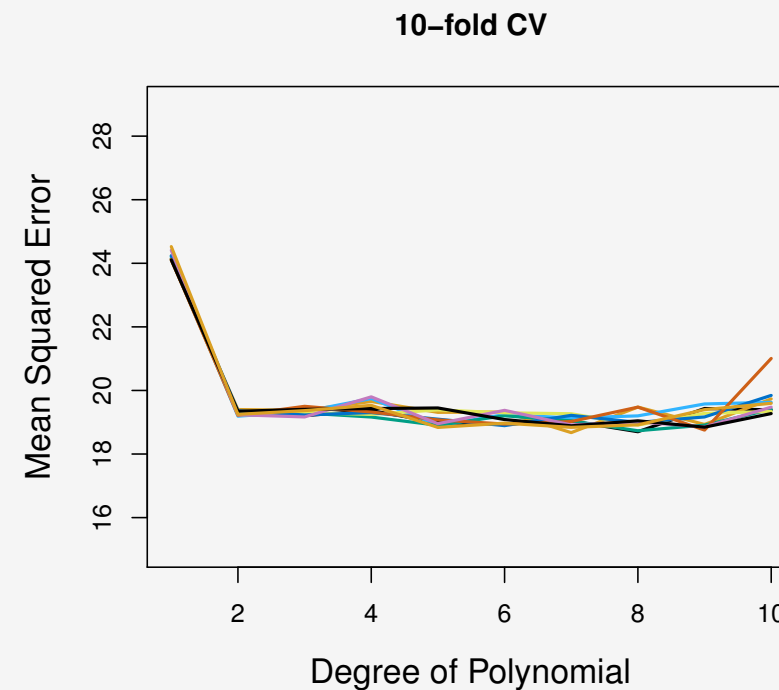
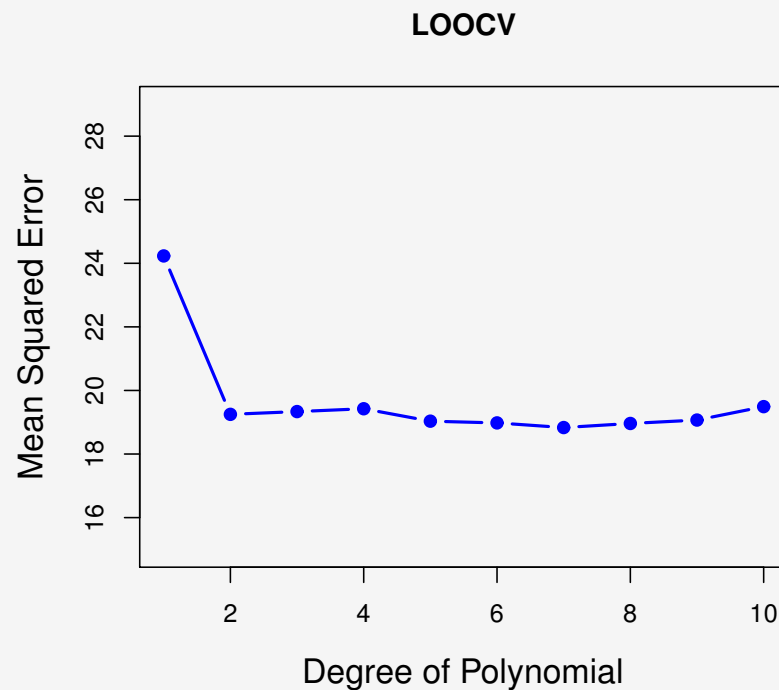
$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k \text{Err}_i$$

# k-Fold Cross-Validation

Note that LOOCV is a special case of k-Fold CV where  $k = n$

## Advantages:

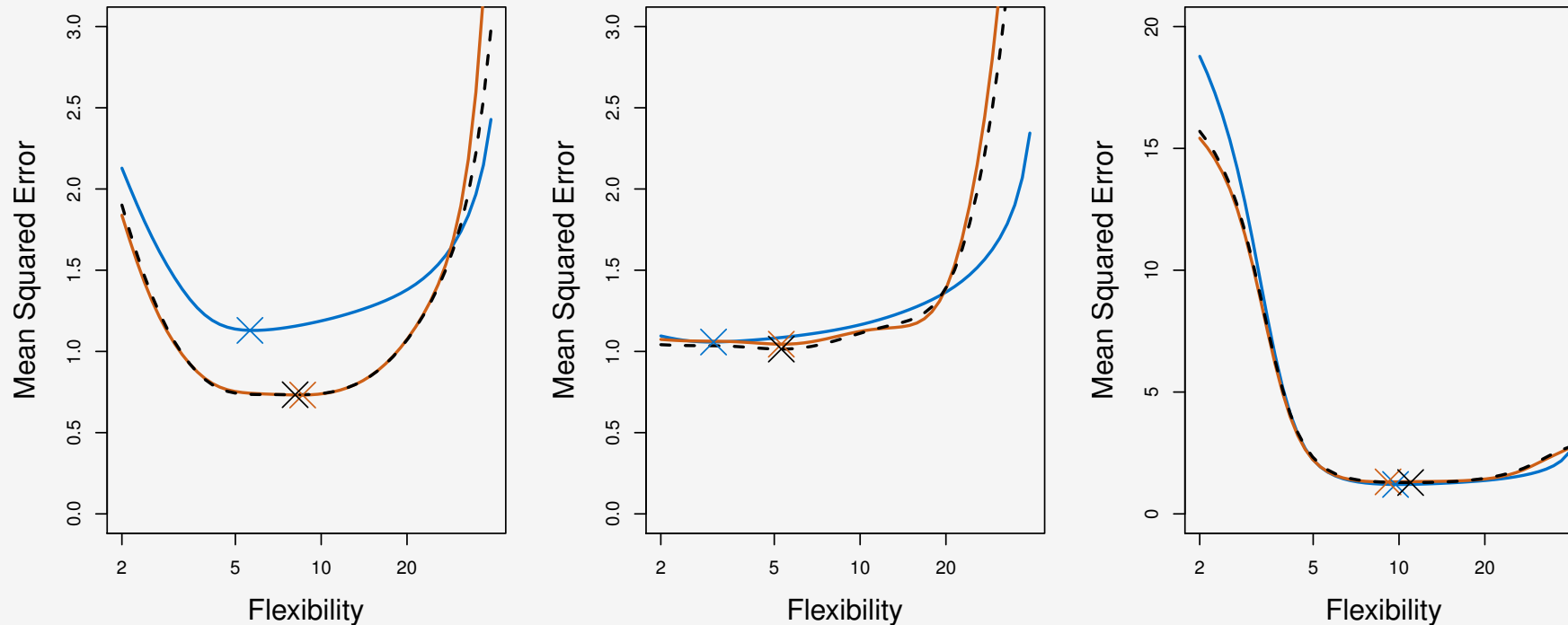
- In practice, people usually choose  $k=5$  or  $k=10$ , requiring much lower training cost



# k-Fold Cross-Validation

Cross-Validation can be a good indicator of generalization error

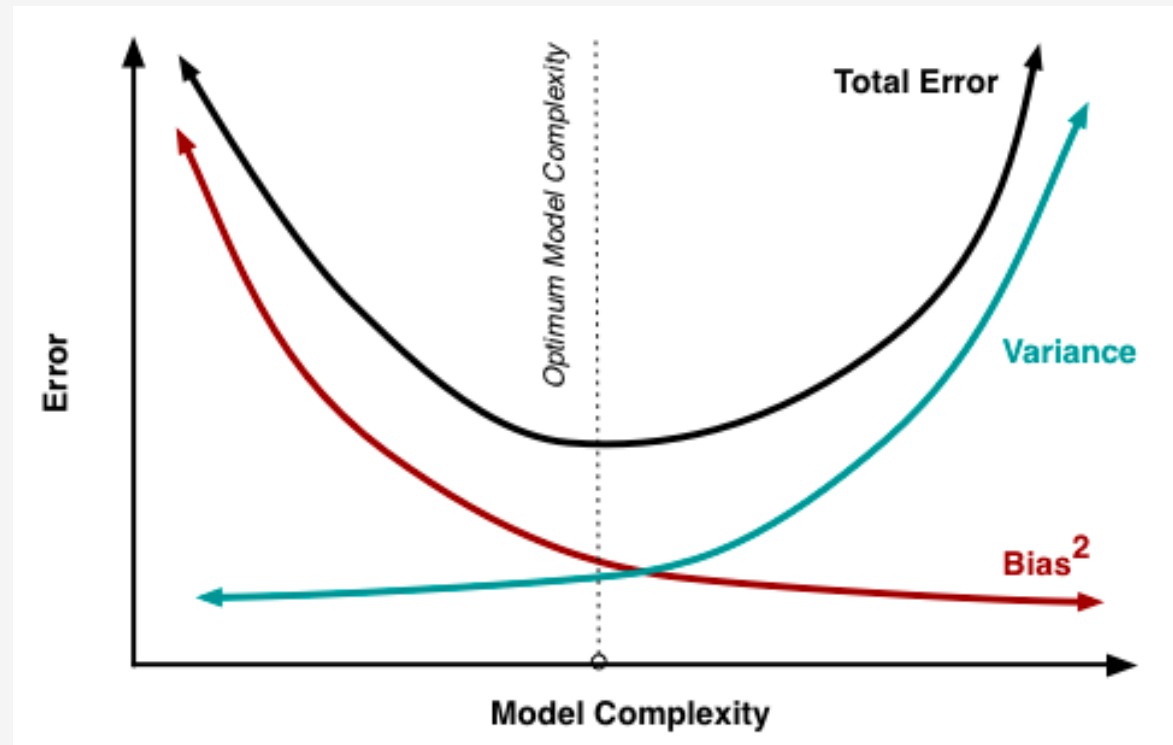
Difficult to verify on real data, but can see with simulated data



# The Bias-Variance Trade-Off

The generalization error is a combination of the bias and variance of a model

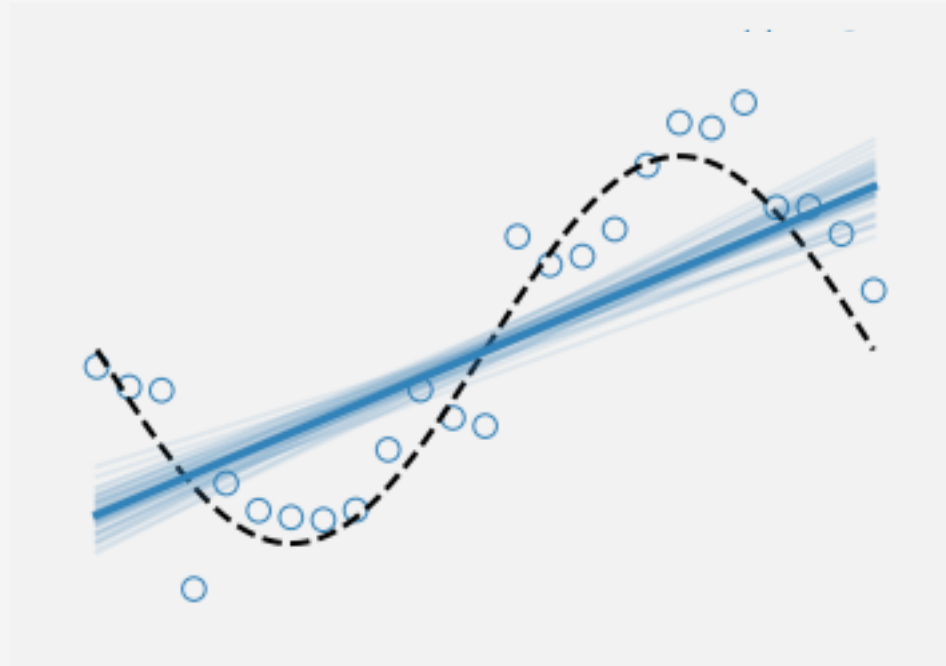
$$MSE = \left[ \text{Bias}(\hat{f}) \right]^2 + \text{Var}(\hat{f}) + \text{Var}(\epsilon)$$



# High Bias Intuition

The squared bias is  $\left[ \text{Bias}(\hat{f}) \right]^2 = \left( f - E[\hat{f}] \right)^2$

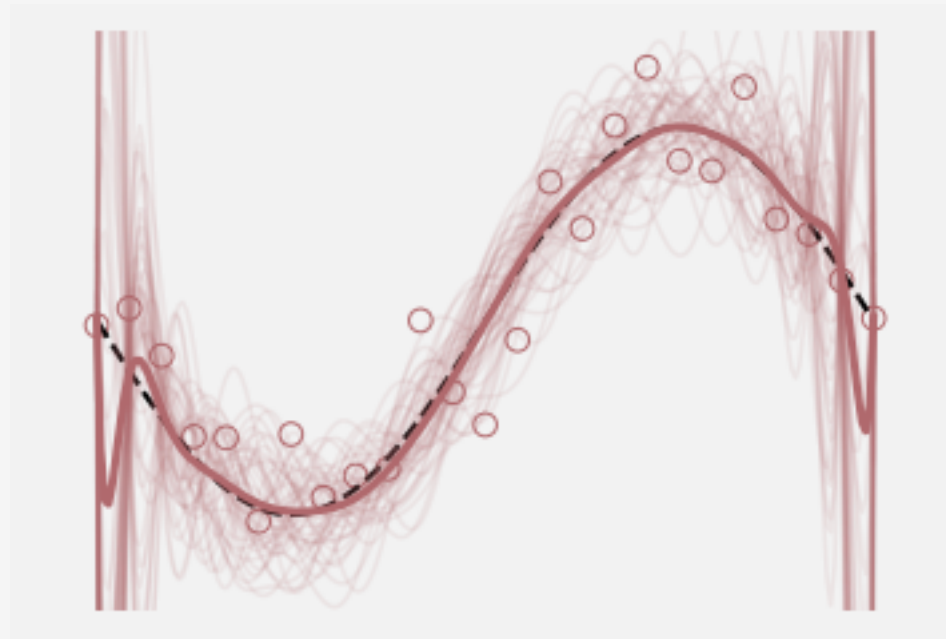
A method has high bias when, even with all the training data in the world, the error is still high. **Model is much less flexible than true function.**



# High Variance Intuition

The variance is  $\text{Var}(\hat{f}) = E[(\hat{f} - E[\hat{f}])^2]$

On average, over many training sets, our learned model is far from the model we could learn with infinite data. **Model is very sensitive to training data.**

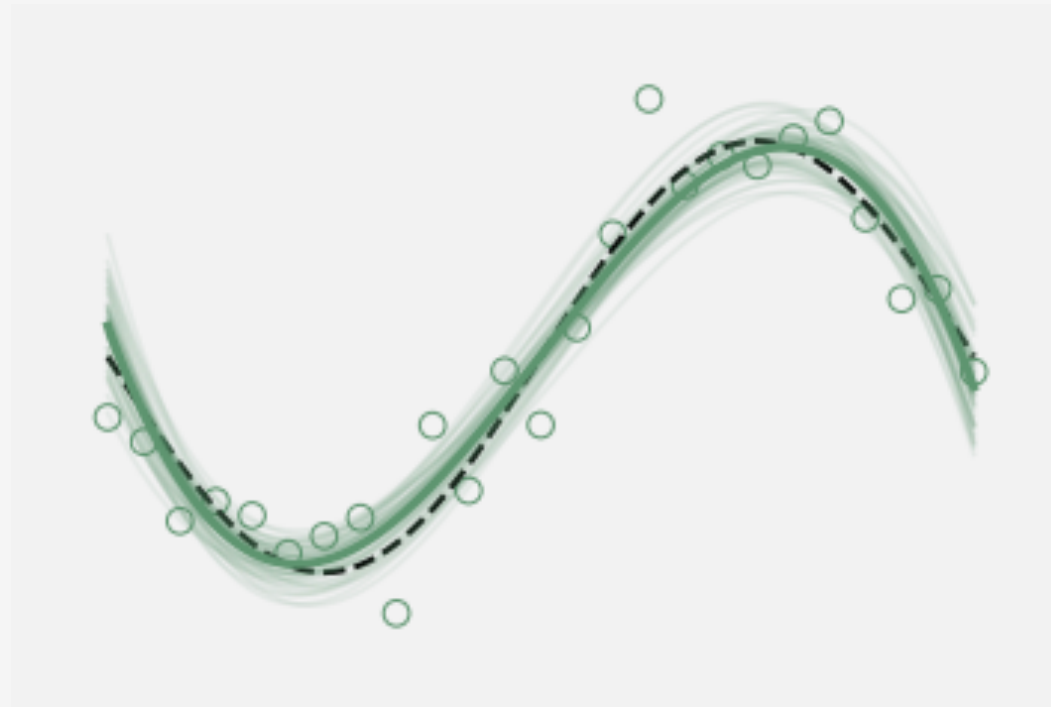




# The Bias-Variance Trade-Off

The generalization error is a combination of the bias and variance of a model

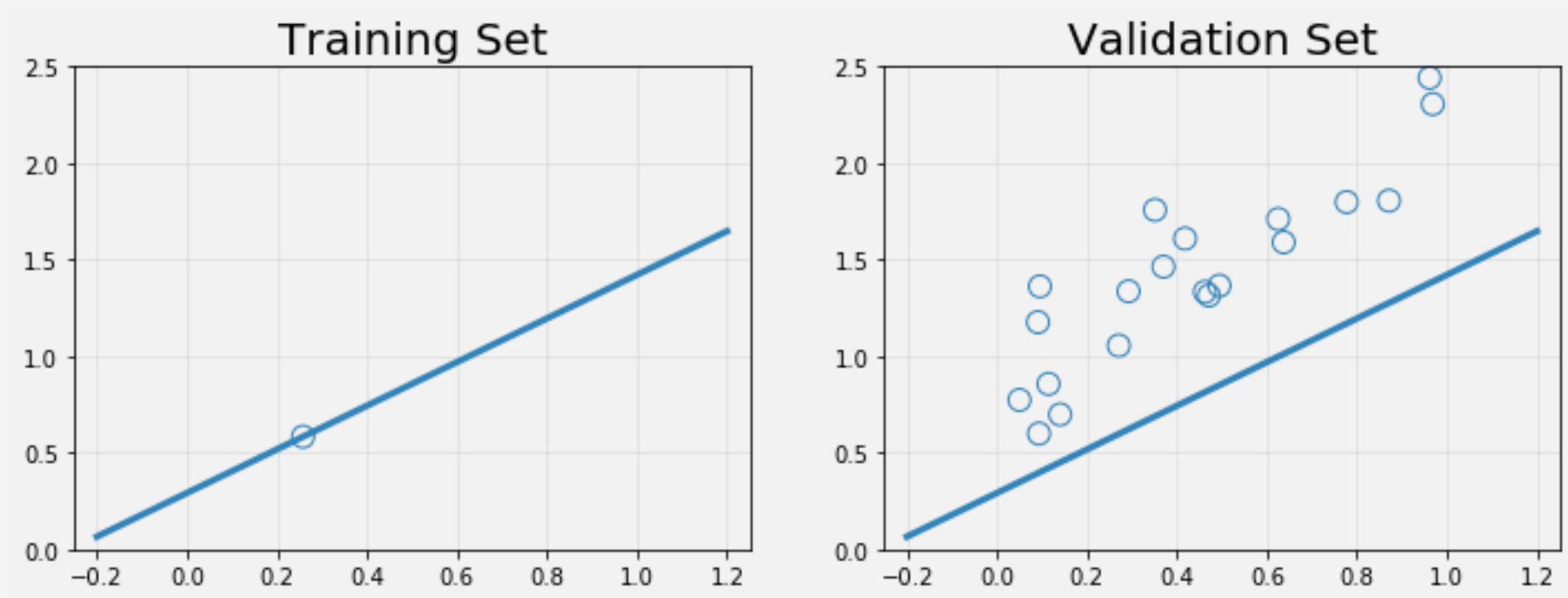
$$MSE = \left[ \text{Bias}(\hat{f}) \right]^2 + \text{Var}(\hat{f}) + \text{Var}(\epsilon)$$



# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

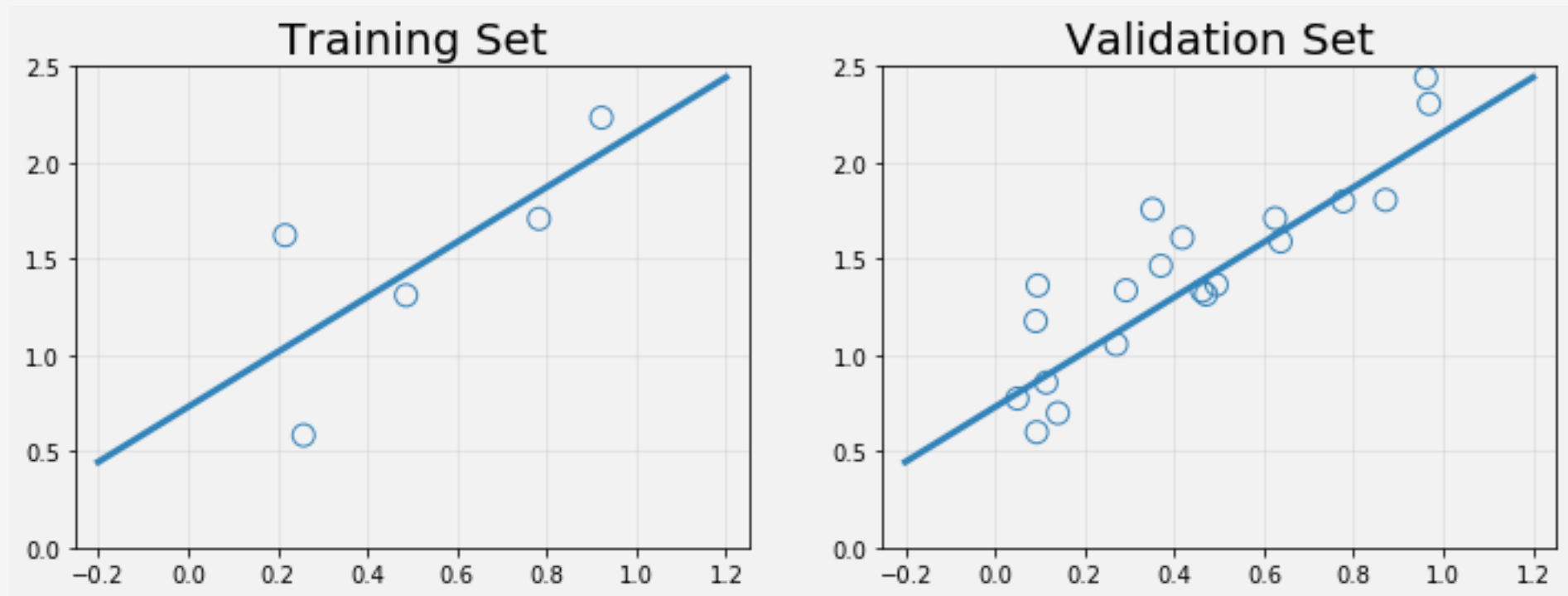
Evaluate your training and test error for increasing training set sizes



# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

Evaluate your training and test error for increasing training set sizes

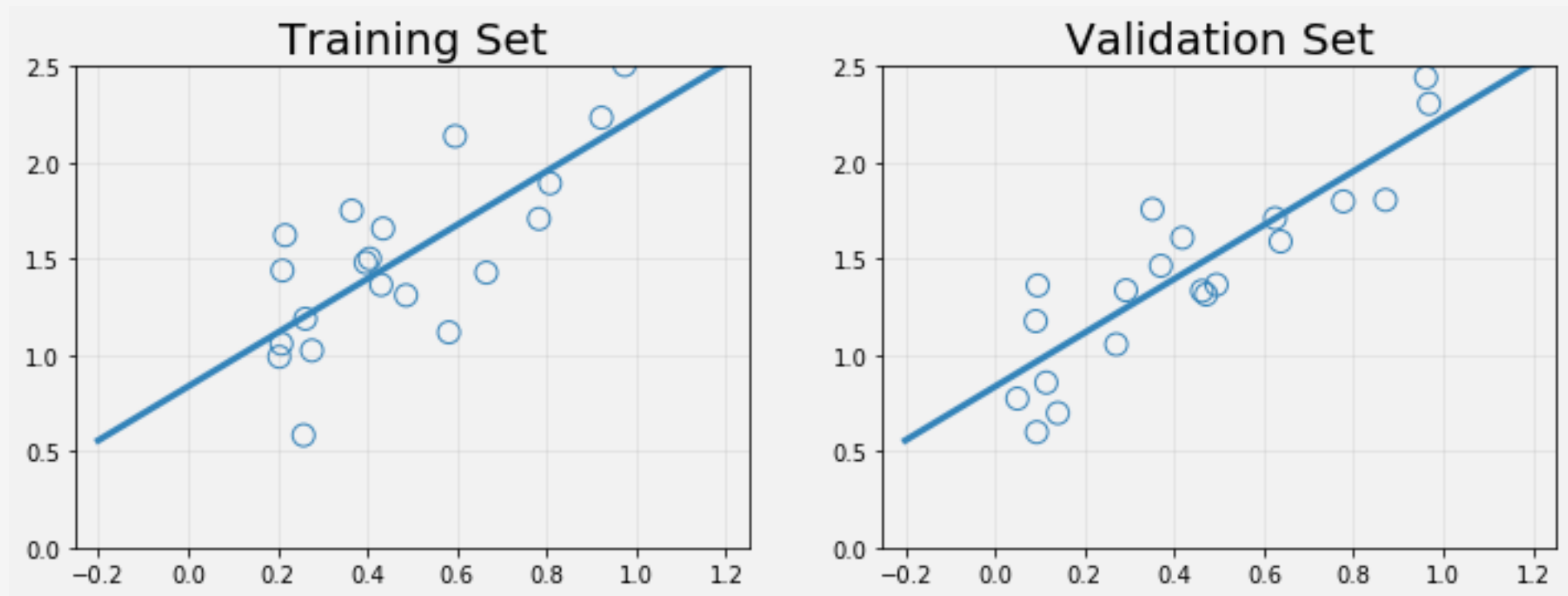


# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

Evaluate your training and test error for increasing training set sizes

*CROSS-VALIDATION*

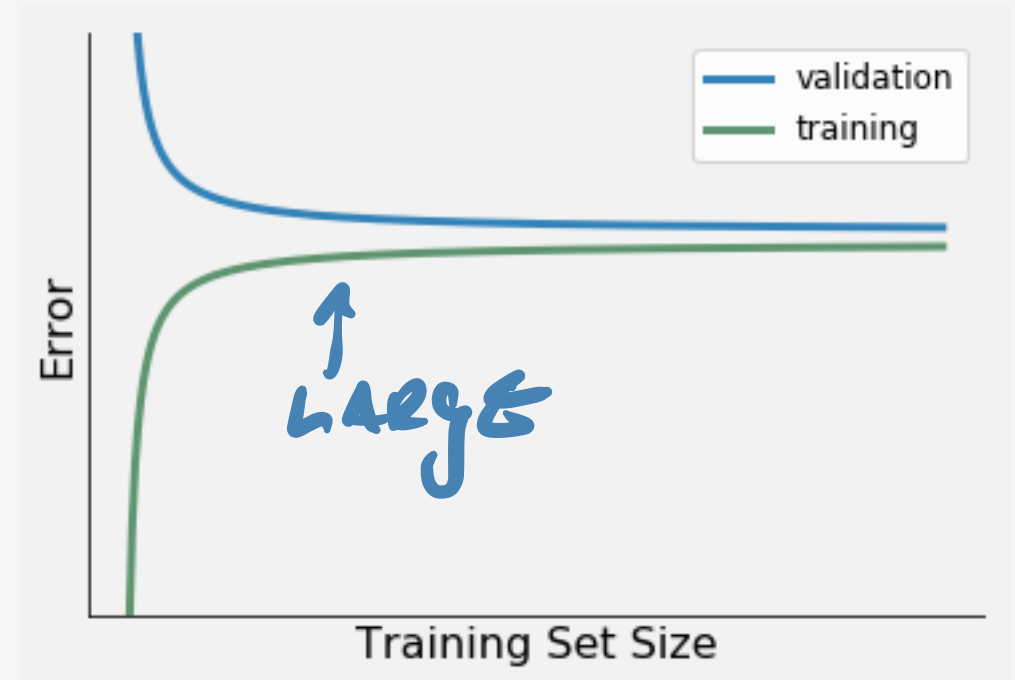


# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

## Low Variance / High Bias

- Large Training Error
- Small gap between train and validation
- Meeting between the two very fast

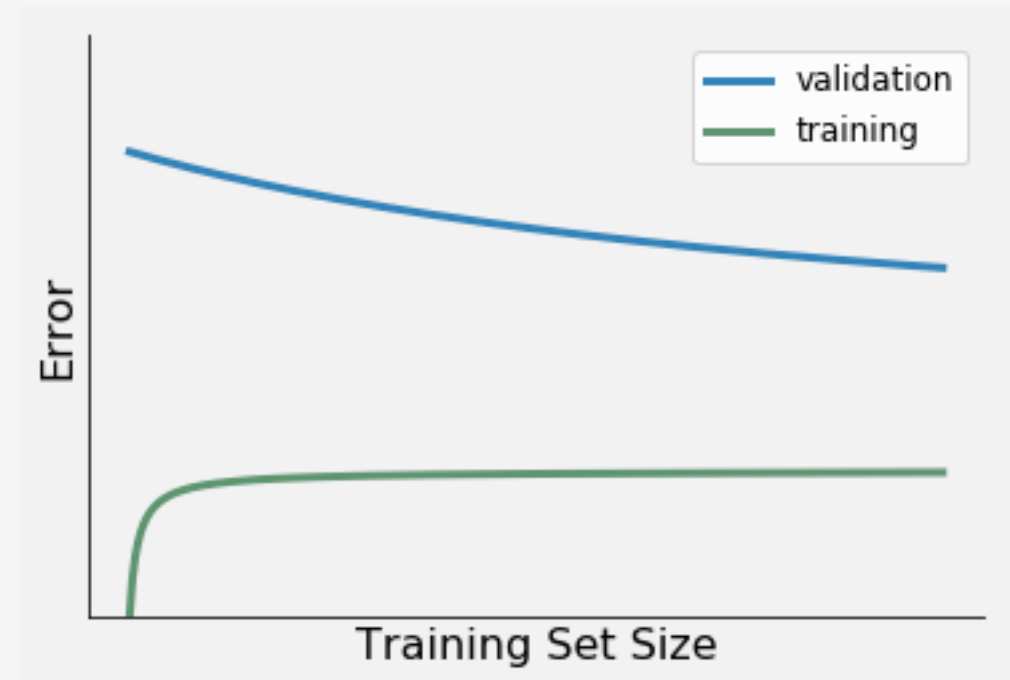


# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

High Variance / Low Bias

- Small Training Error
- Large gap between train and validation
- Downward trend in validation error tells us that we'll keep improving if we can get lots of data

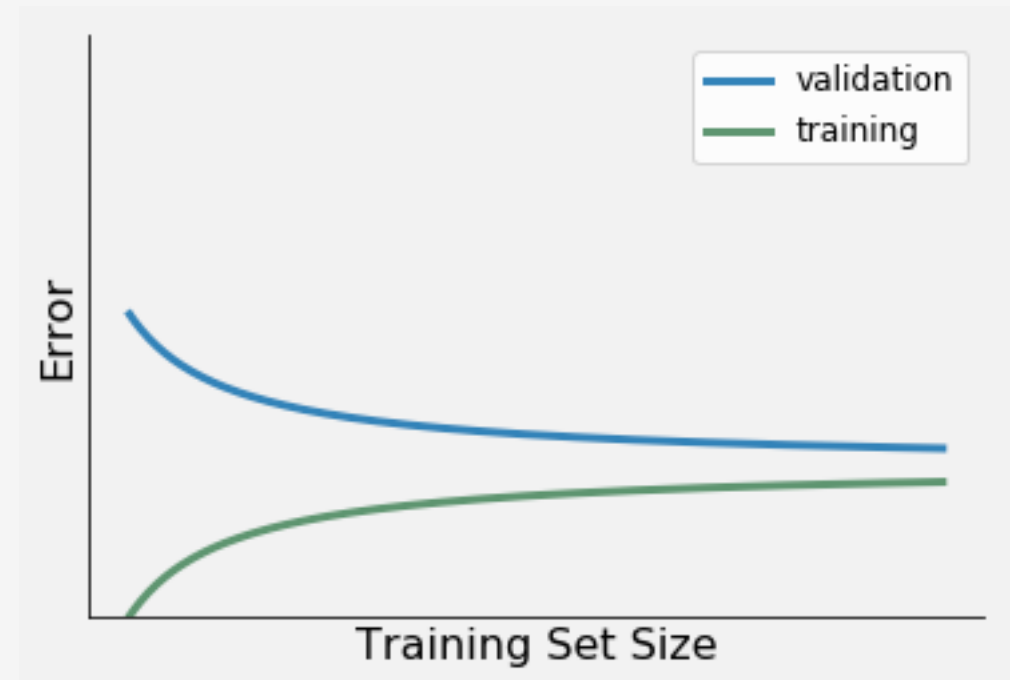


# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

Low Variance / Low Bias (Our Goal)

- Small Training Error
- Small gap between train and validation

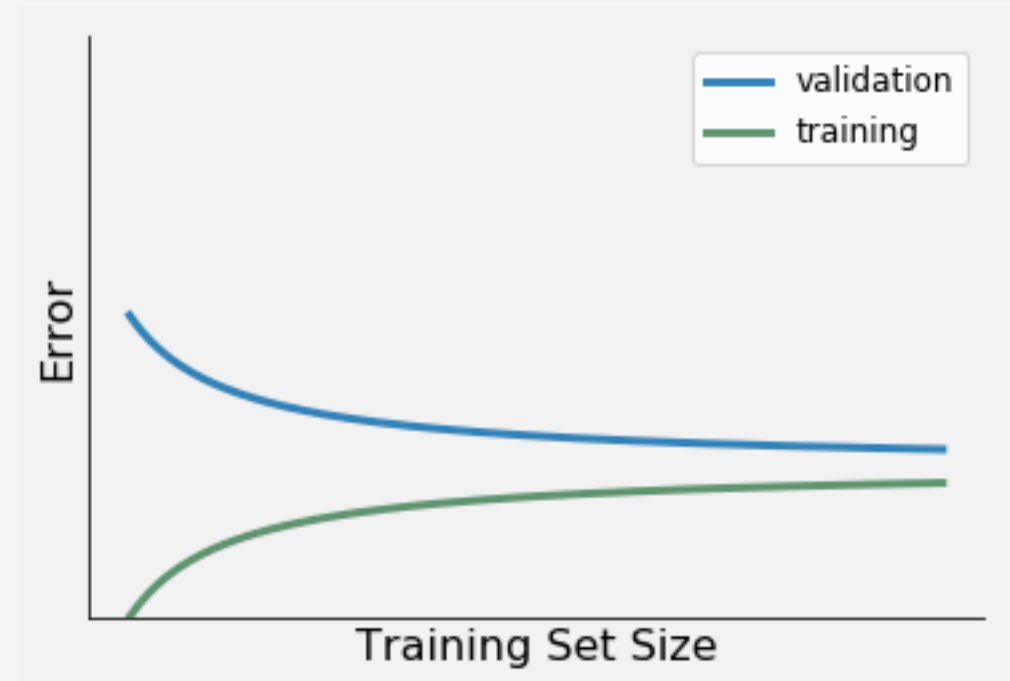


# Learning Curves and What They Tell Us

A learning curve is a great way to diagnose bias and variance in a model

## Learning Curve Summary:

- Gap tells you about variance
- Size of Training error tells you about bias
- Slope of validation error tells you if you should bother getting more data





# Acknowledgements

Many images in this lecture were taken from:

- **James, et al.** *Introduction to Statistical Learning with Applications in R*





