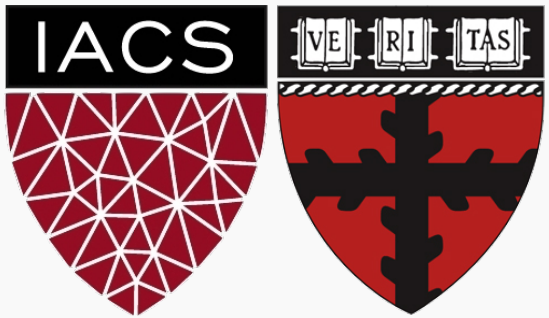
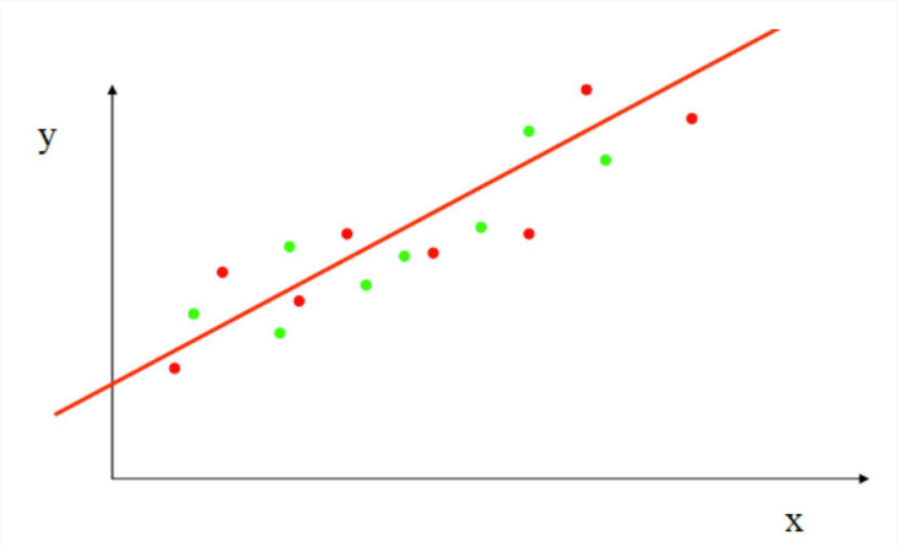
Generalization

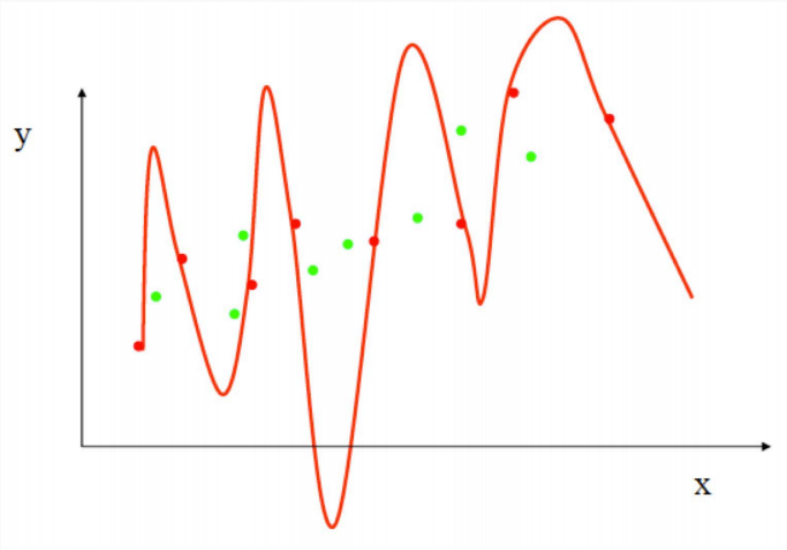
Kigali, Rwanda
June 2019



Evaluating Your 'Best Model'

In the following, we have a linear regression model and a polynomial regression model fitted on the same training data (in red). The green is a new data that we did not train our model on.



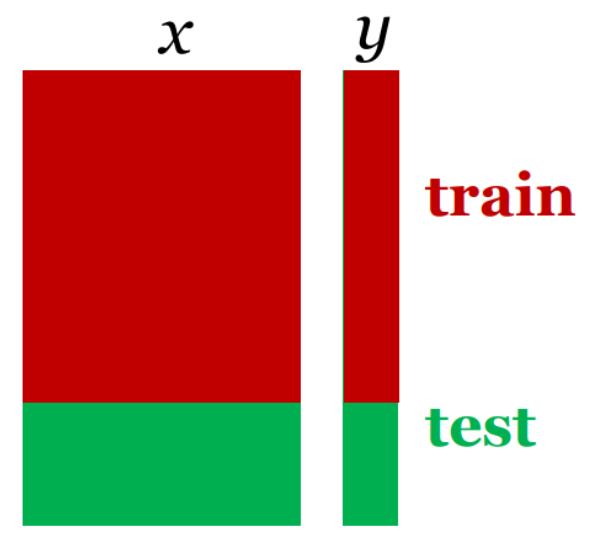


Evaluating on the new data tells us how well can our trained model ***generalize***.

Question: But where do we get new data?

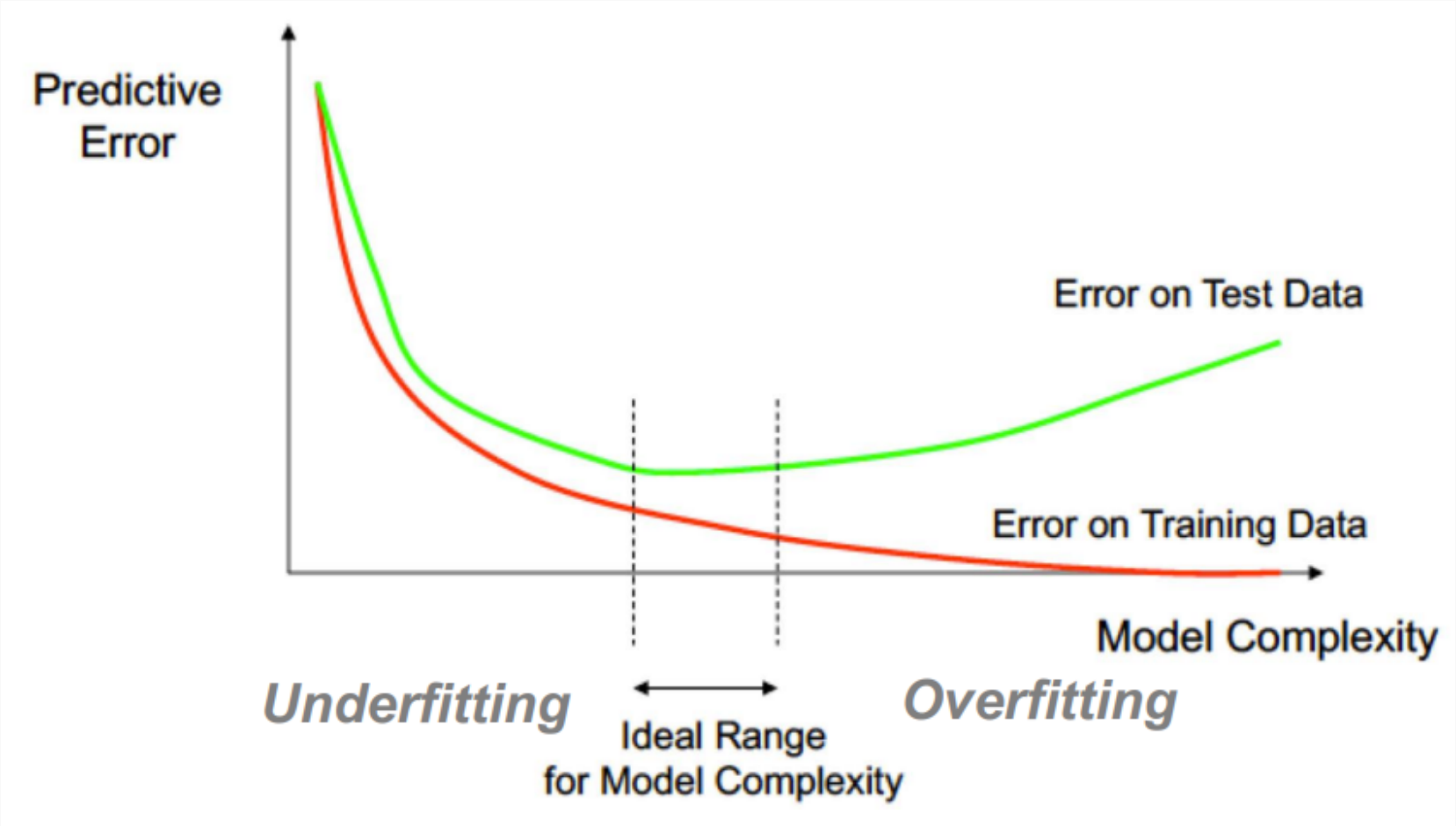
The Train/Test Split

Often times, data collection is done once. Collecting new data to evaluate your model may be prohibitively expensive or unethical. So we create training and testing data from a single dataset.



Comparing performance of model on training and testing data can help us identify ***overfitting*** and ***underfitting***. Can you see how?

The Train/Test Split



The Train/Validation/Test Split

We're often not just evaluating one 'best model' but choosing amongst many (different kNN regressors, different polynomials).

Which of the following option is preferable?

1. Train on training data, selecting the degree of polynomial, number of neighbours etc (**hyperparameters**) on training data, test on testing data.
2. Train on training data, selecting the degree of polynomial, number of neighbours etc on testing data, test on testing data.

The Train/Validation/Test Split

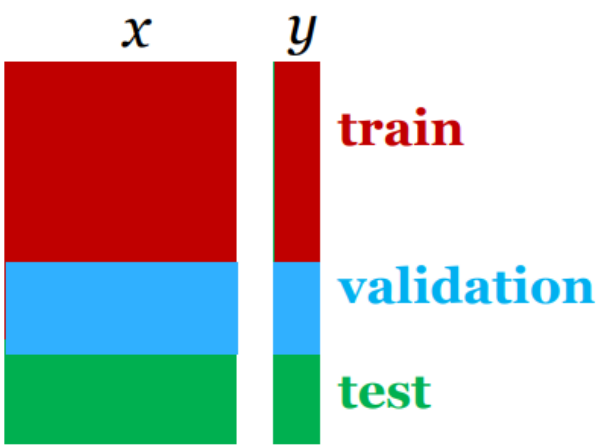
We're often not just evaluating one 'best model' but choosing amongst many (different kNN regressors, different polynomials).

Option: Fit on train, select on **validation**

1) Fit each model to **training** data
2) Evaluate each model on **validation** data
3) Select model with lowest **validation** error
4) Report error on **test** set

Concerns

- Will train be too small?
- Make better use of data?



The Train/Validation/Test Split

We're often not just evaluating one 'best model' but choosing amongst many (different kNN regressors, different polynomials).

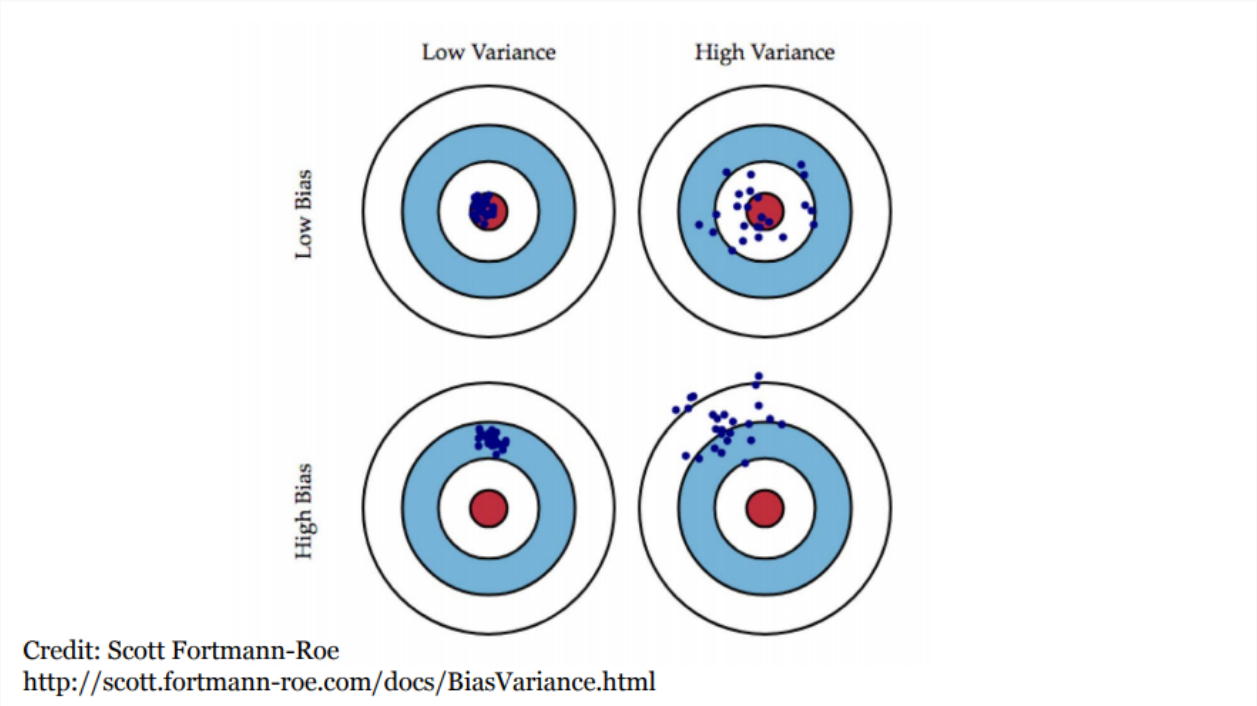
The art of creating new data out of old:

1. **k-fold** - divide data into k equally sized chunks, train on $k-1$ chunks, test on the left-out chunk; rotate the left-out chunk
2. **leave-one-out** - train on $N-1$, test on the left-out 1; rotate the left-out point
3. **bootstrap** - randomly sampling from the dataset, with replacement

Variance and Complexity

We've seen that hyperparameter selection is important to prevent underfitting and overfitting.

Simple models struggle to capture patterns in data and are insensitive to noise in the dataset. Complex models capture the trend as well as the noise in the data.



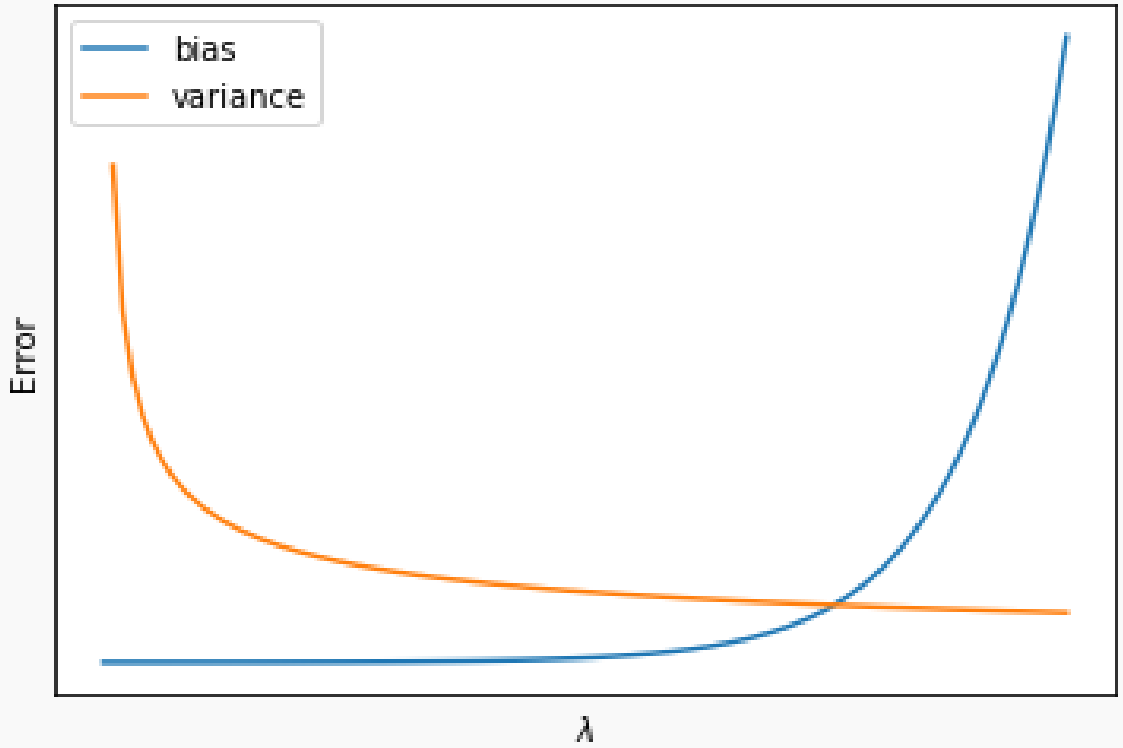
Credit: Scott Fortmann-Roe
http://scott.fortmann-roe.com/docs/BiasVariance.html

Example: Variance and Complexity

Let’s look at an example at the relationship between model complexity and variance.

Variance and Generalization Error

We can intuitively reason about the relationship between variance, bias and generalization error.



That is, to reduce error, we can reduce bias or variance, knowing that decreasing one increases the other.

Variance and Generalization Error

Formally, we can show that, the generalization error, as measured by the mean squared error, can be decomposed as

$$\mathit{MSE} = \mathit{noise}(\mathbf{x}) + \mathit{bias}(f(\mathbf{x}))^2 + \mathit{variance}(f(\mathbf{x}))$$

That is, to reduce error, we can reduce bias or variance, knowing that decreasing one increases the other.

Variance Reduction: Regularization

Intuitively, we see that high variance is caused by model complexity. Often, ***variance reduction*** techniques involve limiting the complexity of the model.

Idea: assign a penalty on model complexity by constraining norm of weights or parameters **w**

1. Ridge regression (L2-norm penalization)

$$\min_{\mathbf{w}} \text{MSE}(\mathbf{w}) + \lambda ||\mathbf{w}||_2^2$$

2. Lasso regression (L1-norm penalization)

$$\min_{\mathbf{w}} \text{MSE}(\mathbf{w}) + \lambda ||\mathbf{w}||_1$$

How do the penalty terms change the loss function? What is the role of λ?

Variance Reduction: Bagging

In contrast to regularization, we may also reduce variance by training a set or ***ensemble*** of complex models and then averaging their outputs during prediction.

The Bagging (Bootstrap Aggregate) model/algorithm:

1. create k number of bootstrap samples from your training dataset
2. train a model on each of the samples
3. when predicting on a new input x, average the predictions of the k models

How does averaging reduce variance?

Exercise: Variance Reduction

Let's try implementing regularization and bagging in `sklearn`.