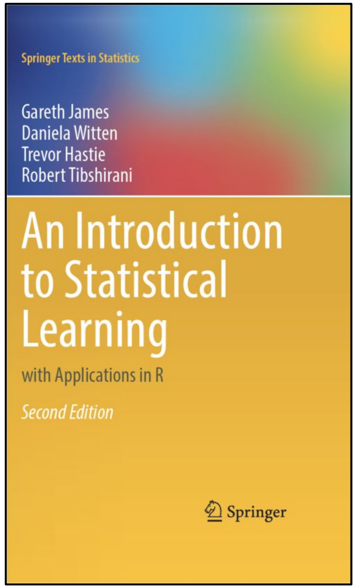
Overfitting and the Bias-Variance Decomposition

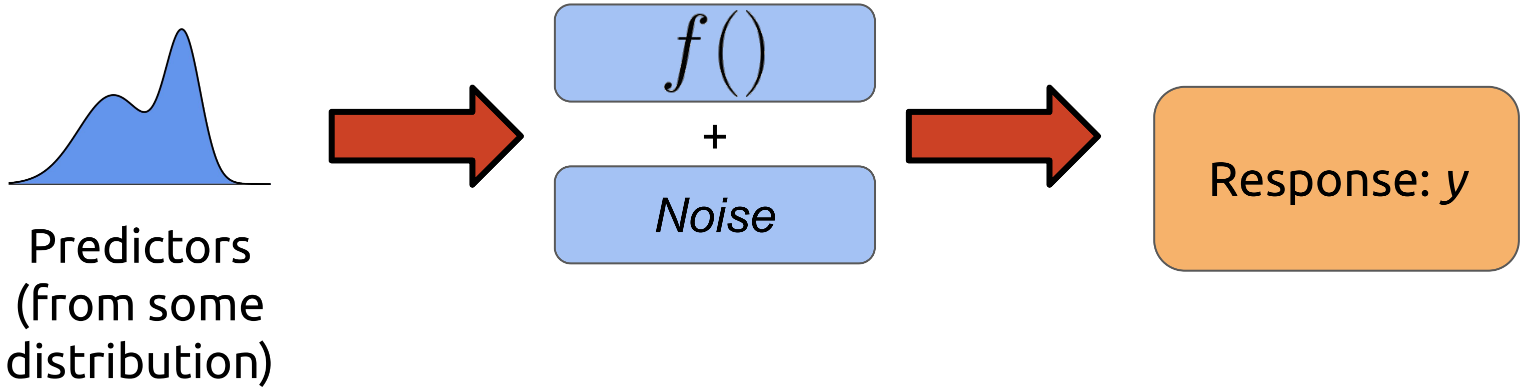
Recommended Reading:

An Introduction to Statistical Learning, Sections 2.2.2 and 6.2

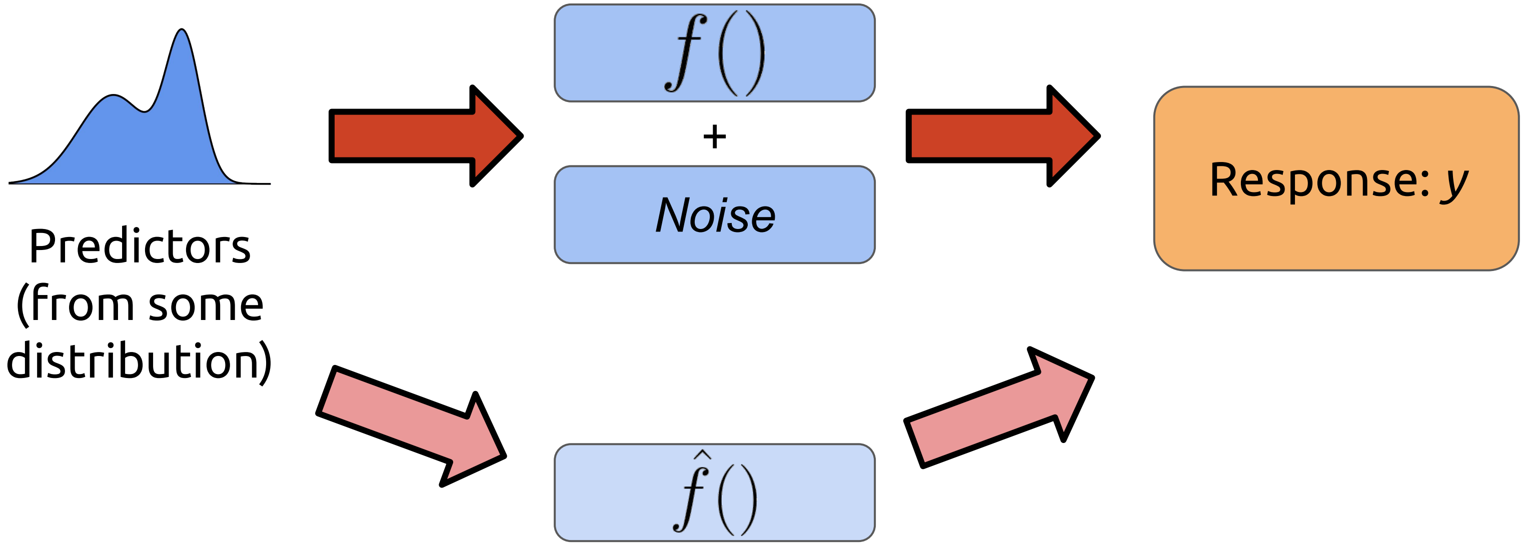
Download it for free here:
https://www.statlearning.com/



Recall: Supervised Learning - Setup

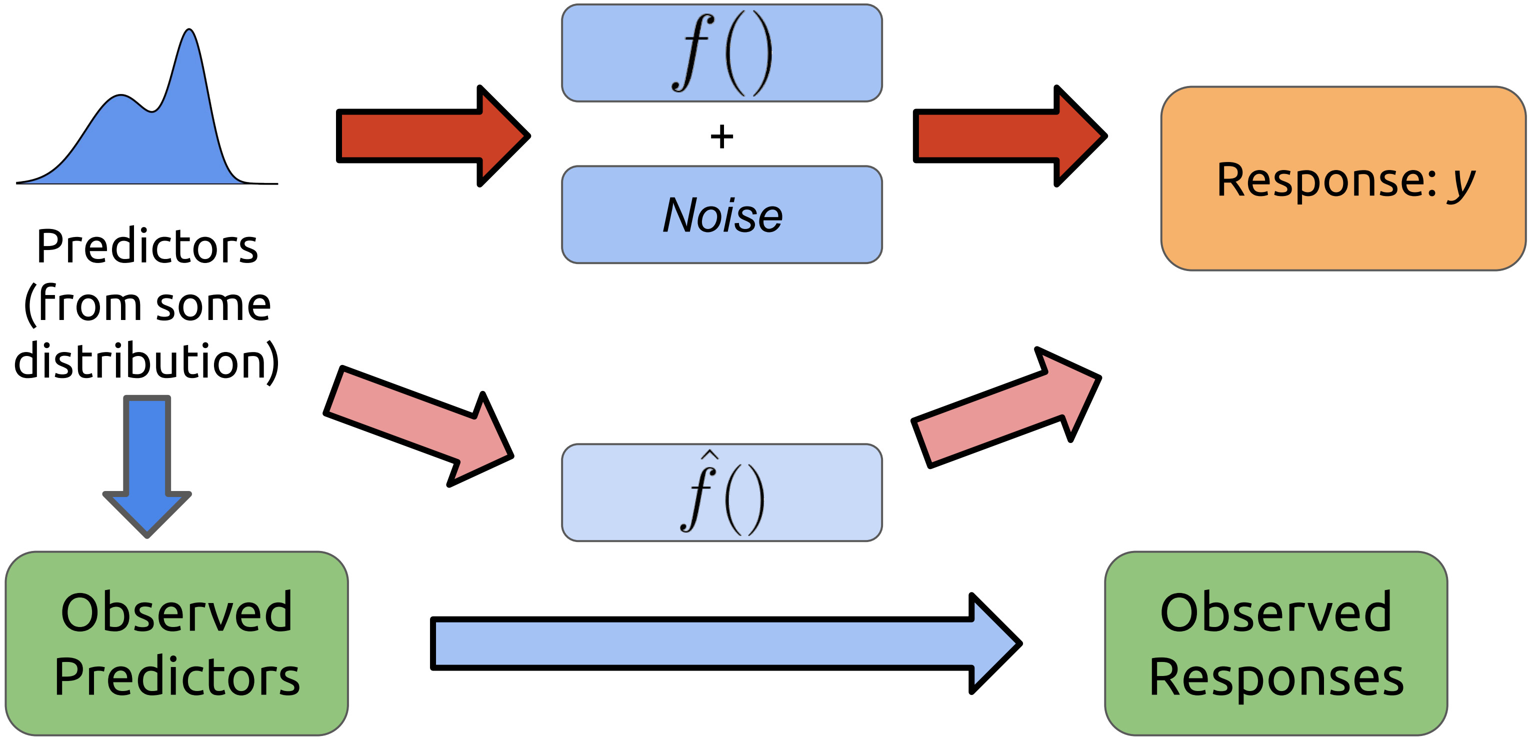


Supervised Learning - Goals



Goal: Choose a function so that the our predictions are close (on average) to the true values.

Supervised Learning - How



Supervised Learning - Goals

To measure how “good” our model is, we need some way to measure “error” (eg. mean squared error).

Our goal is to minimize the expected loss over *new* data.

Important: We are not trying to minimize loss over the observed data (which is often very easy to do), but to minimize the *generalization error* - the performance on unseen data.

Disclaimer:

These slides use polynomials because they are easy to visualize and gauge the complexity.

To apply the concepts from these slides, remember that higher degree polynomial corresponds to a more complex or more flexible model.

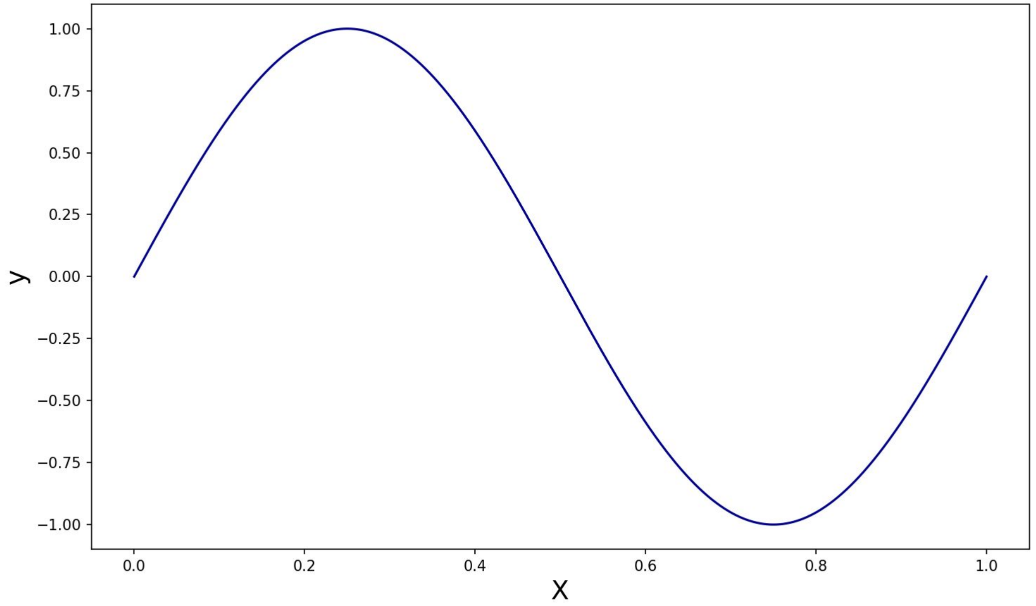
Example:

Let’s say that we are trying to fit data that comes from this data generation process:

X: Uniform distribution on the interval [0,1]

$Y = \sin(2\pi X) + \varepsilon$,

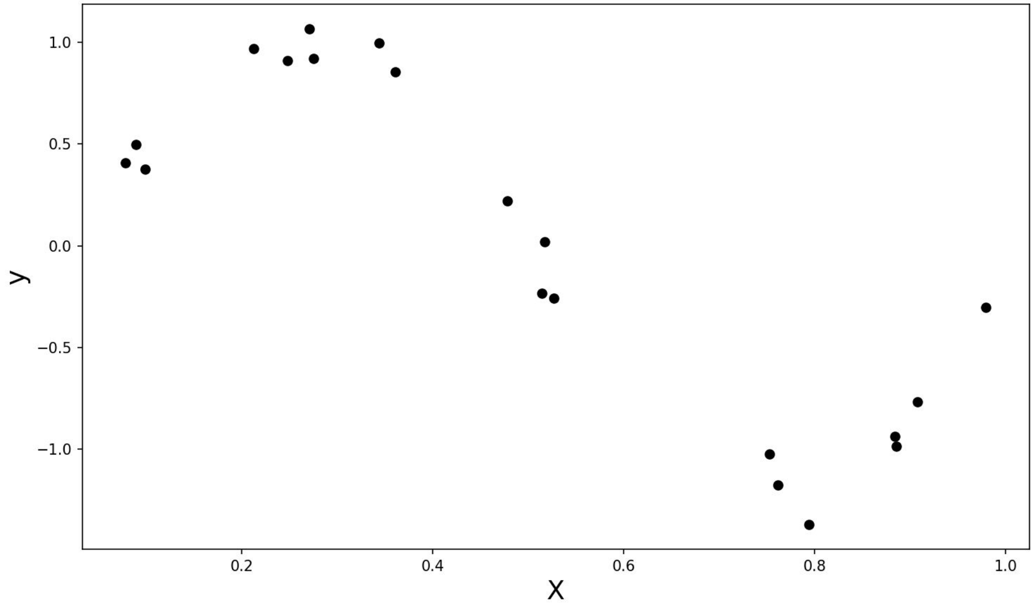
$\varepsilon \sim N(0, 0.2)$



Example:

To attempt to fit this, we have a sample of 20 points.

We’ll stick to polynomial regression to try and fit this data.

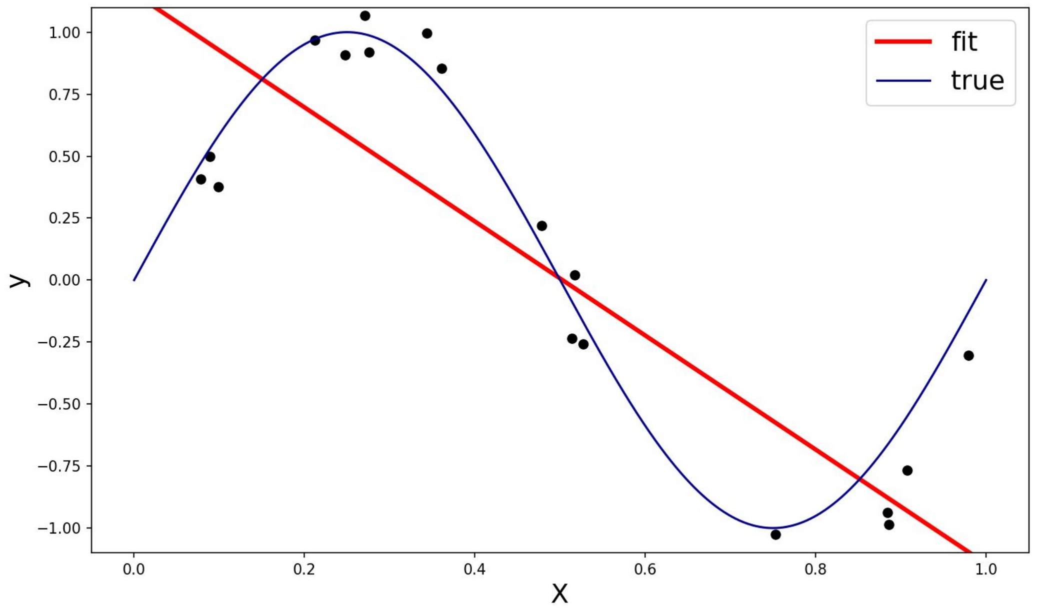


Example:

We could do a simple (degree 1) model.

$MSE_{train} = 0.198$

$MSE_{generalization} = 0.250$



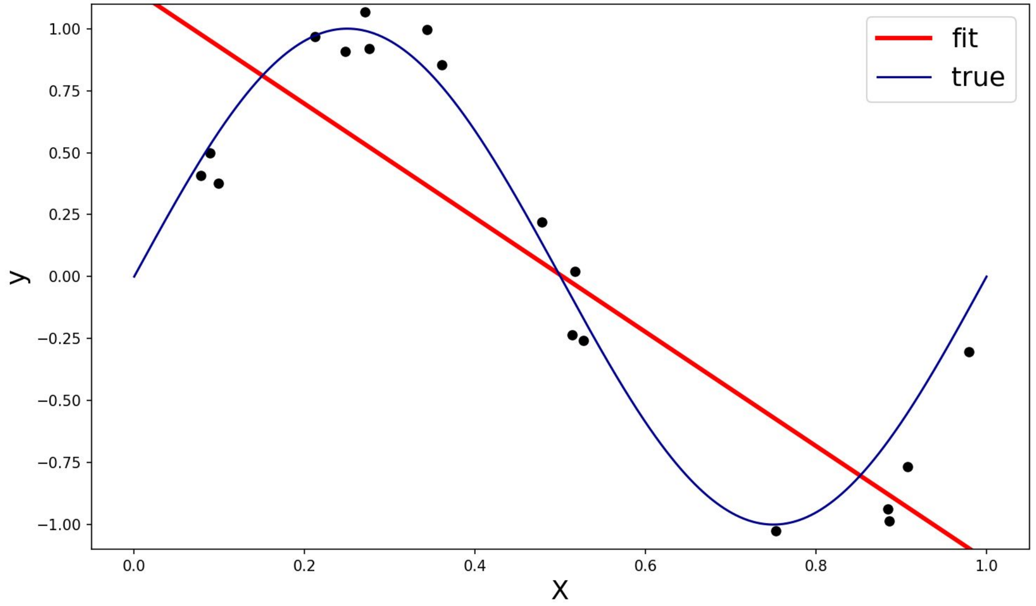
Example:

This model might be a bit **underfit** - it is not really capturing as much of the true trend as possible.

We could do a simple (degree 1) model.

$MSE_{train} = 0.198$

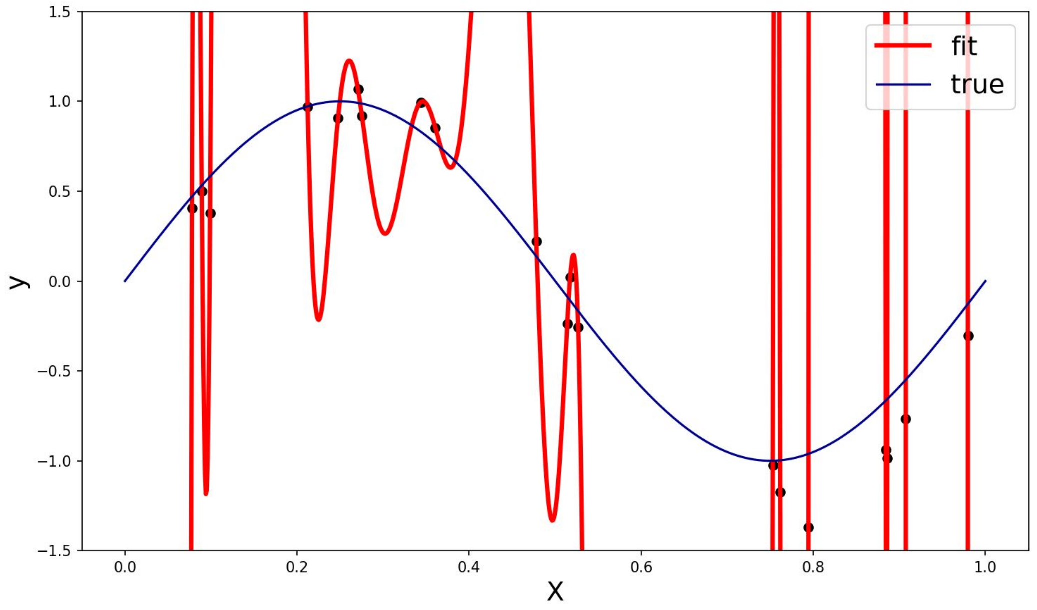
$MSE_{generalization} = 0.250$



Example:

Let’s try a degree 30 model.

$\text{MSE}_{\text{train}} = 0$

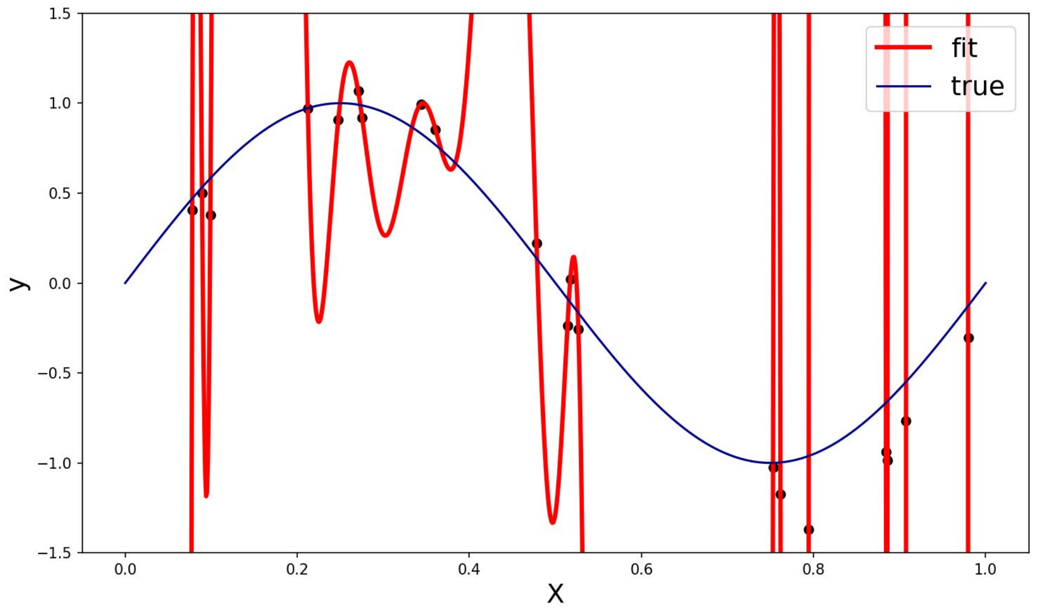


Example:

Let’s try a degree 30 model.

$\text{MSE}_{\text{train}} = 0$

$\text{MSE}_{\text{generalization}} = 92{,}204{,}412{,}974$



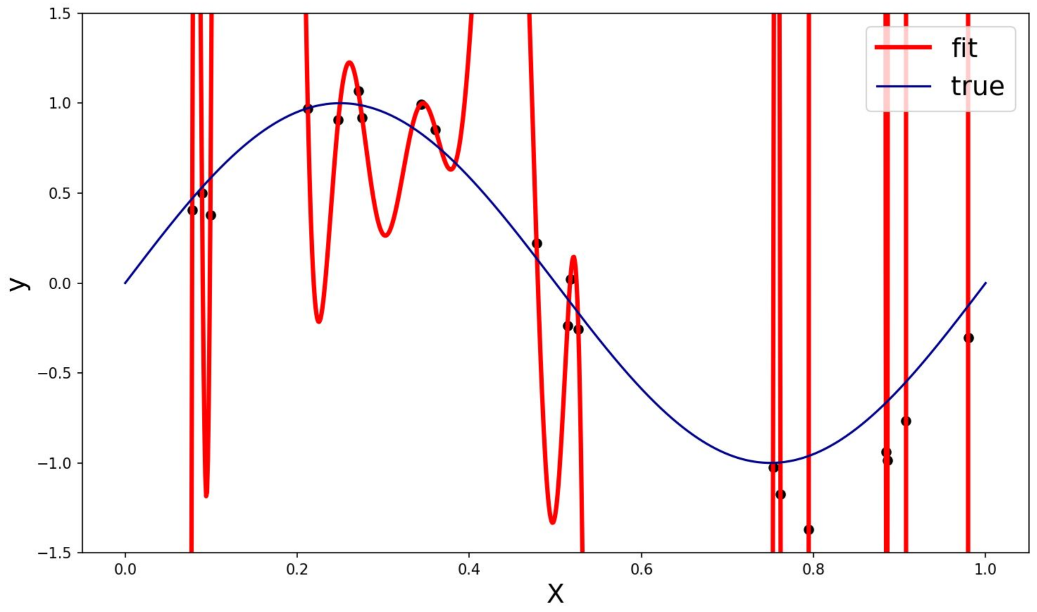
Example:

This model is **overfit** - it fits the training data really well, but does a terrible job of generalizing.

Let's try a degree 30 model.

$\text{MSE}_{\text{train}} = 0$

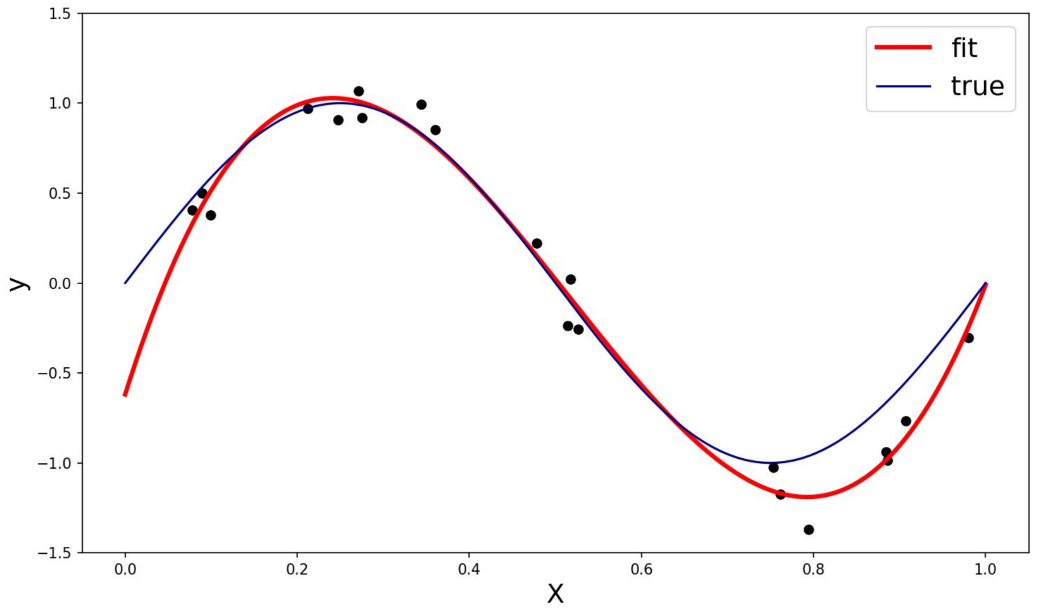
$\text{MSE}_{\text{generalization}} = 92{,}204{,}412{,}974$



Example:

To try and mitigate overfitting, we can choose a simpler model.

Let’s try a degree 3 model.



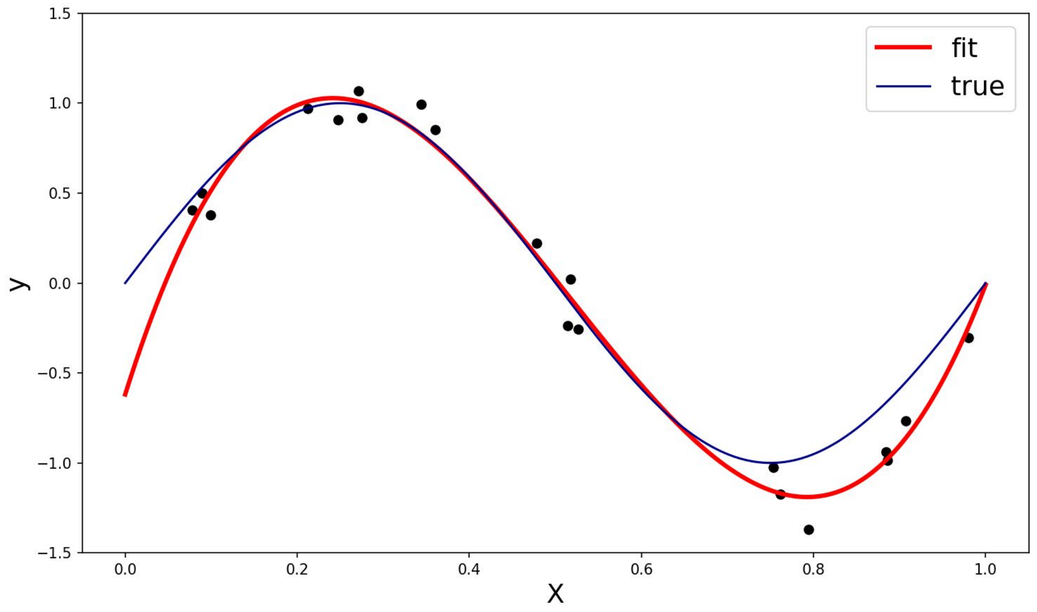
Example:

To try and mitigate overfitting, we can choose a simpler model.

Let’s try a degree 3 model.

$\text{MSE}_{\text{train}} = 0.011$

$\text{MSE}_{\text{generalization}} = 0.068$



Moral

The goal of supervised learning is to find a model which has low generalization error.

Moral

The goal of supervised learning is to find a model which has low generalization error.

Flexible models can be quite good at fitting the training data, but can do very poorly at generalizing to unseen data.

Moral

The goal of supervised learning is to find a model which has low generalization error.

Flexible models can be quite good at fitting the training data, but can do very poorly at generalizing to unseen data.

One strategy to avoid overfitting is to choose a simpler model.

Moral

The goal of supervised learning is to find a model which has low generalization error.

Flexible models can be quite good at fitting the training data, but can do very poorly at generalizing to unseen data.

One strategy to avoid overfitting is to choose a simpler model.

A way to force a simpler model is **regularization**.

Bias-Variance and Regularization

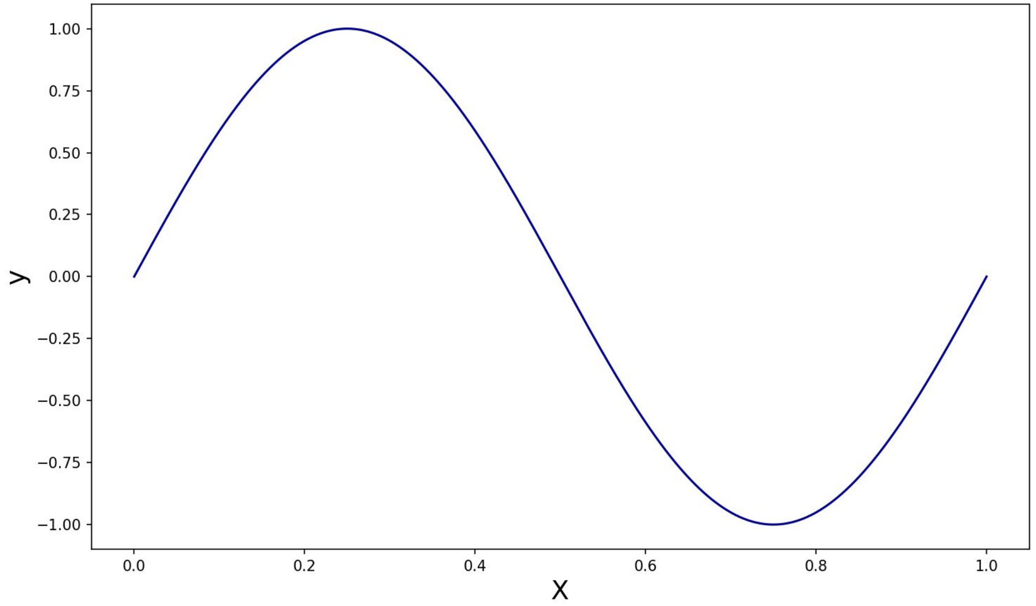
Example:

Let’s say that we are trying to fit data that comes from this data generation process:

X: Uniform distribution on the interval [0,1]

Y = sin(2πX) (no noise)

But, we’re only allowed two points in our training data.

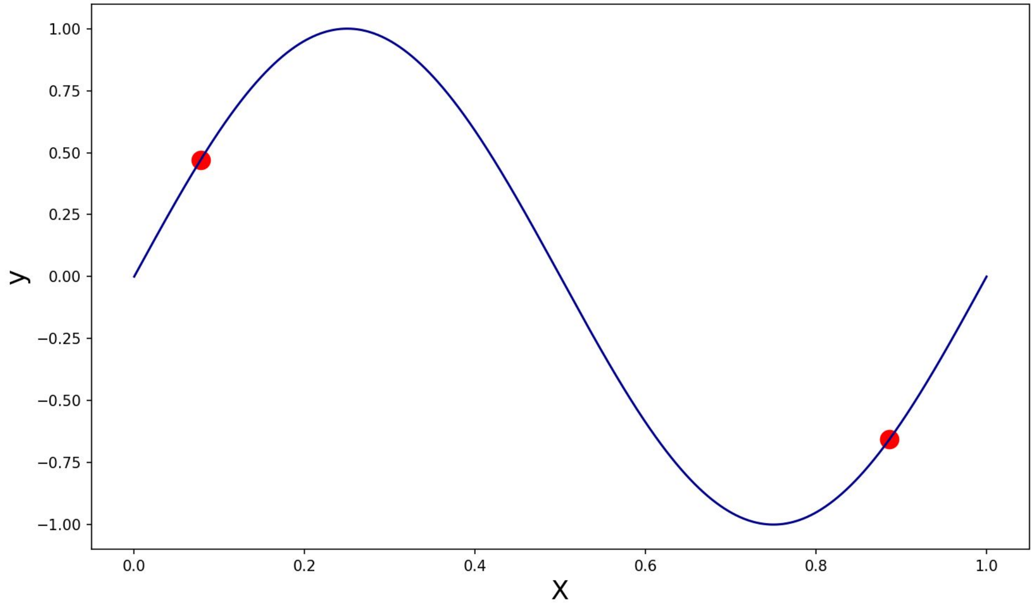


Example:

For example, here is one possible training set.

I’m going to try and choose between two models:

1. A constant only model.
2. A degree 1 linear model.



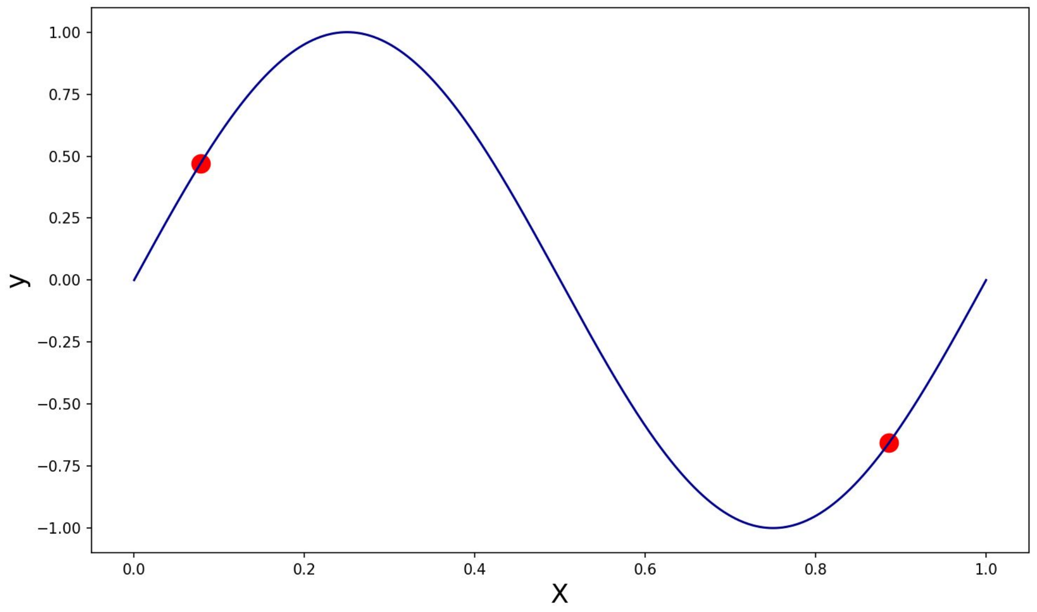
Example:

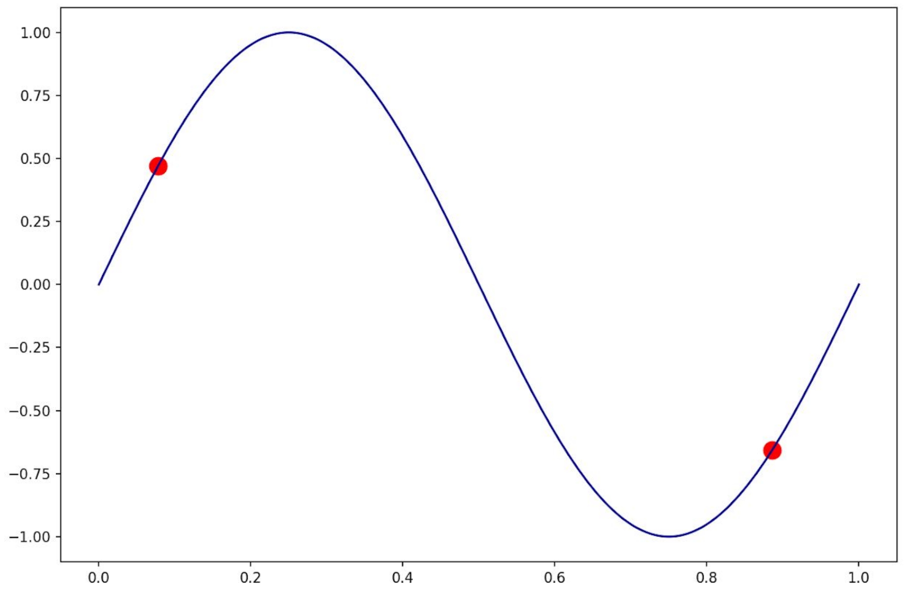
For example, here is one possible training set.

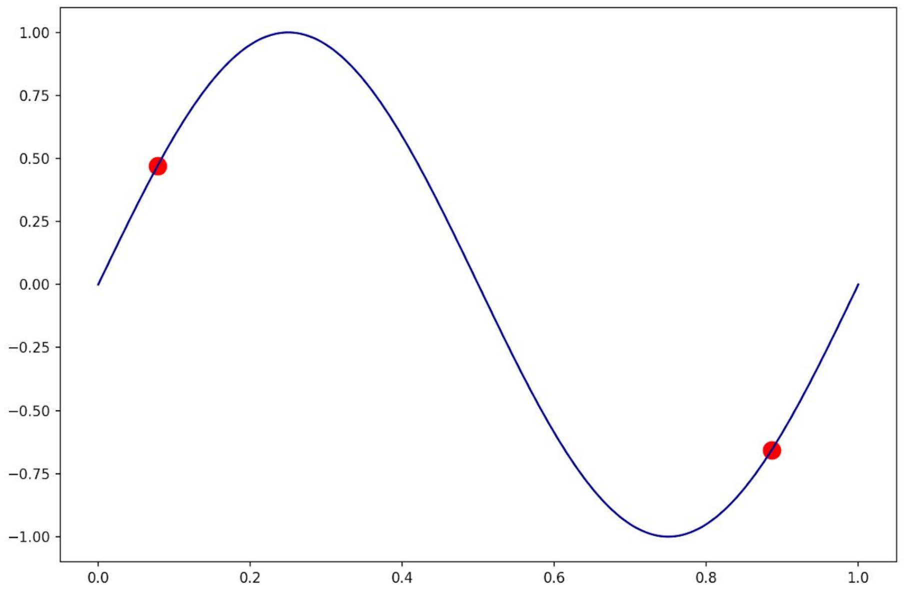
I’m going to try and choose between two models:

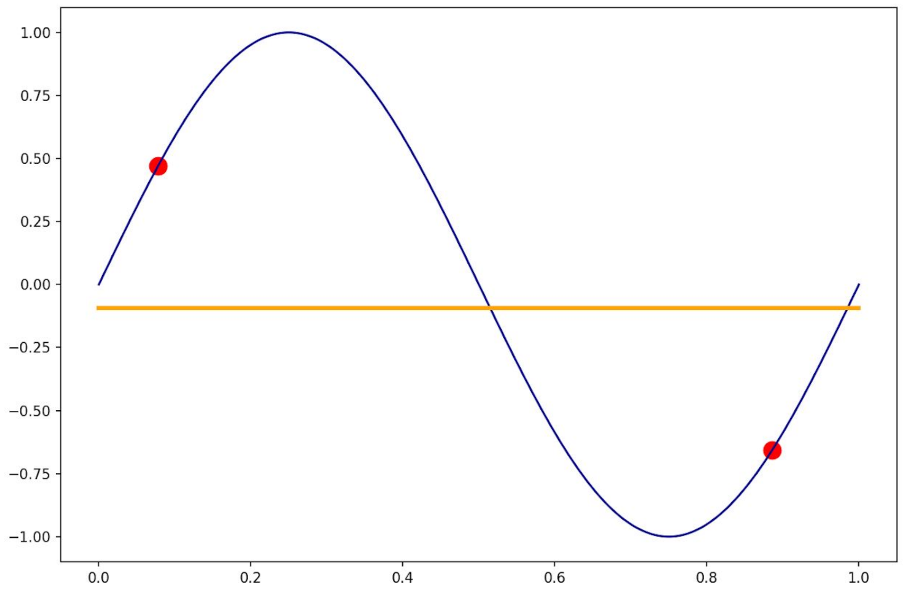
1. A constant only model.
2. A degree 1 linear model.

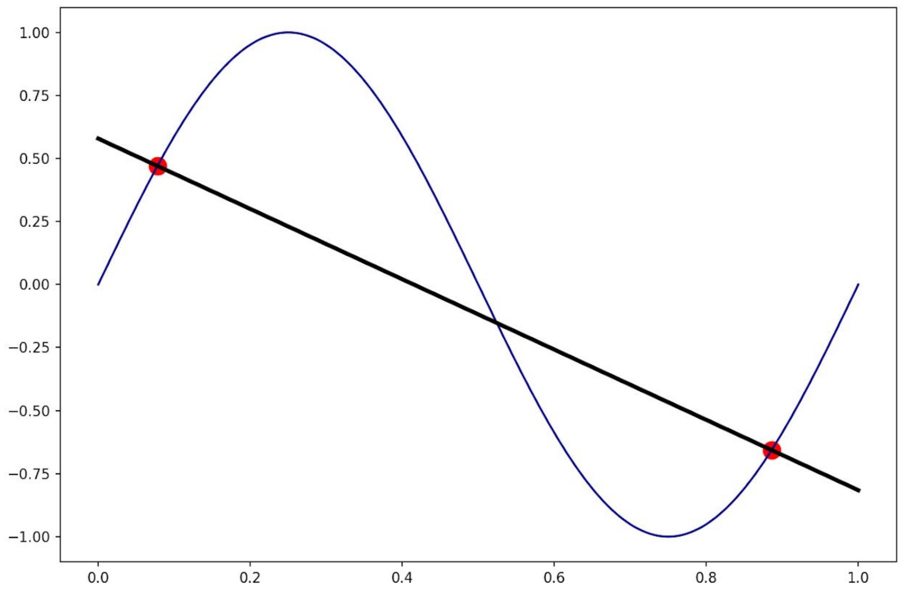
Which do you think will have lower generalization MSE (over all possible samples)?




1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

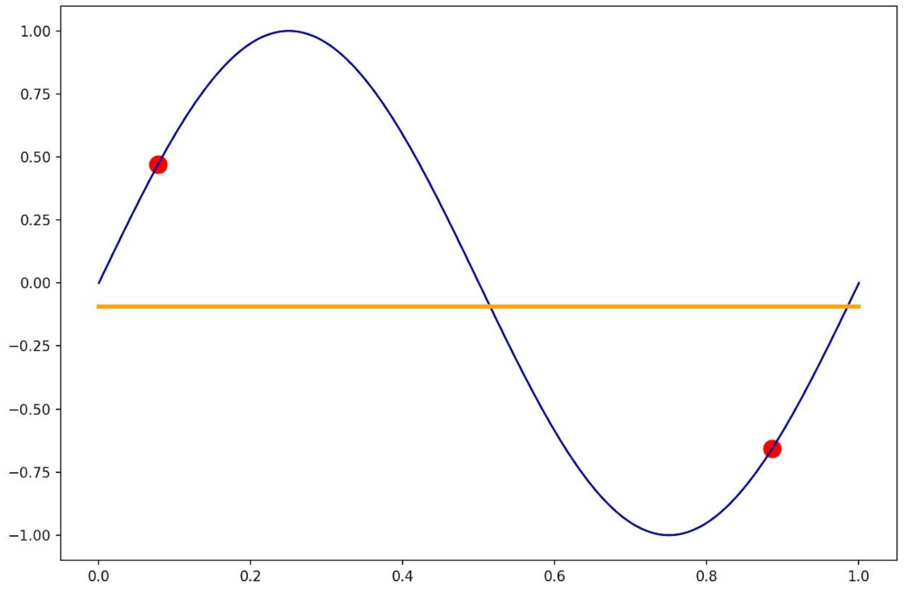

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0



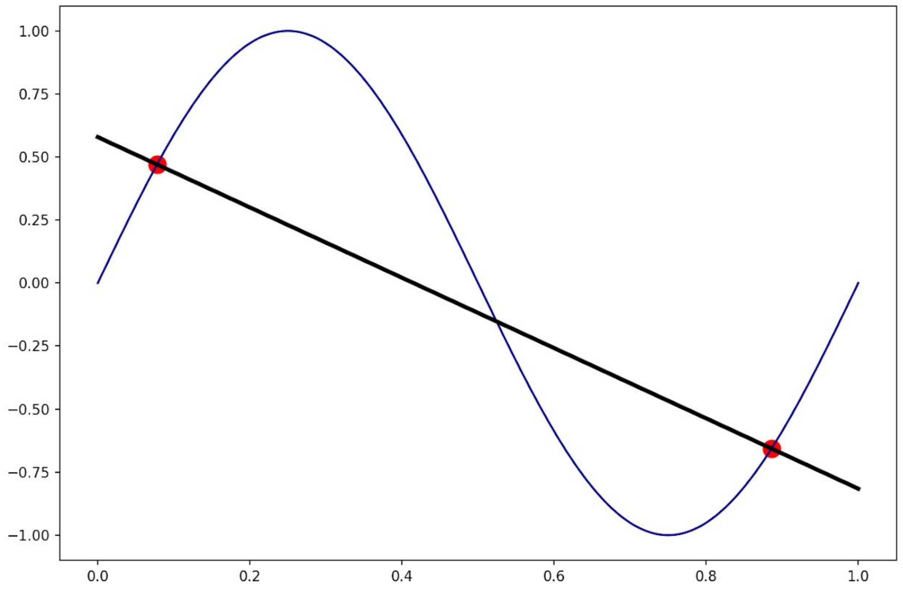


$\text{MSE}_{\text{generalization}} = 0.508$ $\text{MSE}_{\text{generalization}} = 0.232$

Here is the fit on one possible sample.



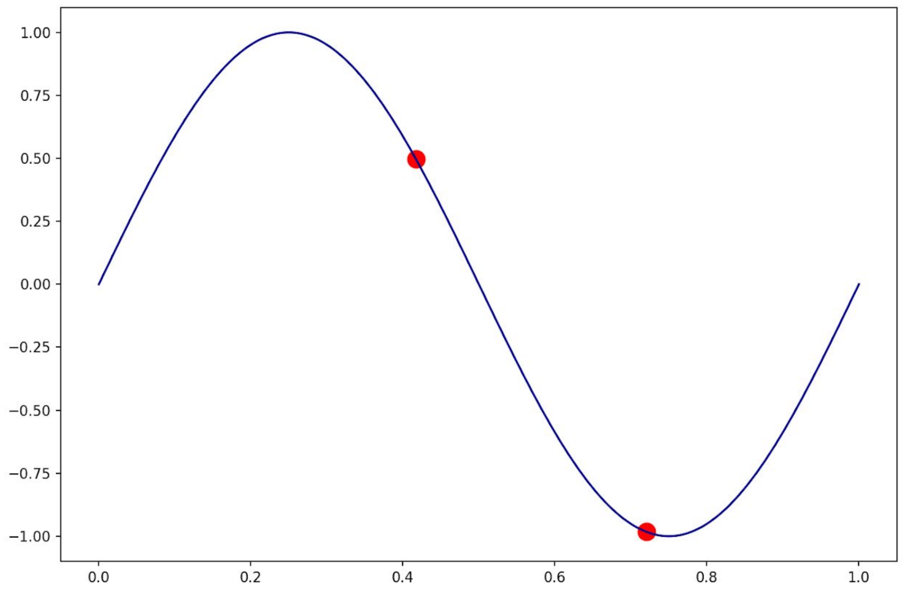
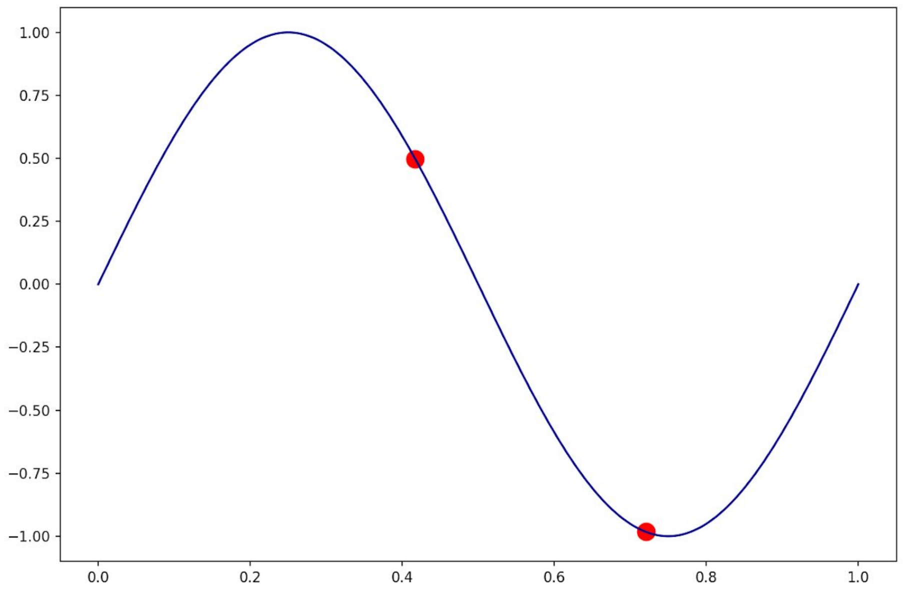
$MSE_{generalization} = 0.508$



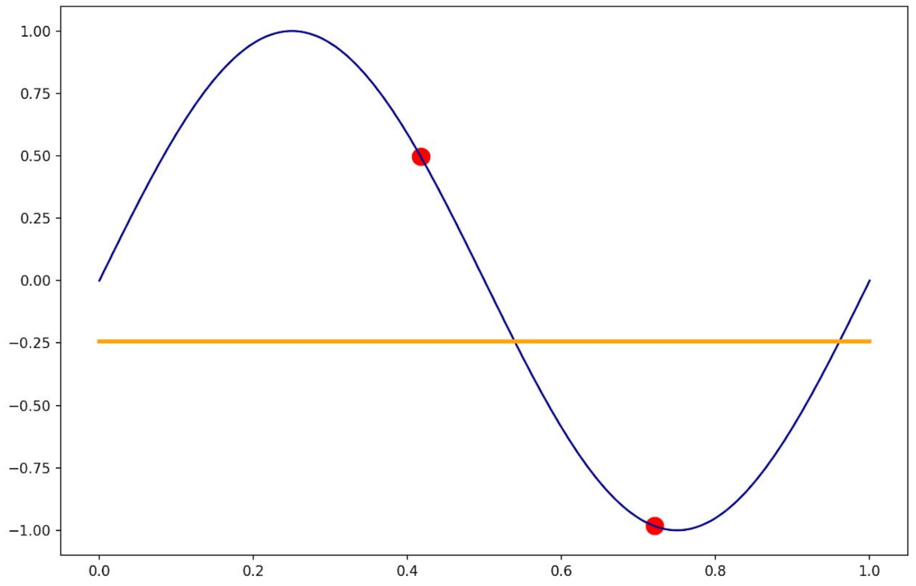
$MSE_{generalization} = 0.232$

Here is the fit on one possible sample.

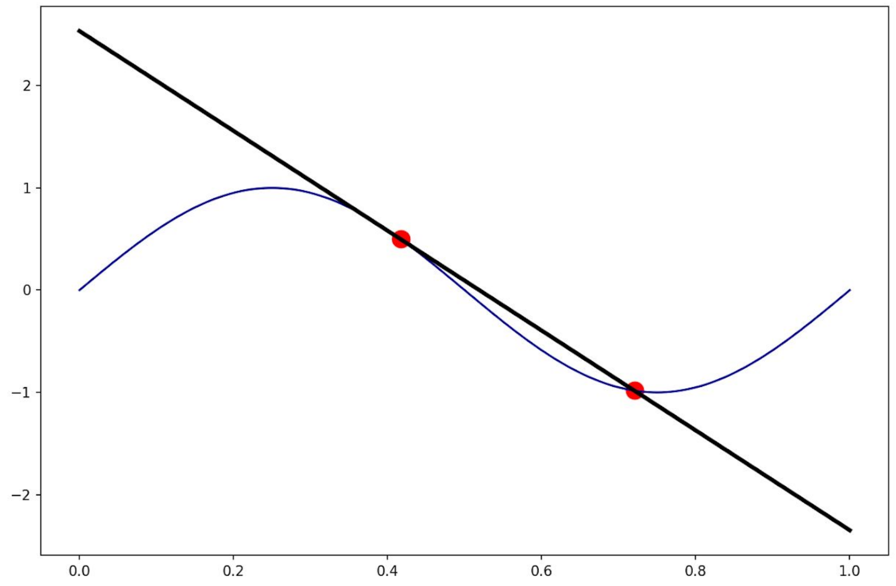
But, our goal is to find the model with the lower MSE over all possible samples, so we need to consider other possible samples.

What about on this sample?

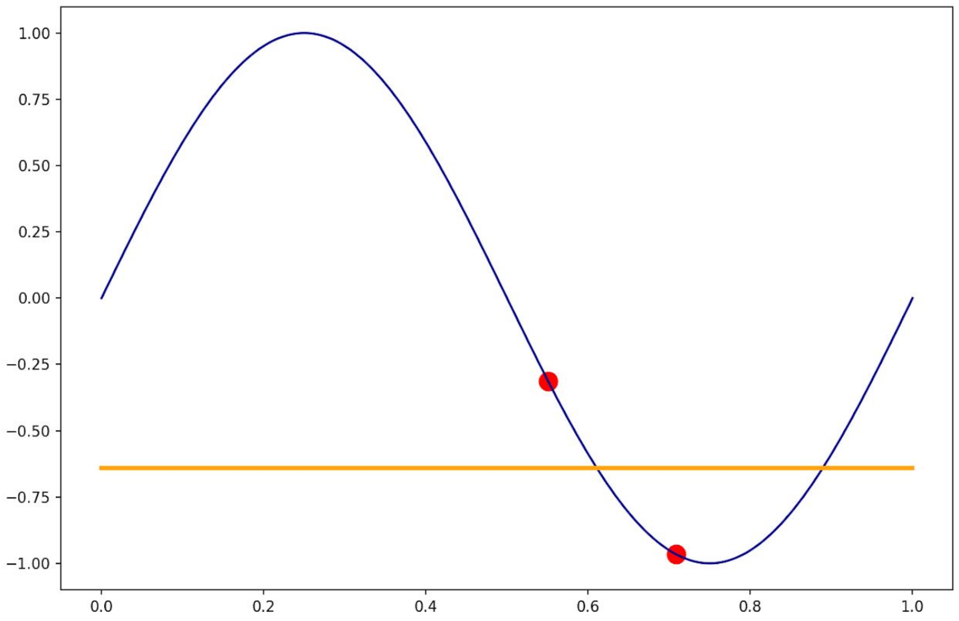


$\text{MSE}_{\text{generalization}} = 0.558$

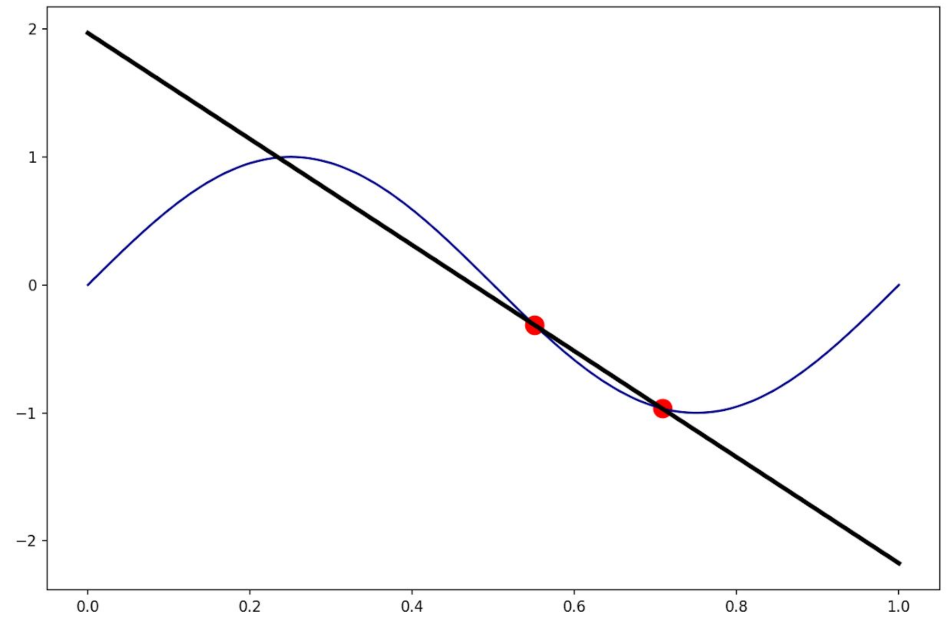


$\text{MSE}_{\text{generalization}} = 0.946$

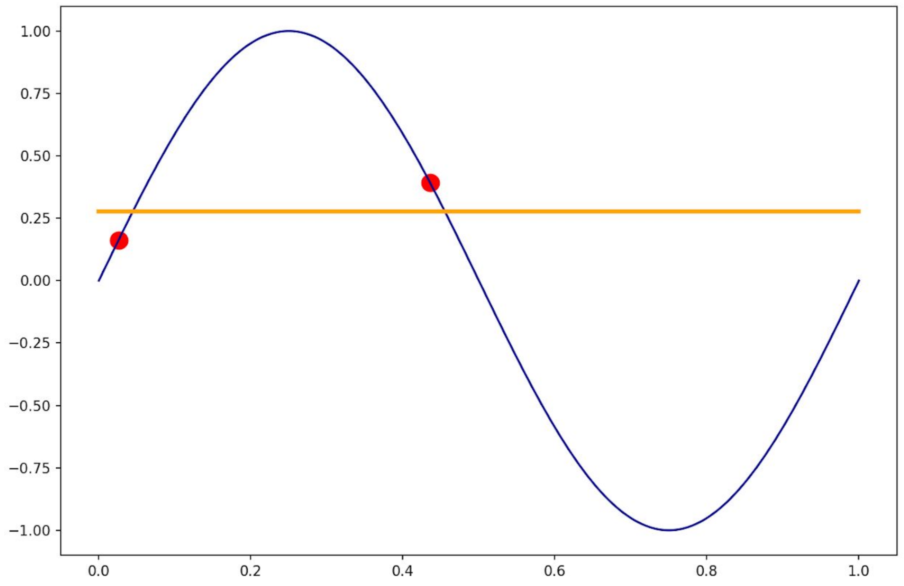
What about on this sample?

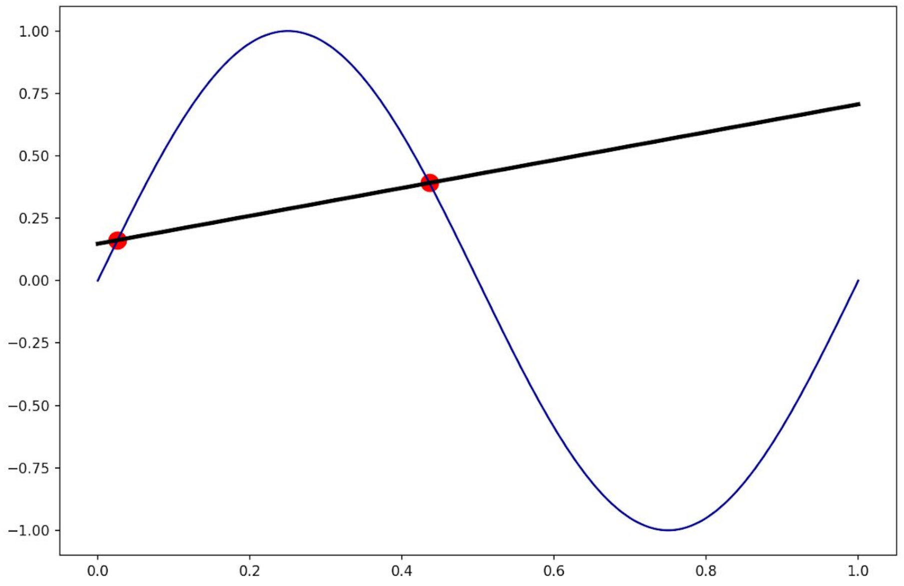


$MSE_{generalization} = 0.909$



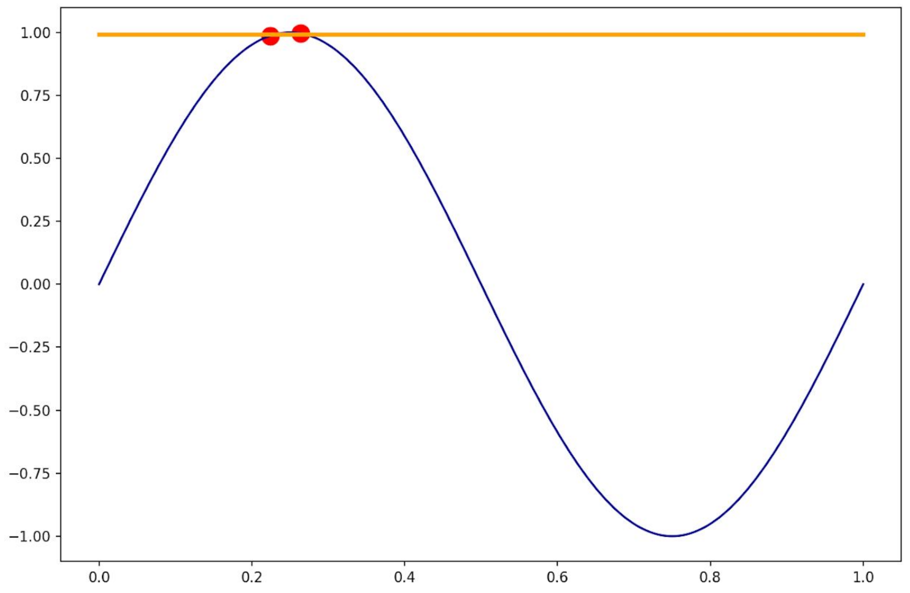
$MSE_{generalization} = 0.626$



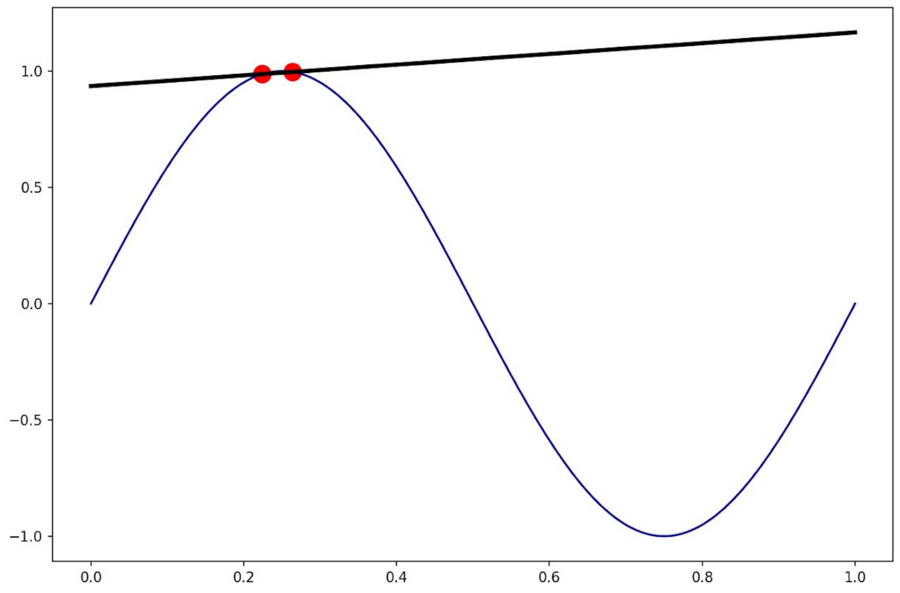


$MSE_{generalization} = 0.576$

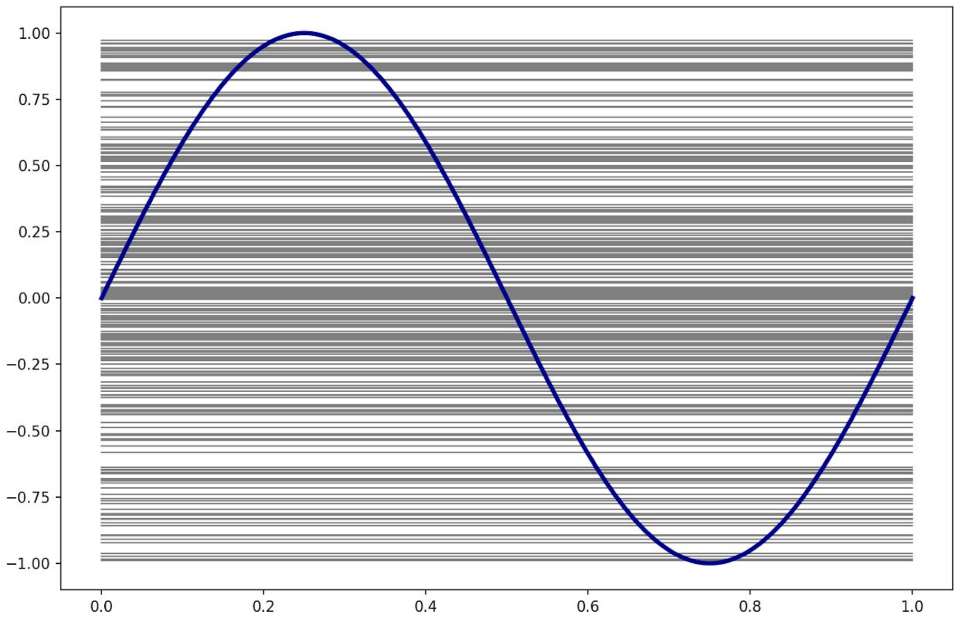
$MSE_{generalization} = 0.889$

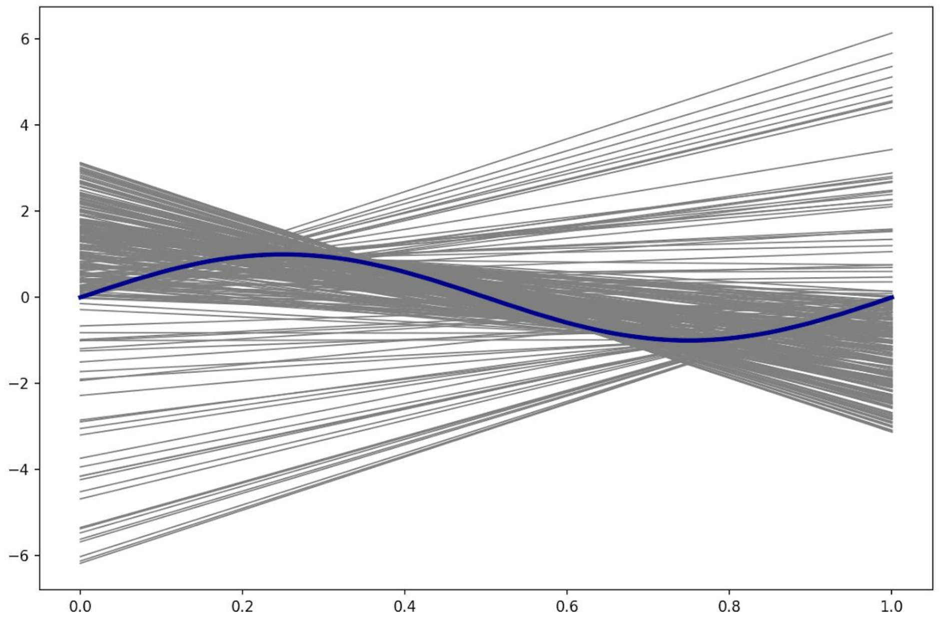


$\text{MSE}_{\text{generalization}} = 1.483$

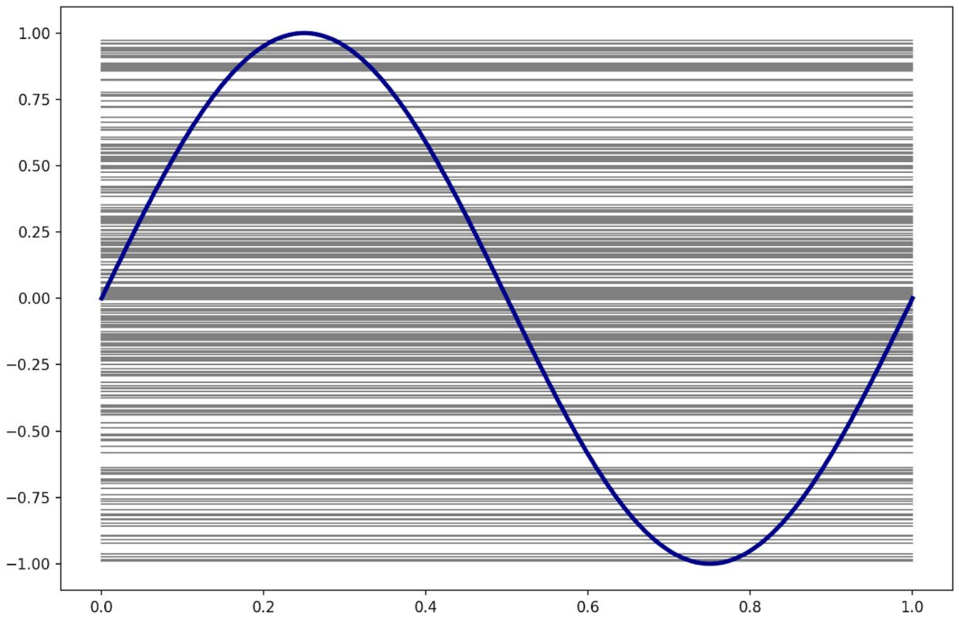


$\text{MSE}_{\text{generalization}} = 1.681$

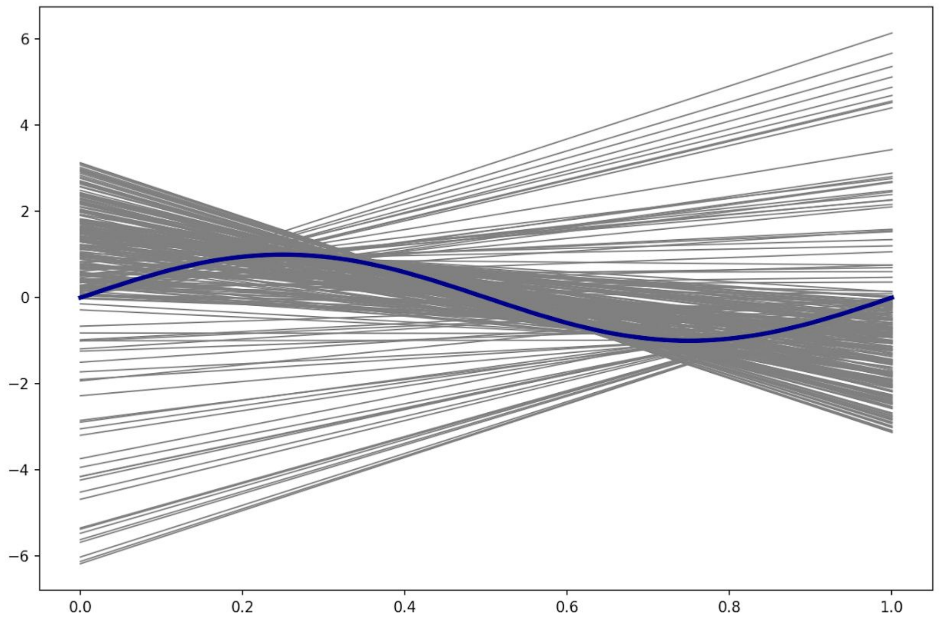




This is what the fits look like over 250 possible samples.

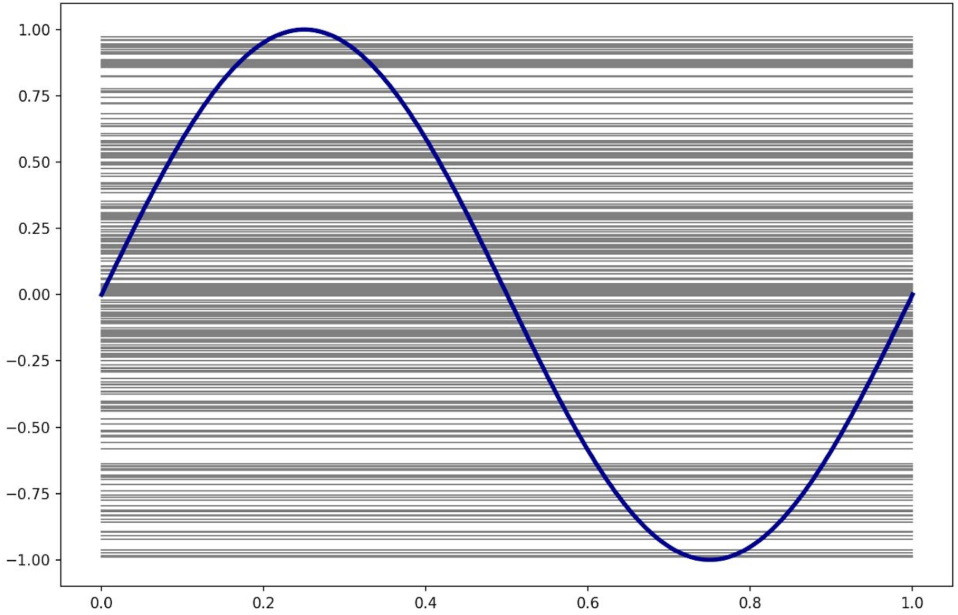


$MSE_{generalization} = 0.749$

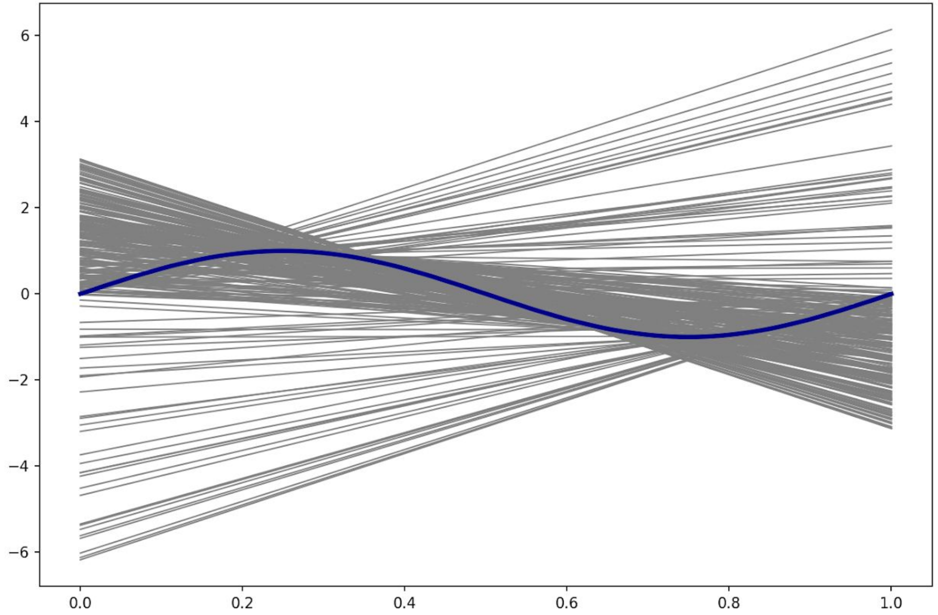


$MSE_{generalization} = 1.920$

This is what the fits look like over 250 possible samples.



$\mathrm{MSE}_{\text{generalization}} = 0.749$



$\mathrm{MSE}_{\text{generalization}} = 1.920$

Why is there such a big difference?

The Bias-Variance Decomposition

Bias: $E[\hat{f}(x)] - f(x)$

How far off can we expect to be, on average?

The Bias-Variance Decomposition

Bias: $E[\hat{f}(x)] - f(x)$

How far off can we expect to be, on average?

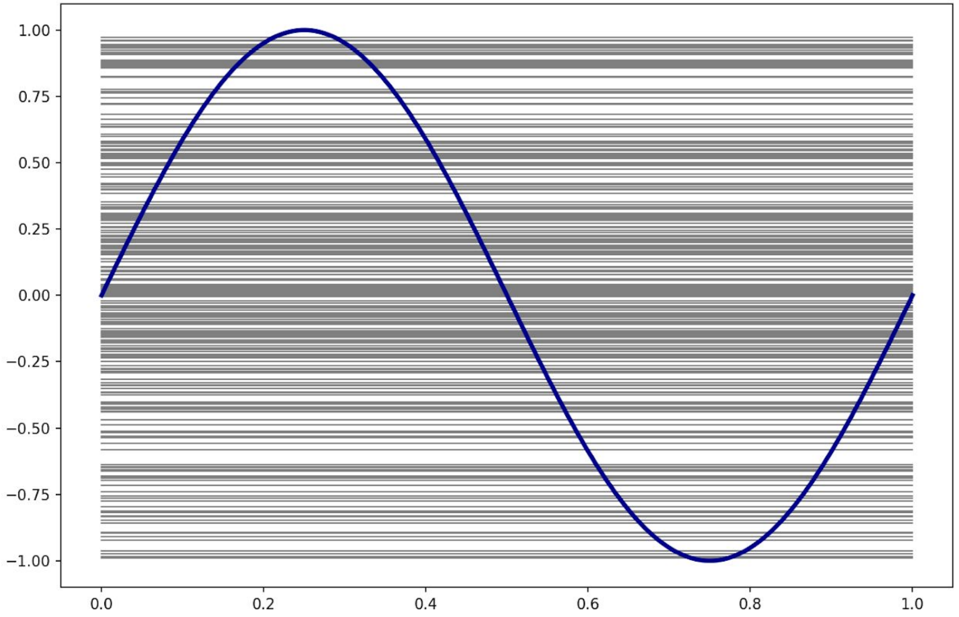
Variance: $E[(\hat{f}(x) - E[\hat{f}(x)])^2]$

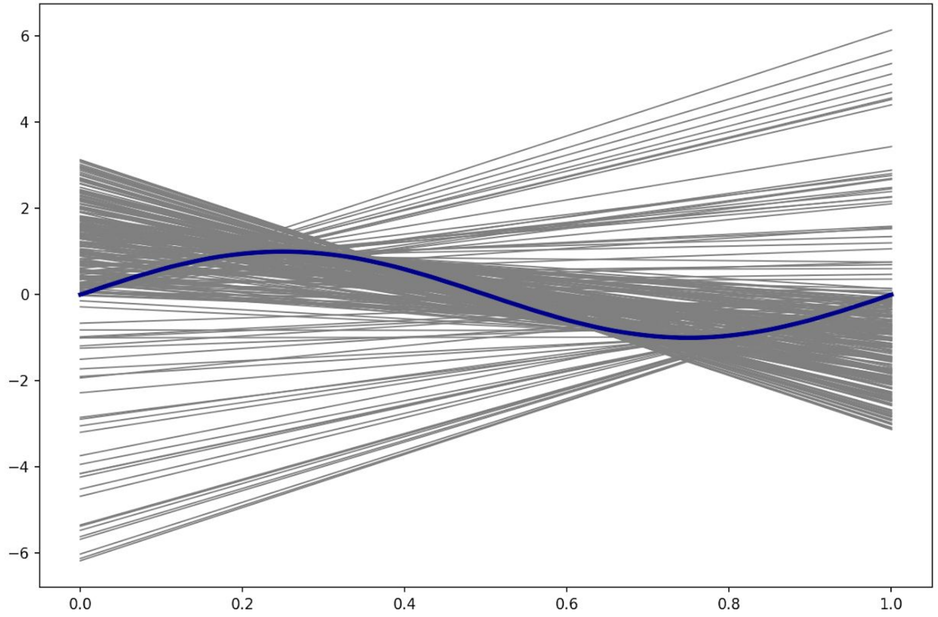
How much variability is there in the predictions (over all possible training sets?)

The Bias-Variance Decomposition

Expected MSE = Bias^2 + Variance + Irreducible Error

Irreducible Error is the noise in the system.





$\text{MSE}_{\text{generalization}} = 0.749$

$\text{Bias}^2 = 0.5$
Variance = 0.249

$\text{MSE}_{\text{generalization}} = 1.920$

$\text{Bias}^2 = 0.209$
Variance = 1.711

Regularization

Regularization is a method to reduce chances of overfitting.

It takes advantage of the fact that sometimes you can add a little bit of bias in order to greatly reduce variance (and get a better model overall).

Ridge Regression

Recall that linear regression fits a model of the form

$$\hat{f}(\vec{x}) = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \cdots + \beta_k x^{(k)}$$

By minimizing the residual sum of squares:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2$$

Ridge Regression

Ridge regression also fits a model of the same form:

$$\hat{f}(\vec{x}) = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \cdots + \beta_k x^{(k)}$$

But seeks to minimize a different objective function:

$$L = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2 + \lambda \sum_{i=1}^{k} \beta_k^2$$

Ridge Regression

Want small residuals

$$L = \boxed{\sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2} + \lambda \sum_{i=1}^{k} \beta_k^2$$

Ridge Regression

$$L = \overset{\text{Want small residuals}}{\overset{\downarrow}{\boxed{\sum_{i=1}^{n}(y_i - \hat{f}(\vec{x}_i))^2}}} + \overset{\text{But keep the coefficients small}}{\overset{\downarrow}{\boxed{\lambda \sum_{i=1}^{k} \beta_k^2}}}$$

Ridge Regression

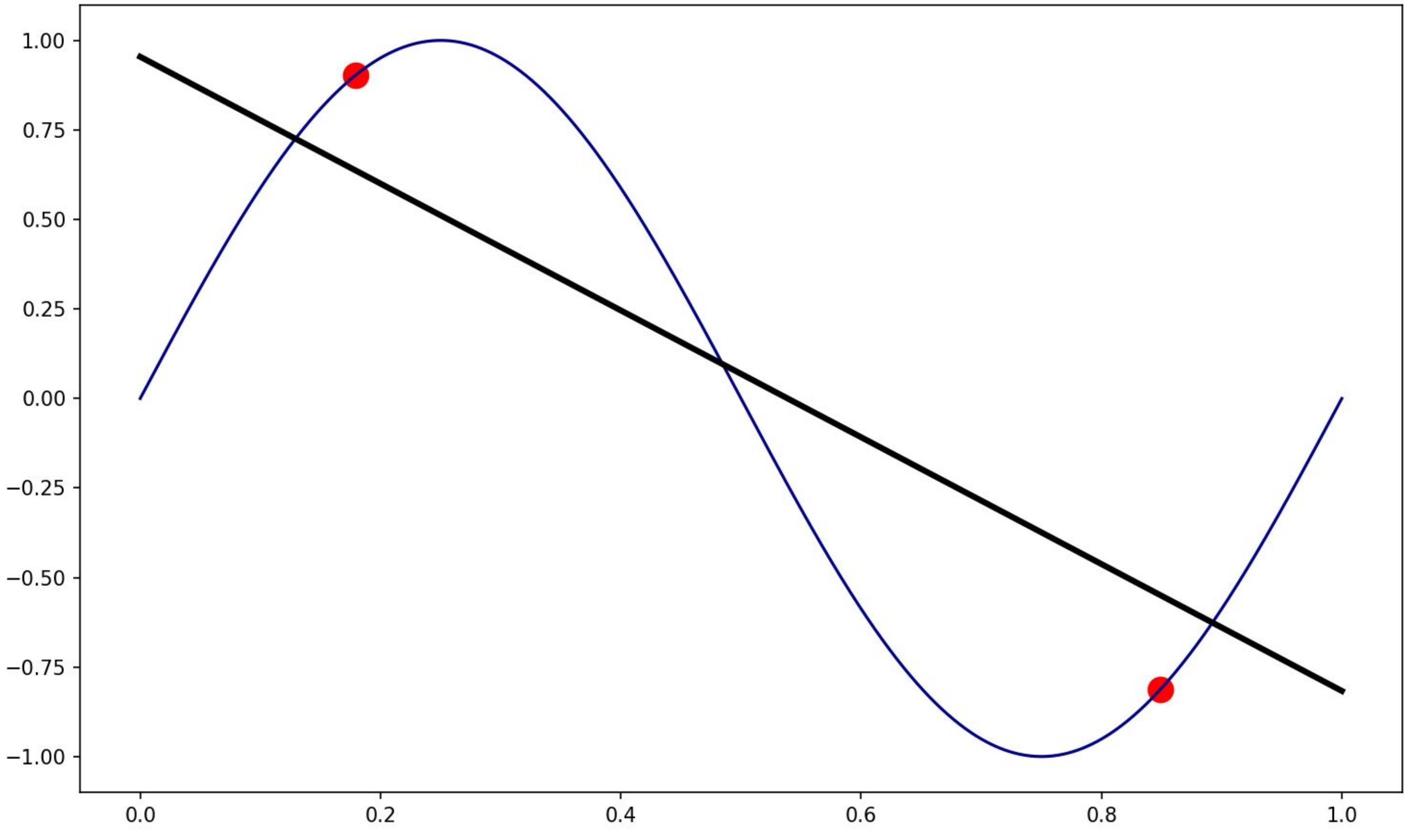
A hyperparameter
that must be set

$$L = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2 + \boxed{\lambda} \sum_{i=1}^{k} \beta_k^2$$

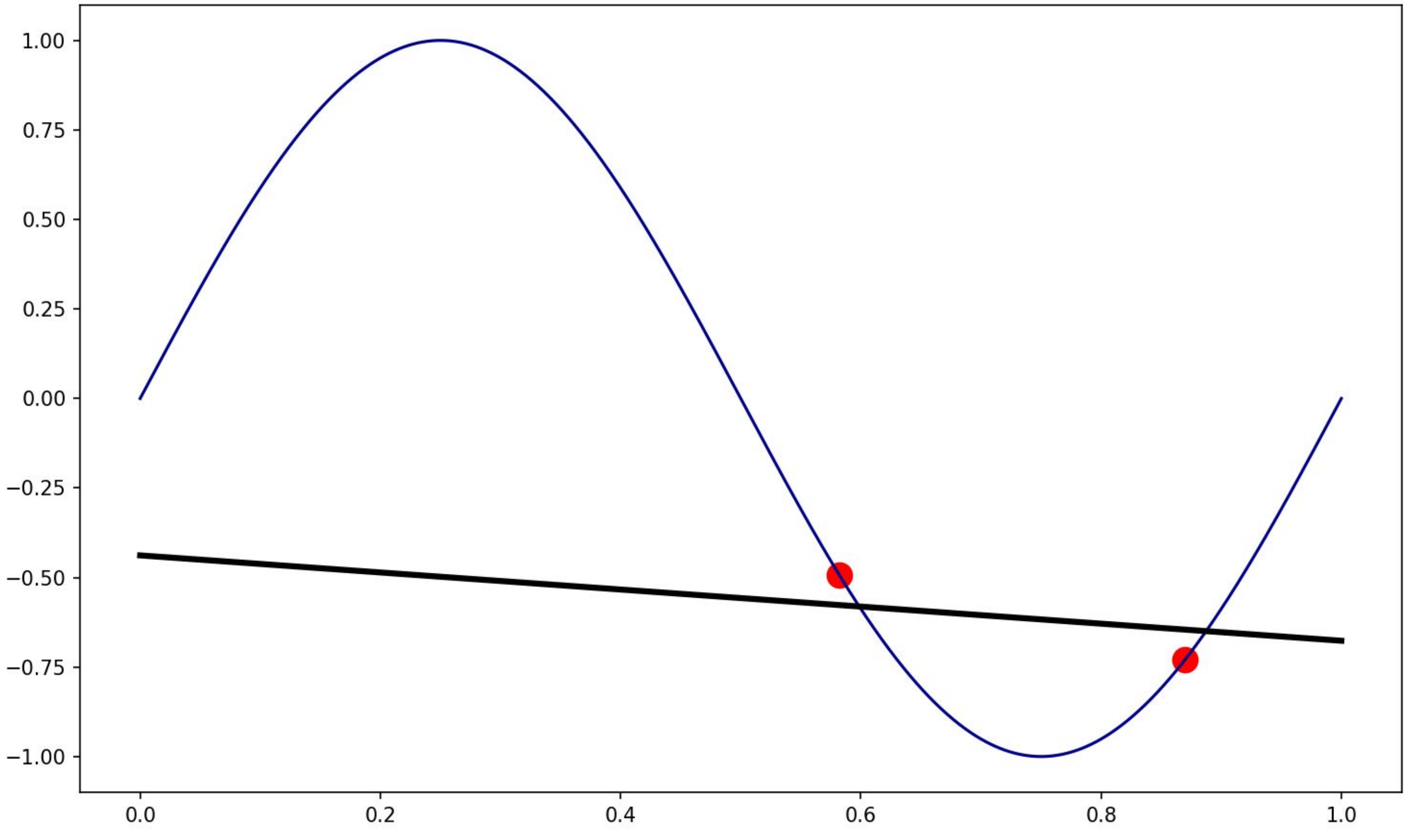
Regularization

Let’s return to our example with the sine function but use ridge regression models with $\lambda = 0.1$

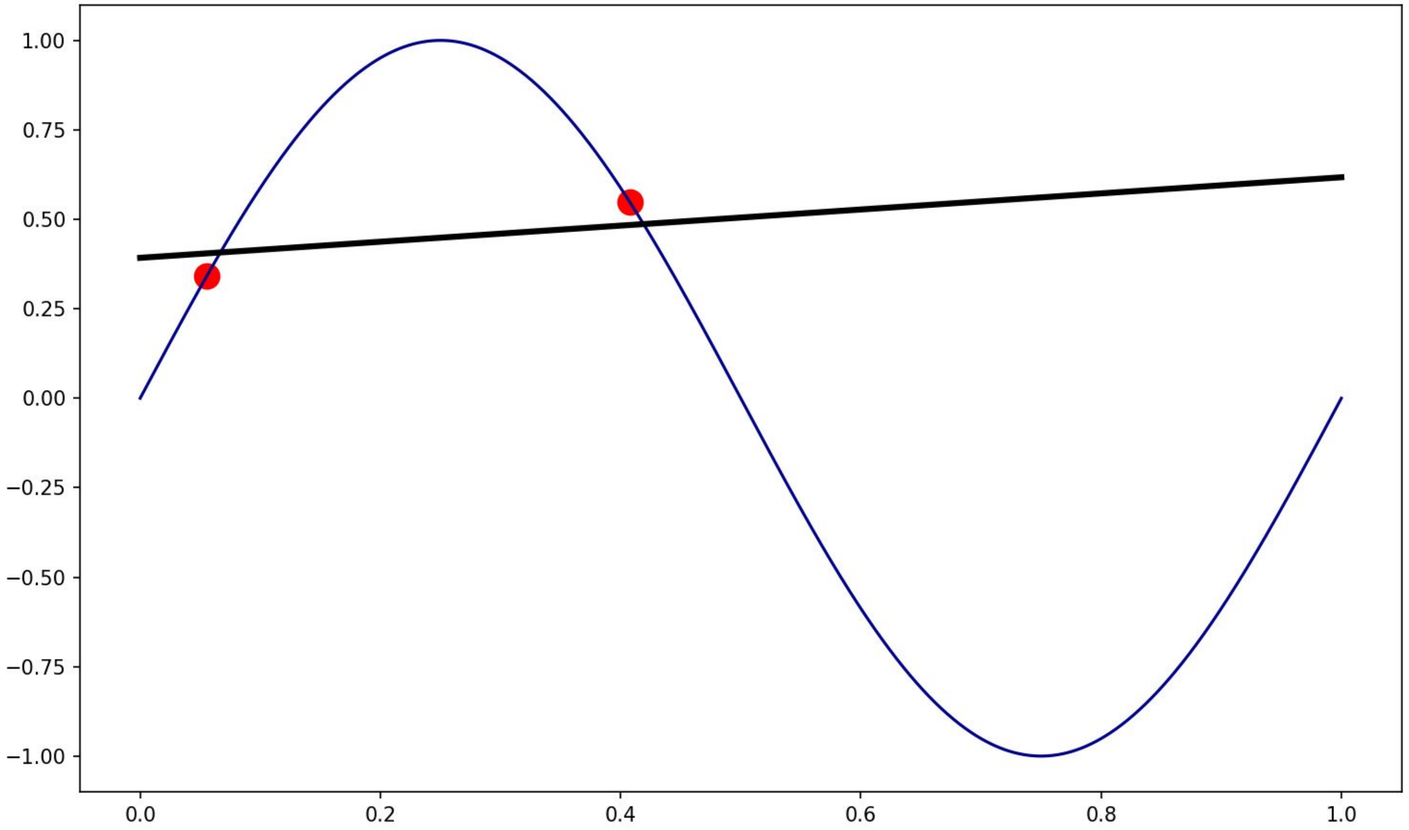
Ridge Regression with λ = 0.1

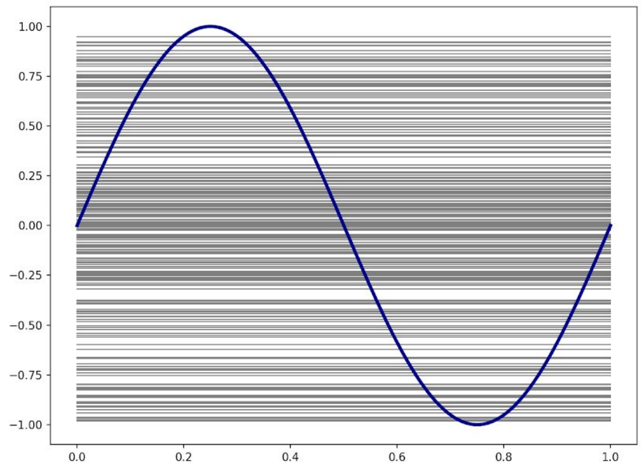


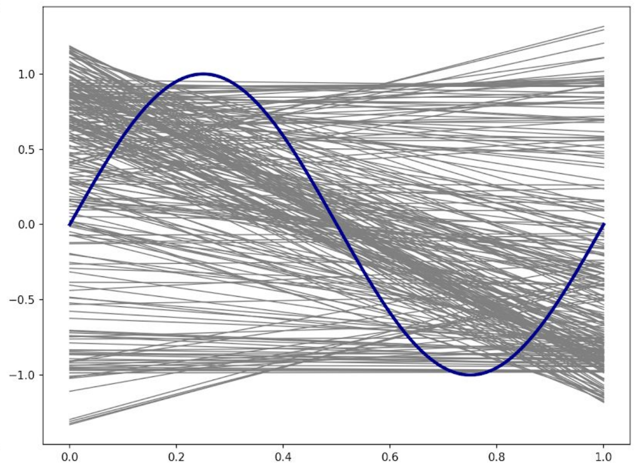
Ridge Regression with $\lambda = 0.1$

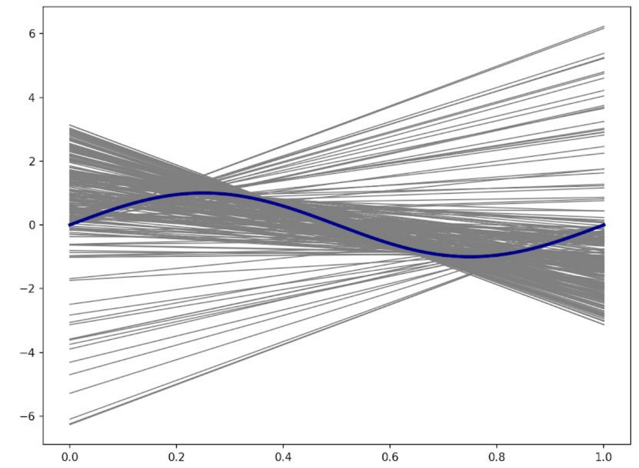


Ridge Regression with λ = 0.1









Constant

$\text{MSE}_{\text{generalization}} = 0.749$

$\text{Bias}^2 = 0.5$
Variance = 0.249

Ridge

$\text{MSE}_{\text{generalization}} = 0.654$

$\text{Bias}^2 = 0.327$
Variance = 0.327

Degree 1

$\text{MSE}_{\text{generalization}} = 1.920$

$\text{Bias}^2 = 0.209$
Variance = 1.711

Regularization

Regularization can also help when you have correlated predictors, because it can reduce the variability in the coefficients.

LASSO Regression

LASSO regression also fits a model of the same form:

$$\hat{f}(\vec{x}) = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \cdots + \beta_k x^{(k)}$$

But seeks to minimize a different objective function:

$$L = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2 + \lambda \sum_{i=1}^{k} |\beta_k|$$

LASSO Regression

LASSO regression is useful for regularization.

It also has the benefit that it can perform *feature selection*. Because math, LASSO regression has a tendency to set some of the coefficients equal to 0.

Ridge vs. LASSO

Ridge:

$$L = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2 + \lambda \sum_{i=1}^{k} \beta_k^2$$

LASSO:

$$L = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2 + \lambda \sum_{i=1}^{k} |\beta_k|$$

ElasticNet Regression

ElasticNet regression also fits a model of the same form:

$$\hat{f}(\vec{x}) = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \cdots + \beta_k x^{(k)}$$

But seeks to minimize a different objective function which blends ridge and LASSO regression:

$$L = \sum_{i=1}^{n} (y_i - \hat{f}(\vec{x}_i))^2 + \lambda_1 \sum_{i=1}^{k} |\beta_k| + \lambda_2 \sum_{i=1}^{k} \beta_k^2$$

Logistic Regression - Regularization

We can also do constrain the coefficients of a logistic regression model using Ridge or LASSO regression. However, we'll maximize the *log* likelihood instead of the likelihood to formulate how this is done.

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

$$\Updownarrow$$

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i)))$$

Logistic Regression - Regularization

We can also do constrain the coefficients of a logistic regression model using Ridge or LASSO regression. However, we'll maximize the *log* likelihood instead of the likelihood to formulate how this is done.

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i)))$$

Ridge

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i))) - \lambda \sum_{j=1}^{k} \beta_j^2$$

LASSO

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i))) - \lambda \sum_{j=1}^{k} |\beta_j|$$

Penalized Regression

Question: How do we find the value of the regularization hyperparameter?

We cannot just try several and choose the one that has be best performance on the test set, because then our estimate will be biased!

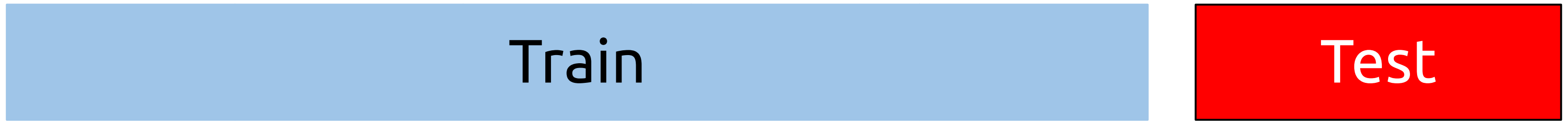
Instead, we can employ **cross-validation**.

k-Fold Cross-Validation

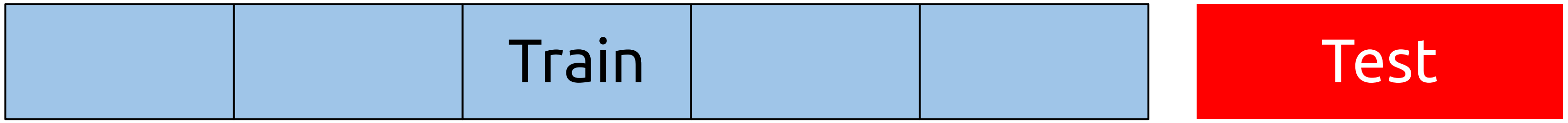
Full Dataset

First, split into a train
and a test set.

k-Fold Cross-Validation

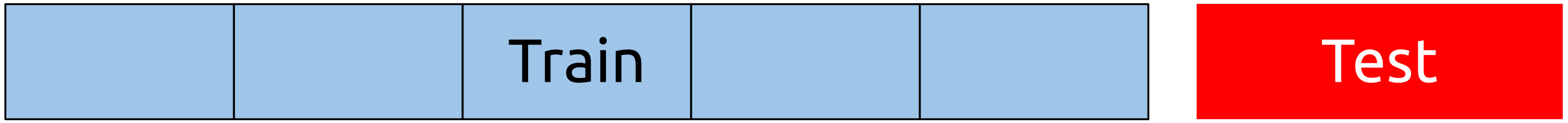


k-Fold Cross-Validation



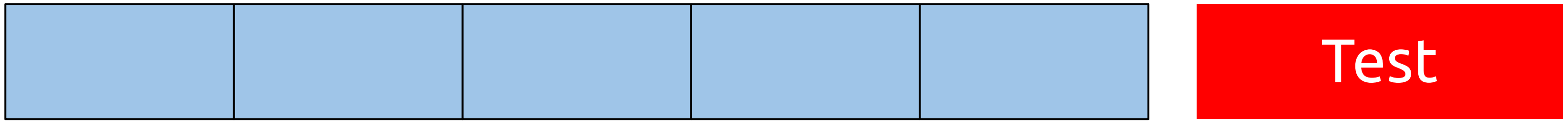
Then, divide the training data into k (here 5) equally-sized pieces.

k-Fold Cross-Validation



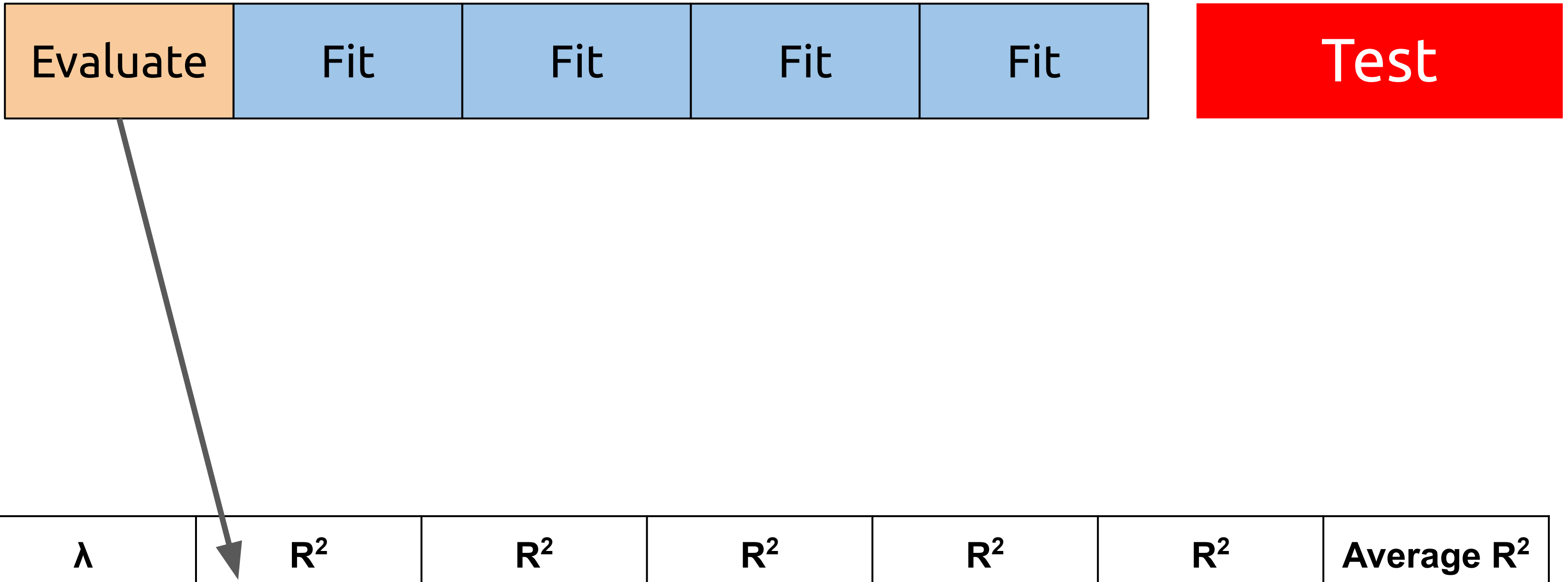
For each value of the hyperparameter(s), fit k different models, each on k-1 of the pieces of the training data and measure the performance on the remaining one piece.

k-Fold Cross-Validation



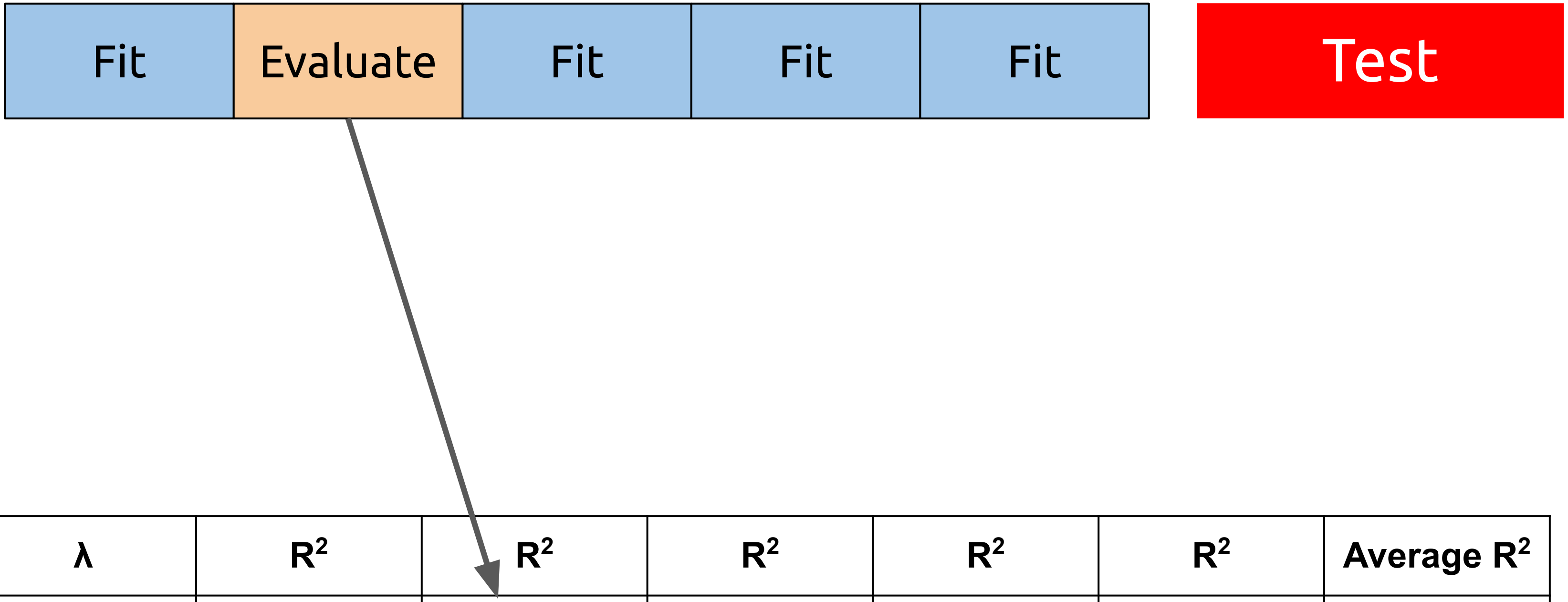
λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1						
0.5						

k-Fold Cross-Validation



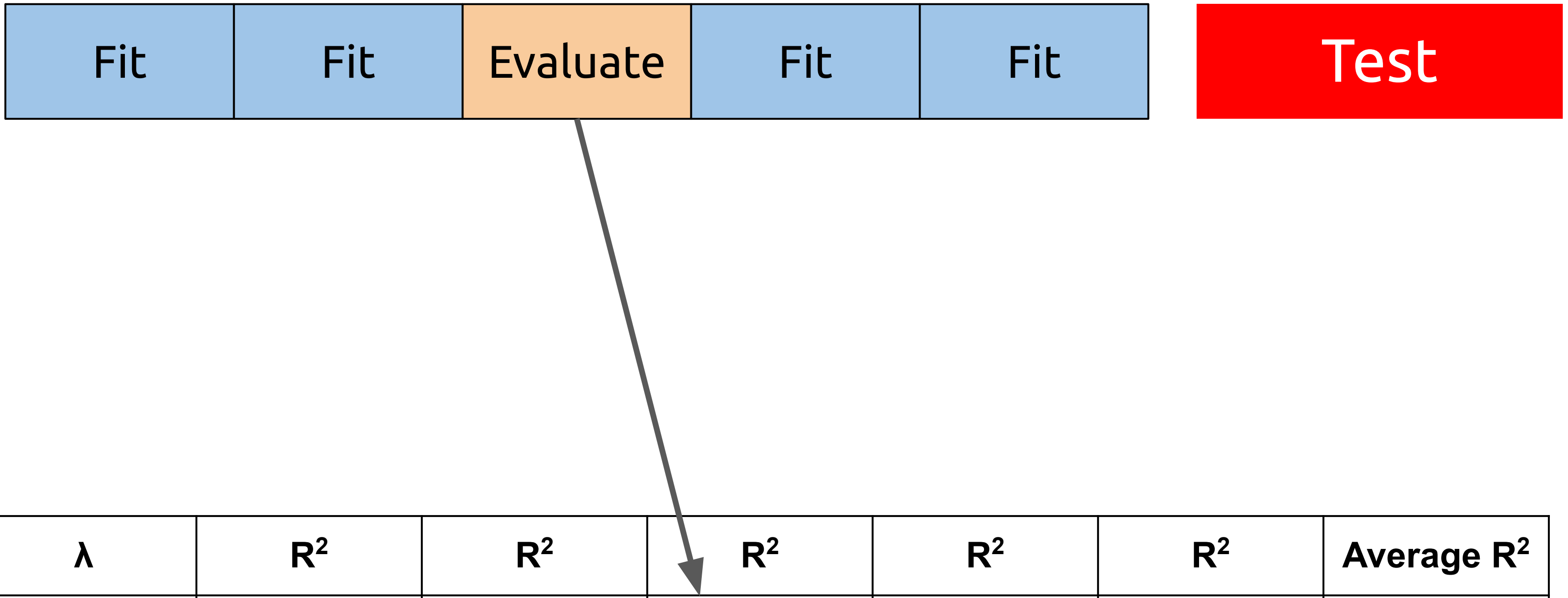
λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83					
0.5						

k-Fold Cross-Validation



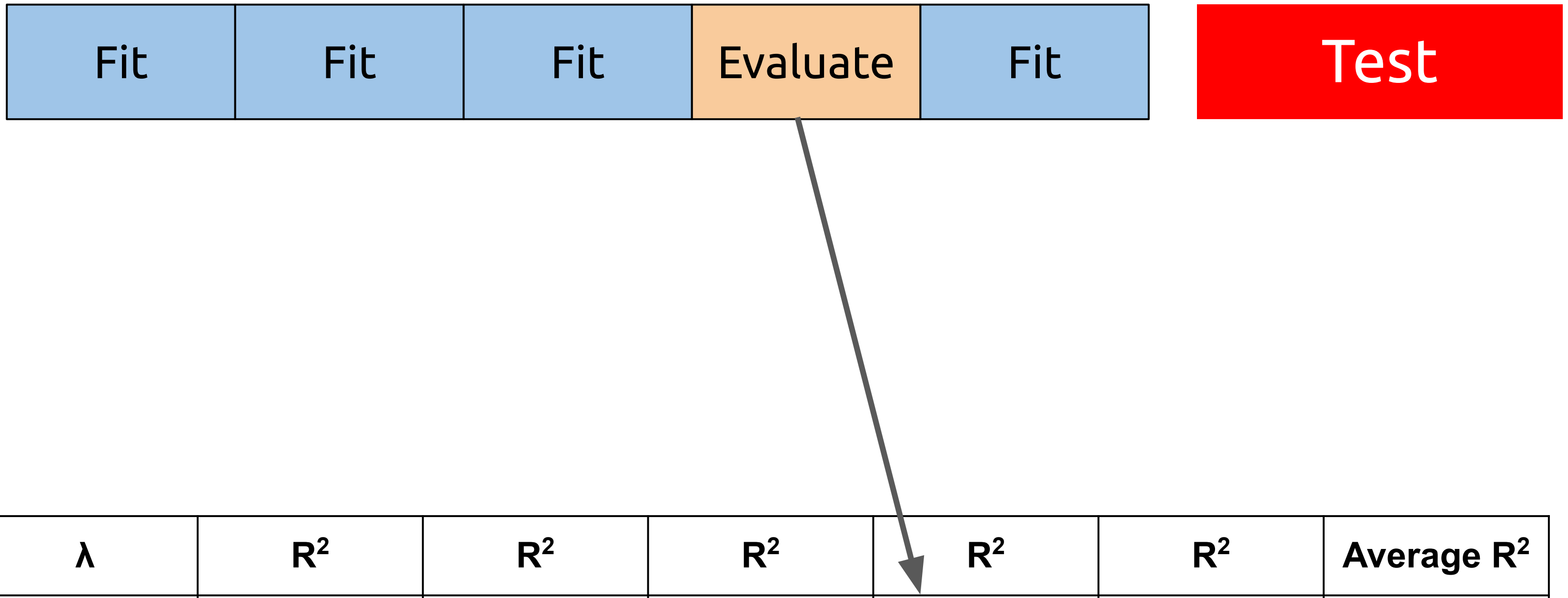
λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83	0.75				
0.5						

k-Fold Cross-Validation



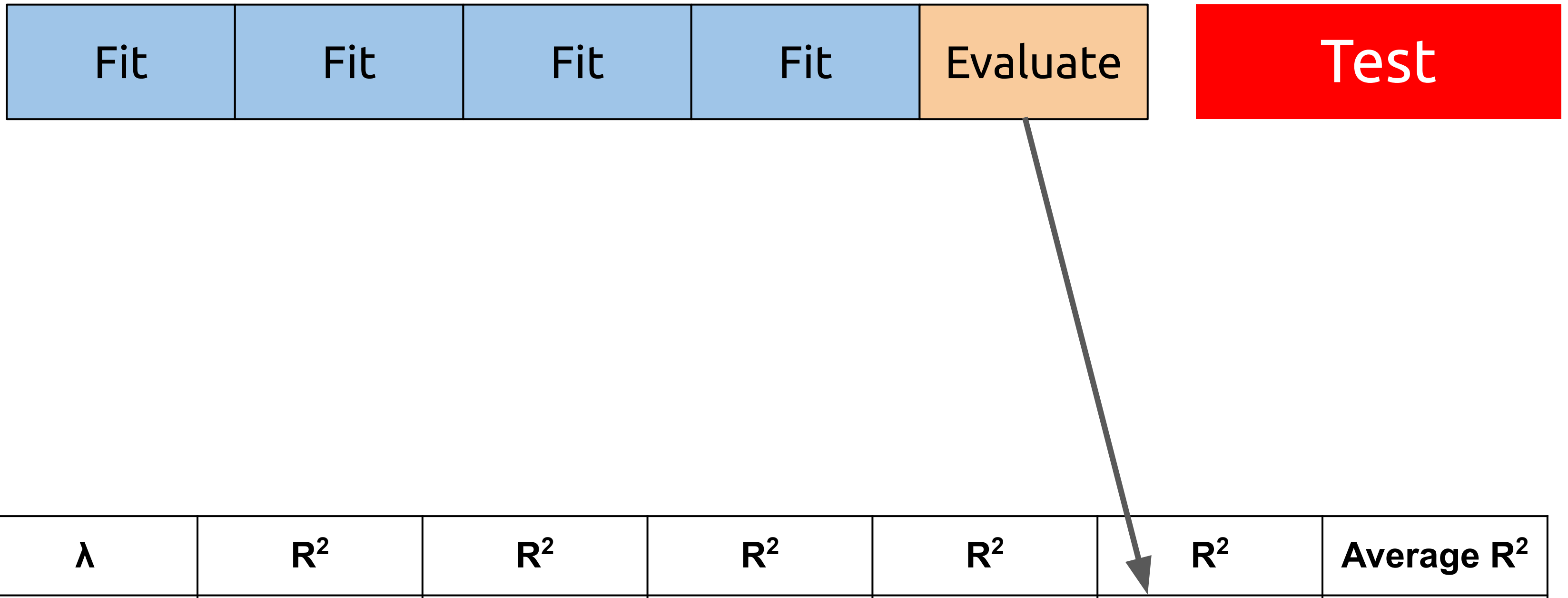
λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83	0.75	0.81			
0.5						

k-Fold Cross-Validation



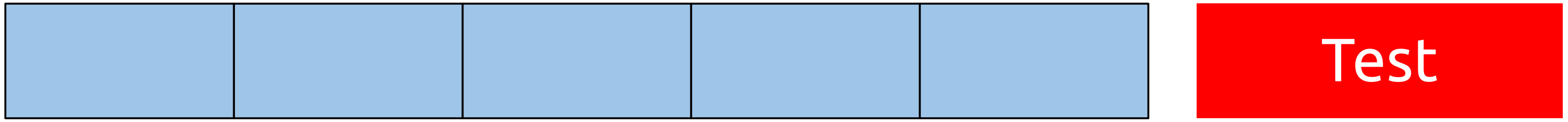
λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83	0.75	0.81	0.82		
0.5						

k-Fold Cross-Validation



λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83	0.75	0.81	0.82	0.85	
0.5						

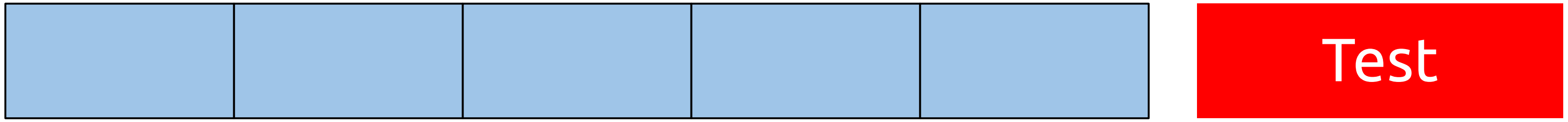
k-Fold Cross-Validation



Then average the results together.

λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83	0.75	0.81	0.82	0.85	0.812
0.5						

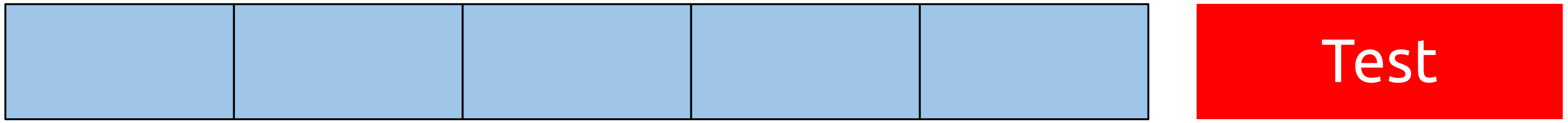
k-Fold Cross-Validation



Repeat for other values of your hyperparameter(s).

λ	R^2	R^2	R^2	R^2	R^2	Average R^2
0.1	0.83	0.75	0.81	0.82	0.85	0.812
0.5	0.73	0.80	0.77	0.88	0.82	0.800

k-Fold Cross-Validation



Choose the option with the best average performance.

To get a final estimate of performance, evaluate on the held-out test data.

λ	**R^2**	**R^2**	**R^2**	**R^2**	**R^2**	**Average R^2**
0.1	0.83	0.75	0.81	0.82	0.85	0.812
0.5	0.73	0.80	0.77	0.88	0.82	0.800