Model creation and evaluation

General supervised learning goal - function approximation

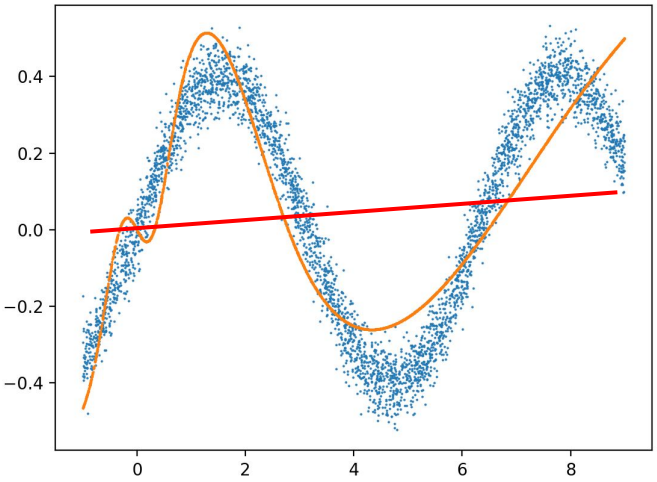
$$Y = f(X) + \epsilon$$ error

f(x) represents the *systematic* information that X provides about Y

f that connects the input variable to the output variable

f(x) is unknown and usually quite complicated

We need to estimate unknown *f(x)* based on observed points



$$Y \approx \hat{f}(x)$$

With hat because it's an estimate based on data

Two general f(x) estimation approaches:

1) Parametric
2) Non-parametric

Parametric

1. We **make an assumption about the functional form, or shape**, of *f(x)* . E.g. assume linear

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p.$$

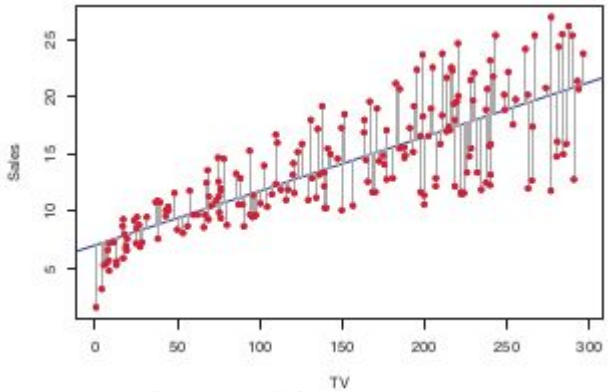
Problem of estimating f is simplified. Instead of having to estimate an entirely arbitrary p-dimensional function f (X), one only needs to estimate the p + 1 coefficients $\beta_0, \beta_1, \ldots, \beta_p$.

2. After a model has been selected, we need a procedure that uses the training data to *fit* or *train* the model. That is, we want to **find values of these parameters** $\beta_0, \beta_1, \ldots, \beta_p$ such that

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p.$$

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p.$$

Common approach to find those parameters is ordinary least squares (OLS).
It does so by **minimizing** the sum of squared errors from the data.



$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

Represents the prediction for Y based on the ith value of X

$$e_i = y_i - \hat{y}_i$$

Represents the ith residual— the difference between the ith observed and predicted value

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$

residual sum of squares (RSS)

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p.$$

OLS is not the only way -- we can add some additional restrictions and this will result in a different set of parameters

Parametric methods to estimate *f(x)*

Advantages

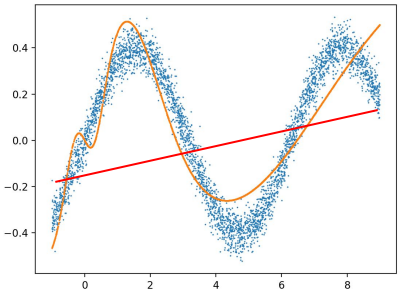
+ Simplifies the problem of estimating f !

It is generally much easier to estimate a set of parameters than it is to fit an entirely arbitrary function *f* .

Disadvantages

- the model we choose will usually not match the true unknown form of *f* .

If the chosen model is too far from the true f , then our estimate will be poor.



E.g. Here data is simulated using function

$$Y = 0.4 * sin(x) + \epsilon$$

Non-parametric methods

- **Do not make explicit assumptions about the functional form of f** .
- Instead they seek an estimate of *f* that gets as close to the data points as possible without being too rough or wiggly.

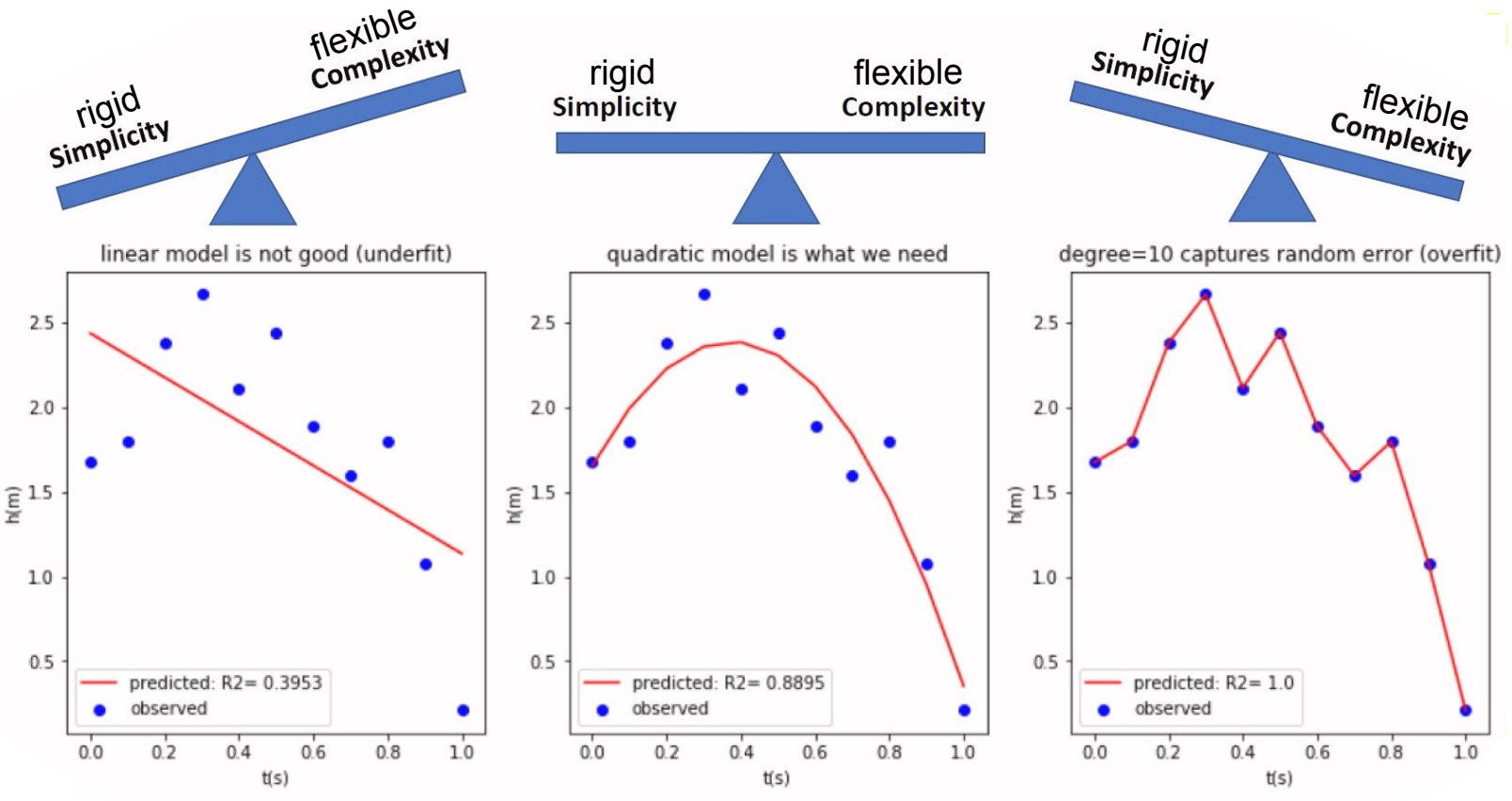
Advantages

+ Wider range of possible *f* shapes
+ Should be close to the true function

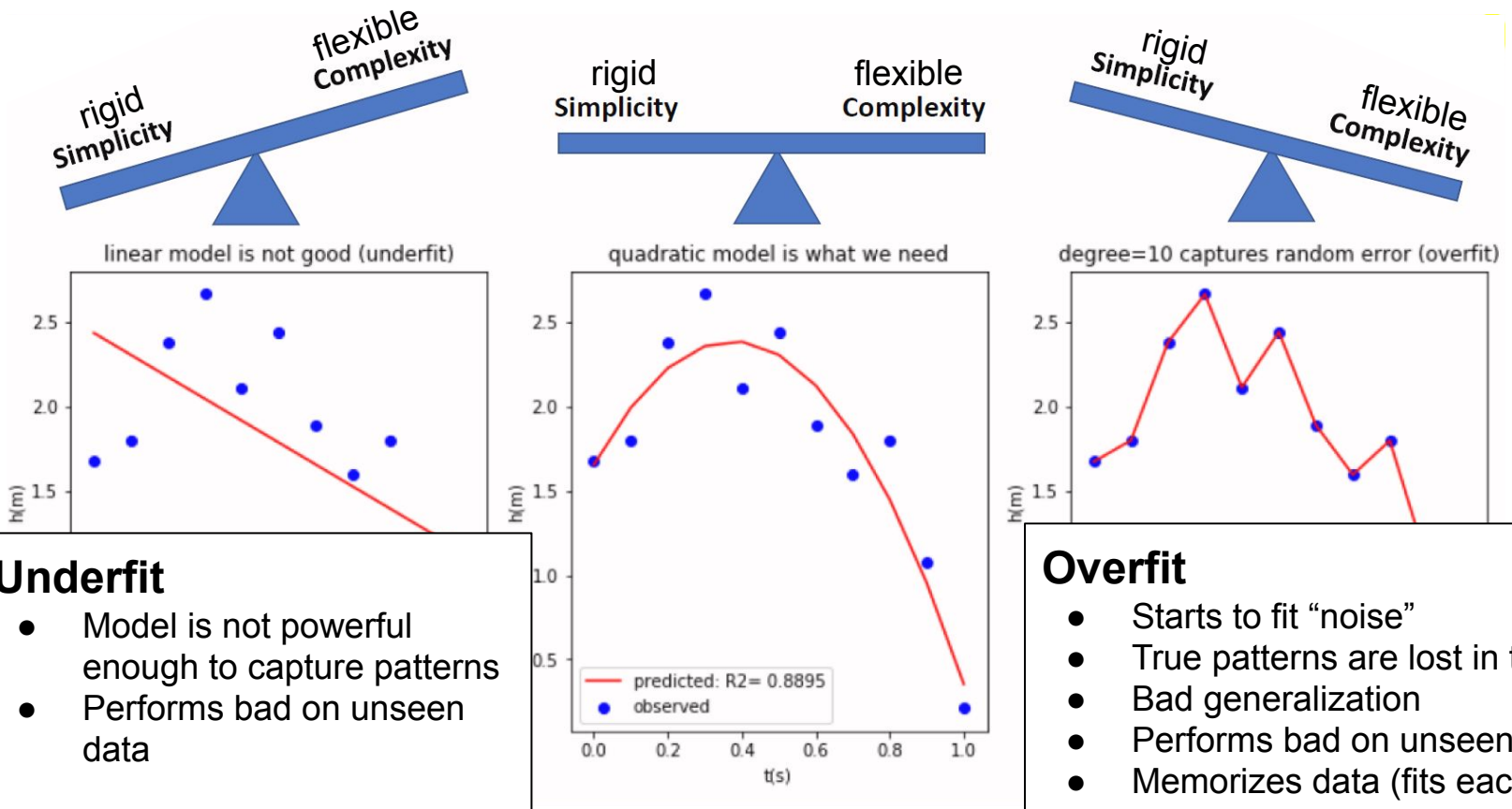
Disadvantages

- Needs more data than parametric approach
- Easier to overfit

Underfitting - Overfitting



Underfitting - Overfitting



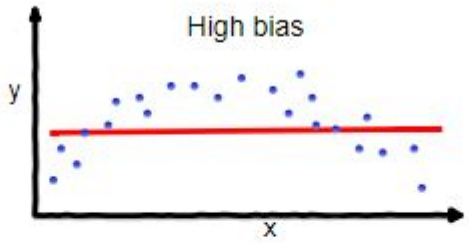
Underfit

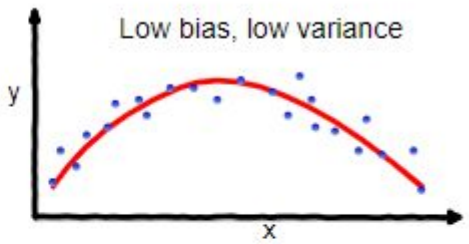
- Model is not powerful enough to capture patterns
- Performs bad on unseen data

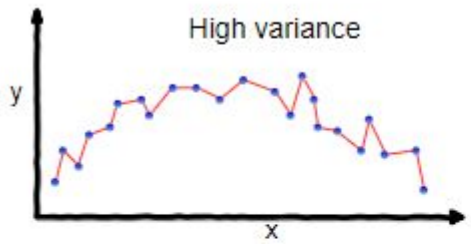
Overfit

- Starts to fit “noise”
- True patterns are lost in the noise
- Bad generalization
- Performs bad on unseen data
- Memorizes data (fits each point if too little data)

Bias-variance vs underfitting-overfitting





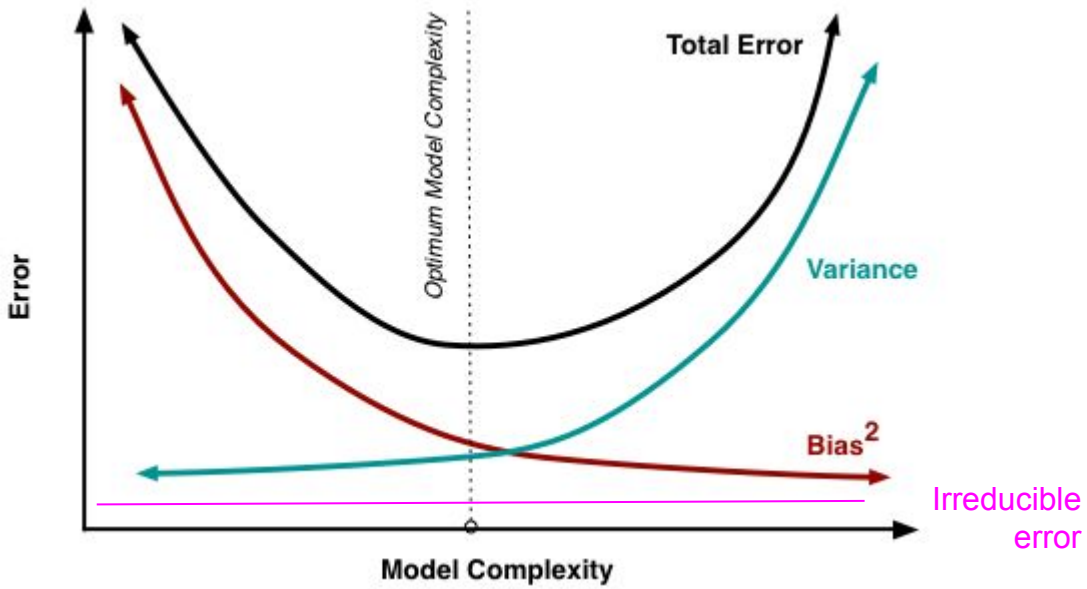


underfitting | **Good balance** | **overfitting**

- Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set
- Bias refers to the error that is introduced by approximating a real-life problem (which may be extremely complicated) by a much simpler model

Bias & variance

- Are terms to differentiate the reason for your model total error
- Low model complexity -- high total error due to bias
- High model complexity -- high total error due to variance

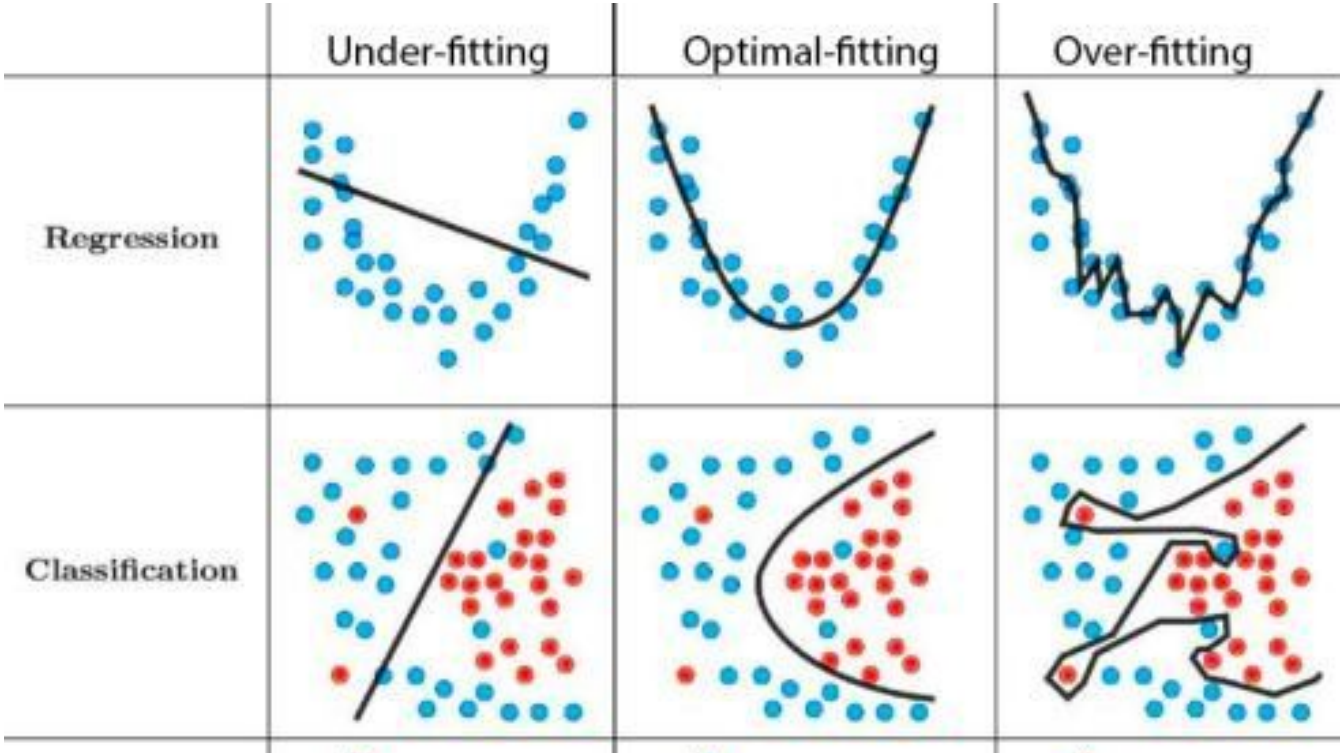


Total Model Error = Variance + Bias^2 + irreducible error

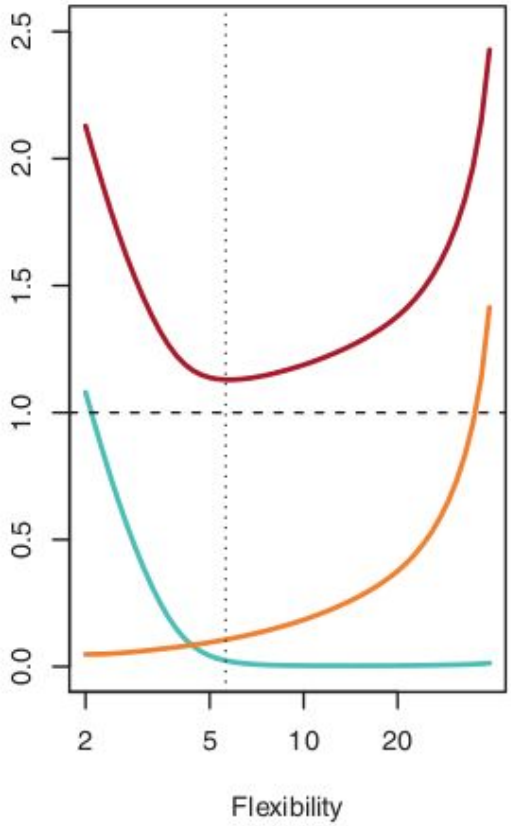
Derived from linear model

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

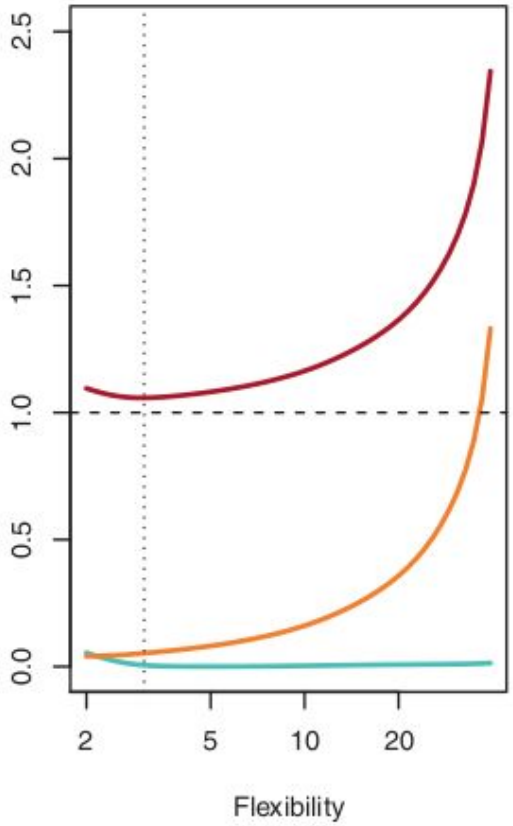
Relevant for regression and classification



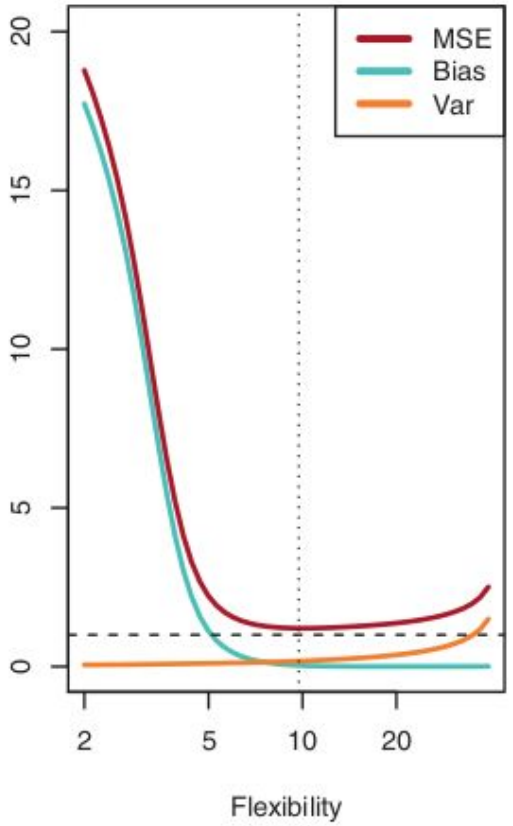
Optimal model complexity is specific for each dataset



"True" *f* is non-linear



"True" *f* is linear

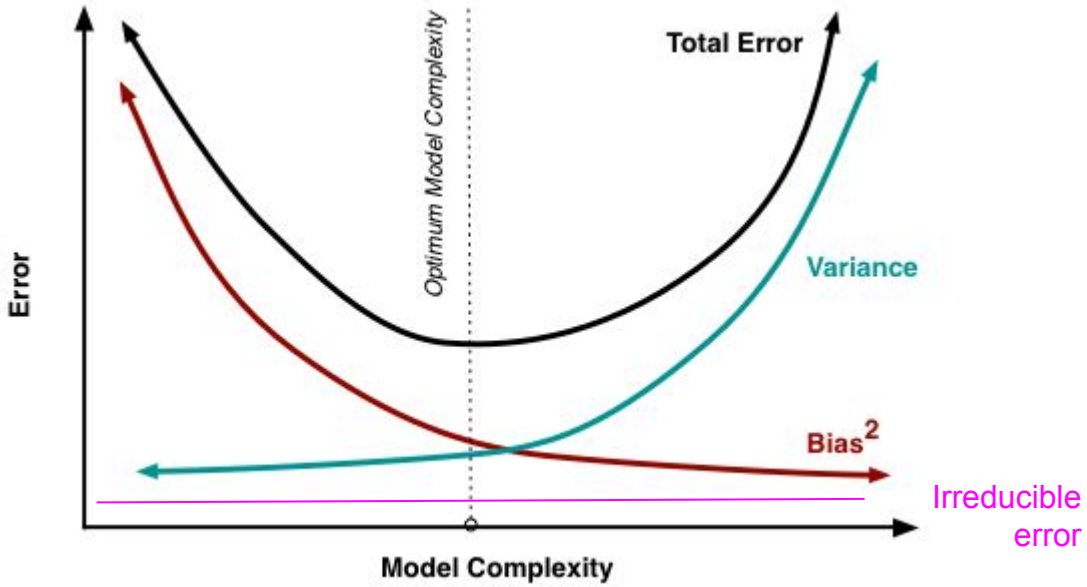


"True" *f* is very non-linear

Optimum complexity -- what can we do?

Change model complexity with:

- Adjusting model hyperparameters (e.g. number of predictors used, hyperparameter value)
- Selecting different algorithm/model



Total Model Error = Variance + Bias^2 + irreducible error

Derived from linear model

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

Some methods are inherently more flexible than others

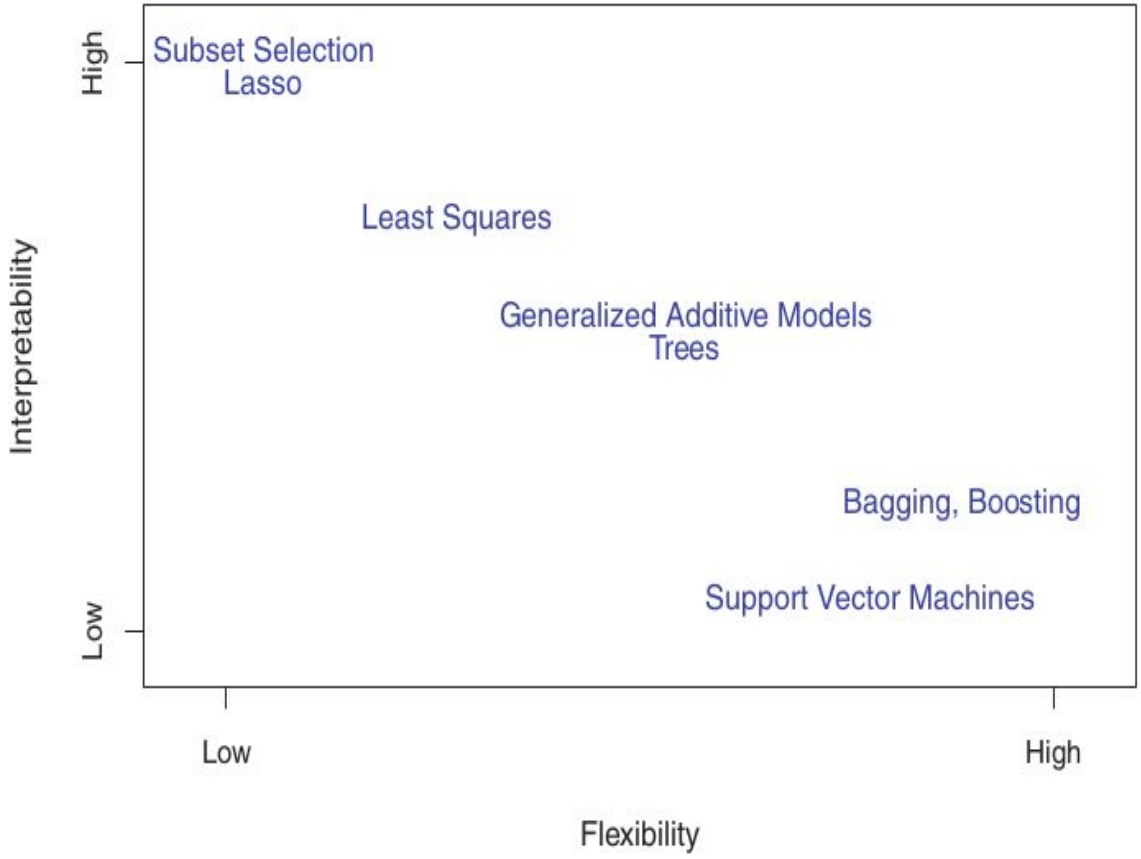
If

- True function *f* is simple -- both simple and flexible methods can perform well
- True function *f* is complex -- flexible method will perform better than simple,

so

- *why would we ever choose to use a more restrictive method instead of a very flexible approach?*

Flexibility is also related to how easy is to interpret the model



More flexible models are generally more like “black boxes”:

- Good results
- Hard to define how predictors are related to target feature y

Model goal:
Inference vs prediction ?

Model goal:
Inference vs prediction ?

Caution !

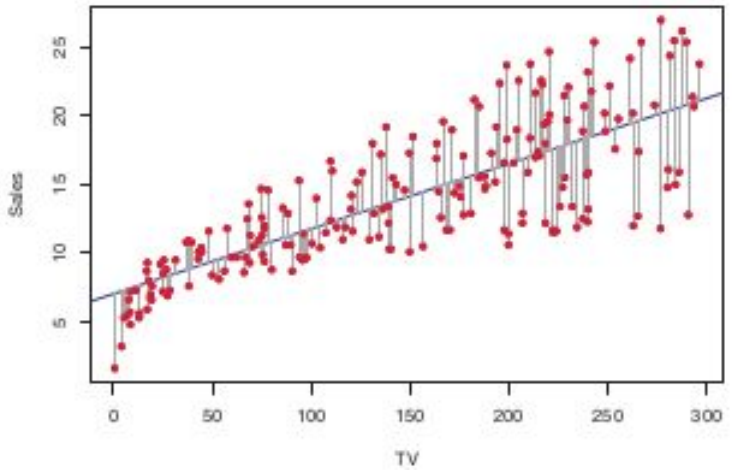
It is very easy to overfit

How to figure out if we overfitted?

Common model assessment statistics for regression is Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Where n is the number of data points, y_i is the actual value for data point i and hat $f(x_i)$ -- value returned by the model



$$testMSE = \frac{1}{n}\sum_{i=1}^{n}(y_0 - \hat{f}(x_0))^2$$

Where y_0 and x_0 belong to **unseen data**

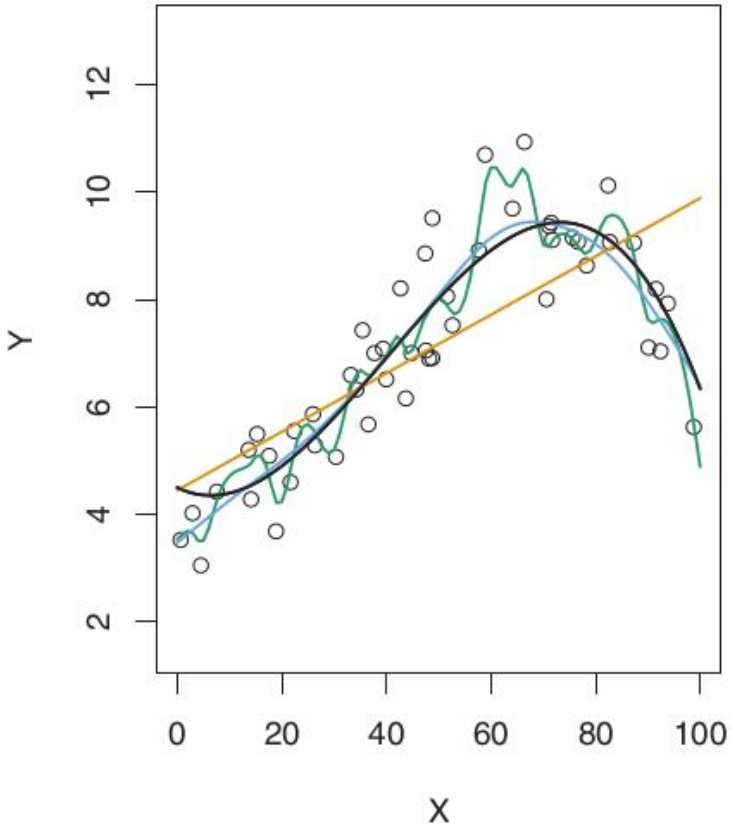
How to figure out if we overfitted?

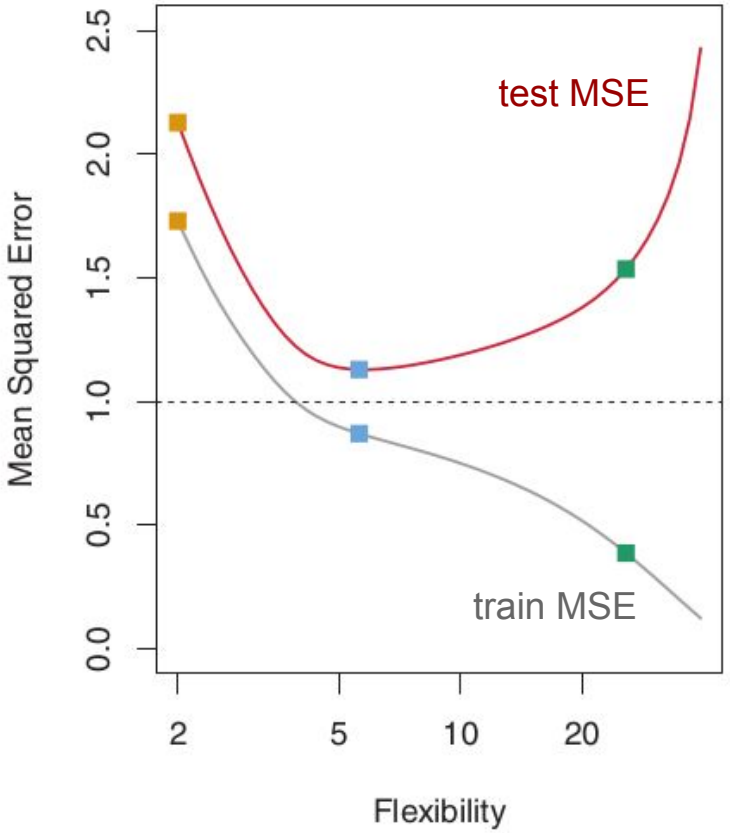
If we use flexible enough model -- we can fit every point. Then all residuals are

$$e_i = y_i - \hat{y}_i = 0$$

and

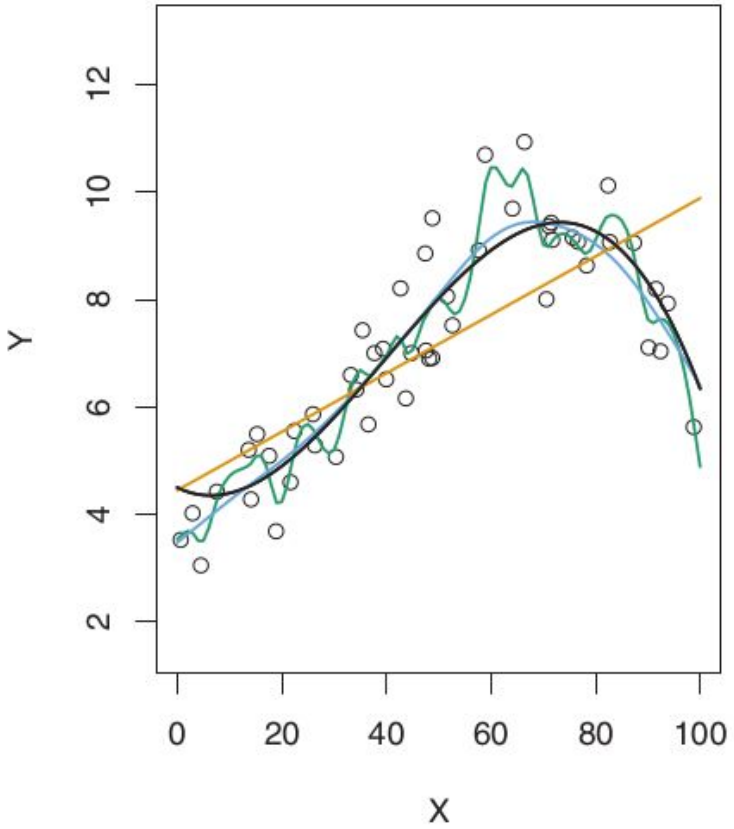
$$trainMSE = 0$$

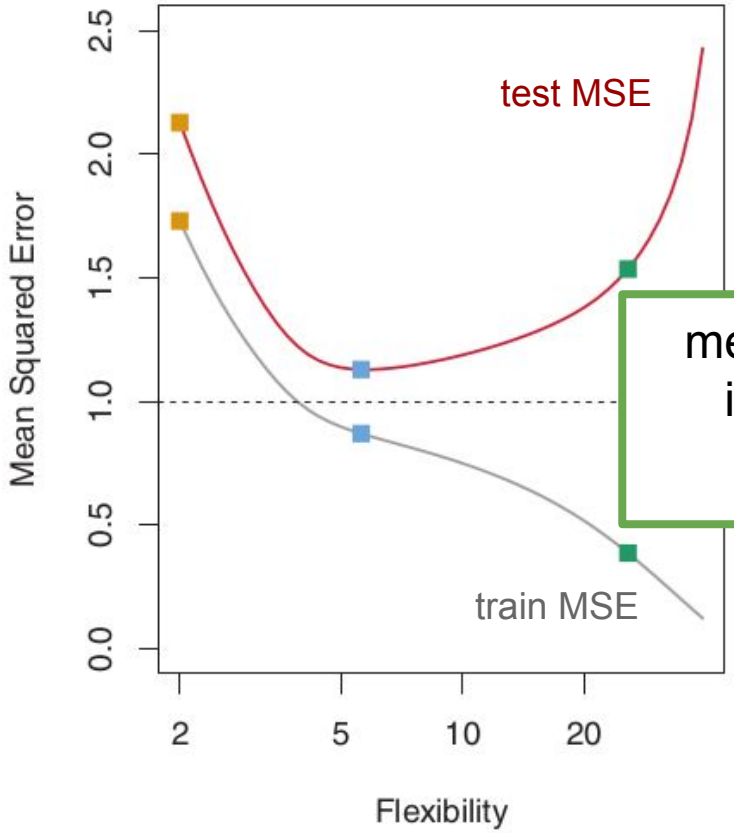




Black line -- “True” function used to simulated data

Yellow, blue, green -- models with different flexibility (linear regression, smoothing spline fit)



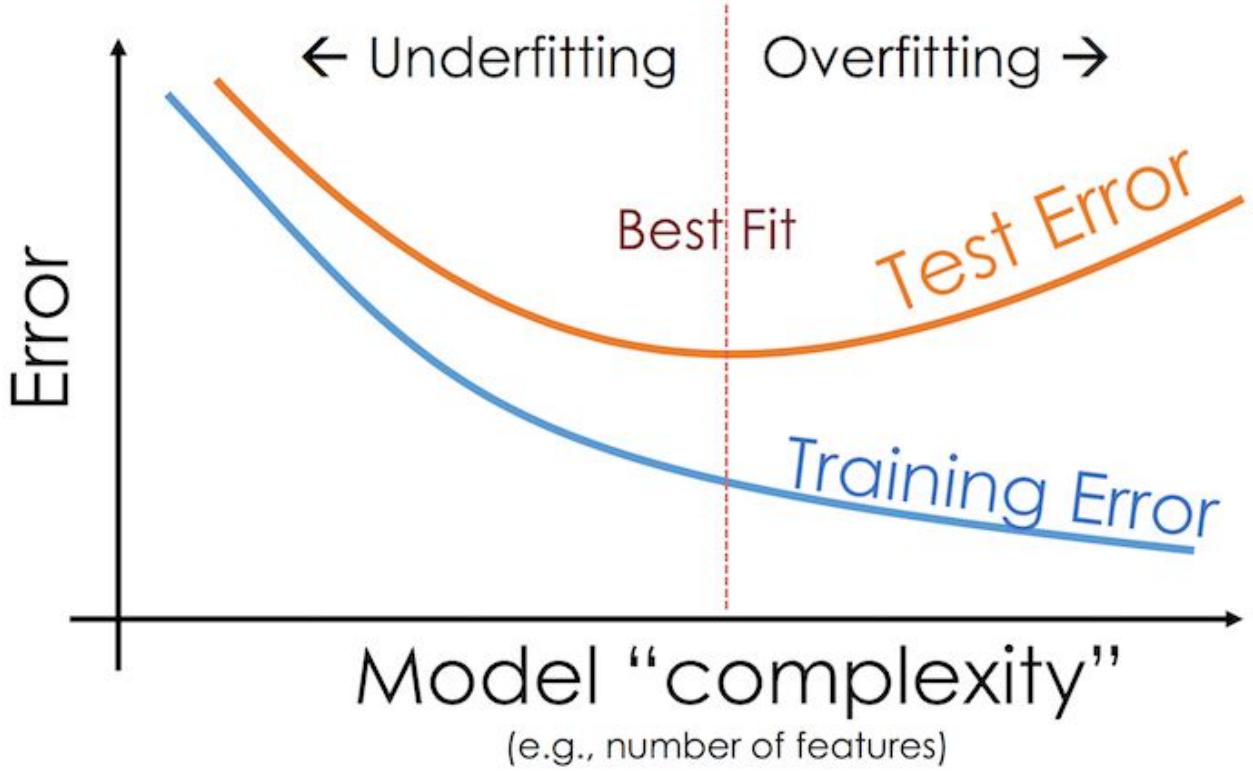


memorization instead of learning!

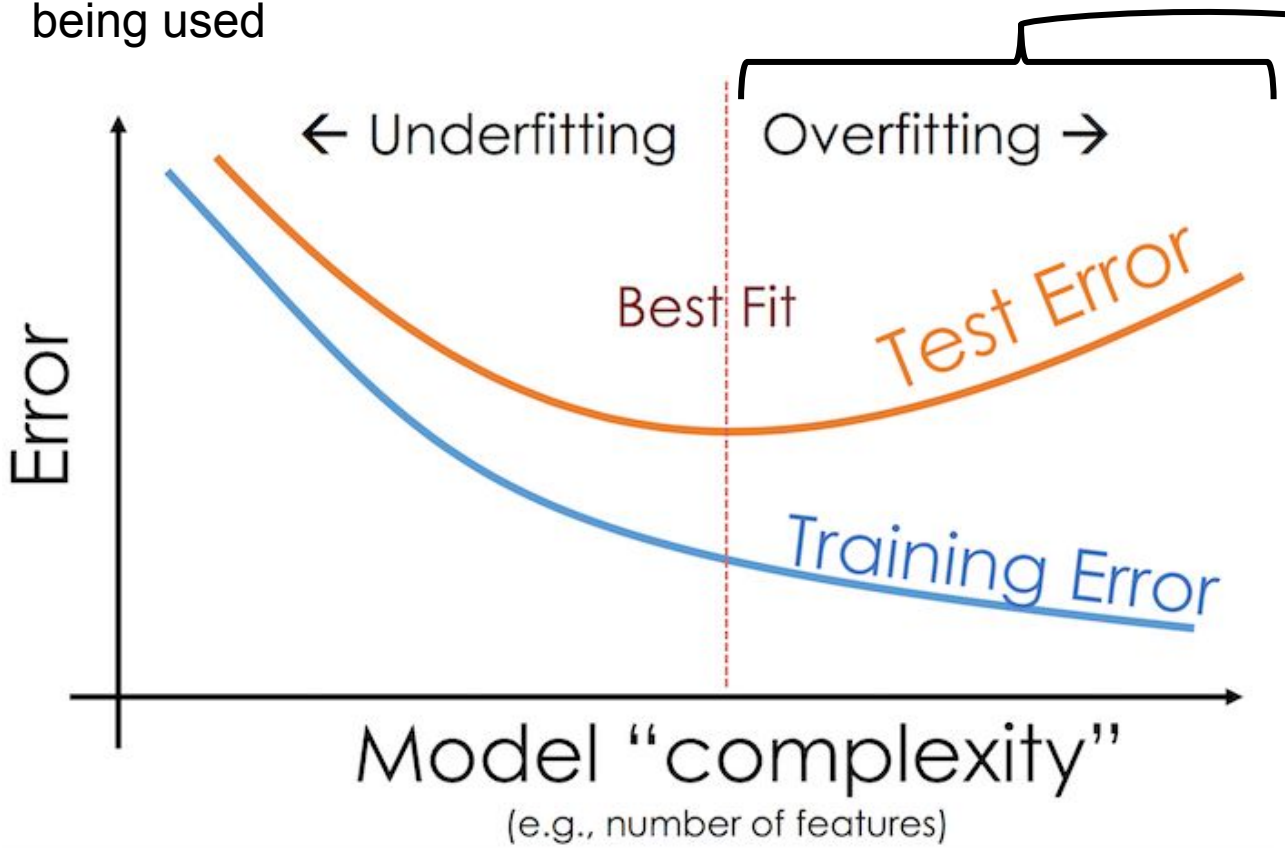
Black line -- “True” function used to simulated data

Yellow, blue, green -- models with different flexibility (linear regression, smoothing spline fit)

U-shaped test MSE is a **fundamental property** of statistical learning that holds regardless of the particular data set at hand and regardless of the statistical method being used



U-shaped test MSE is a fundamental property of statistical learning that holds regardless of the particular data set at hand and regardless of the statistical method being used

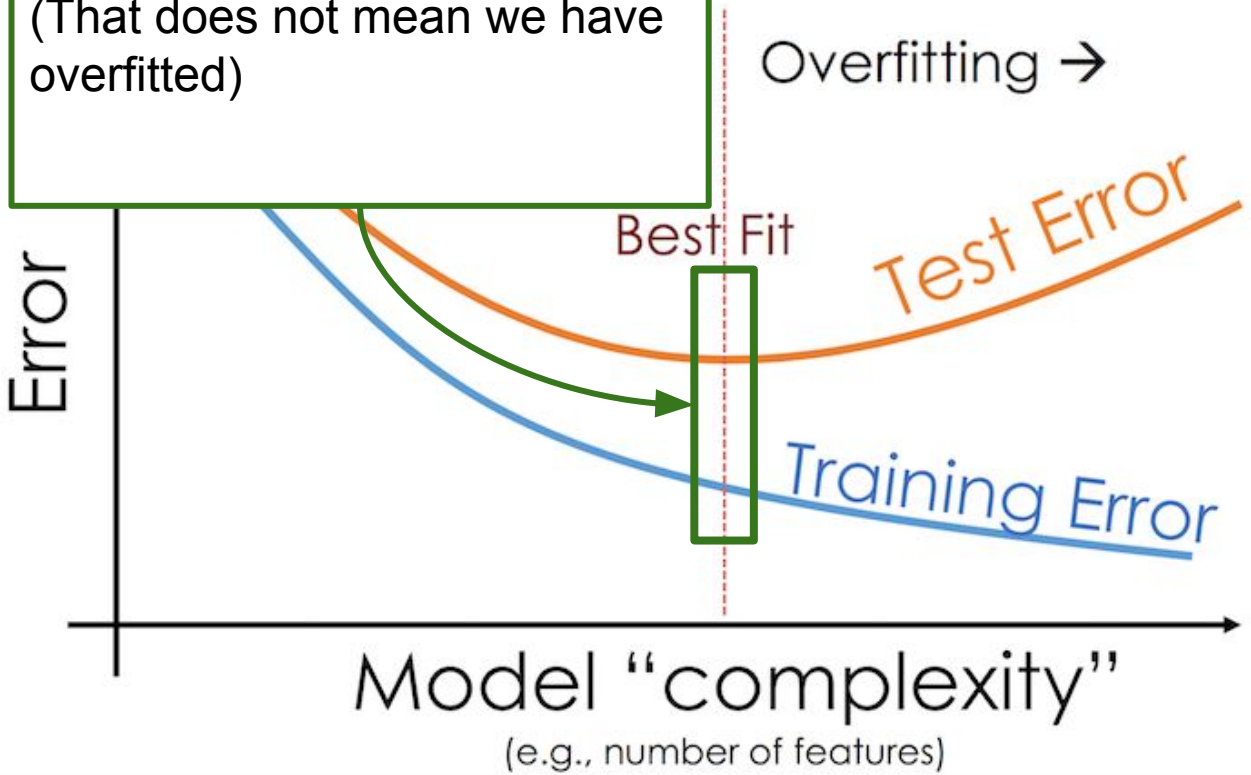


Our method is:

- working too hard to find patterns in the training data
- may be picking up some patterns that are just caused by random chance

ental property of statistical learning that holds
set at hand and regardless of the statistical method

NOTE:
Test error is always bigger then training error.
(That does not mean we have overfitted)

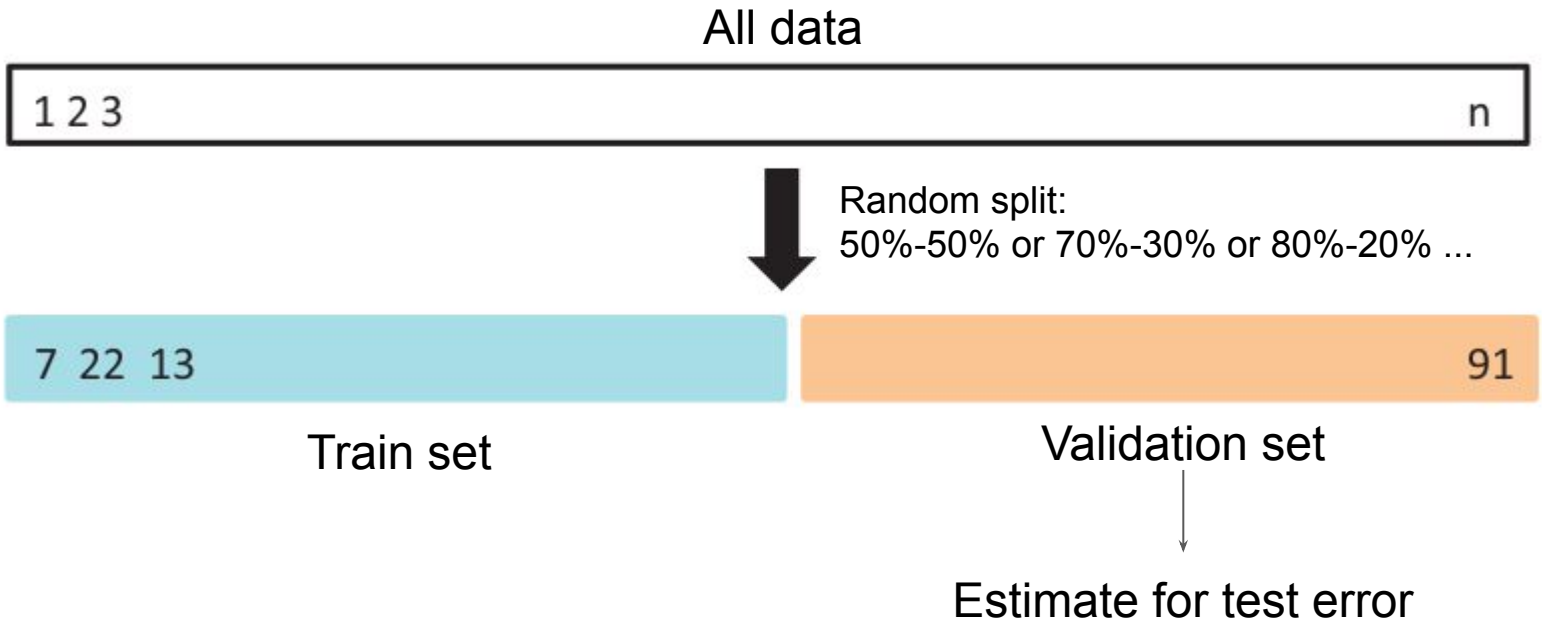


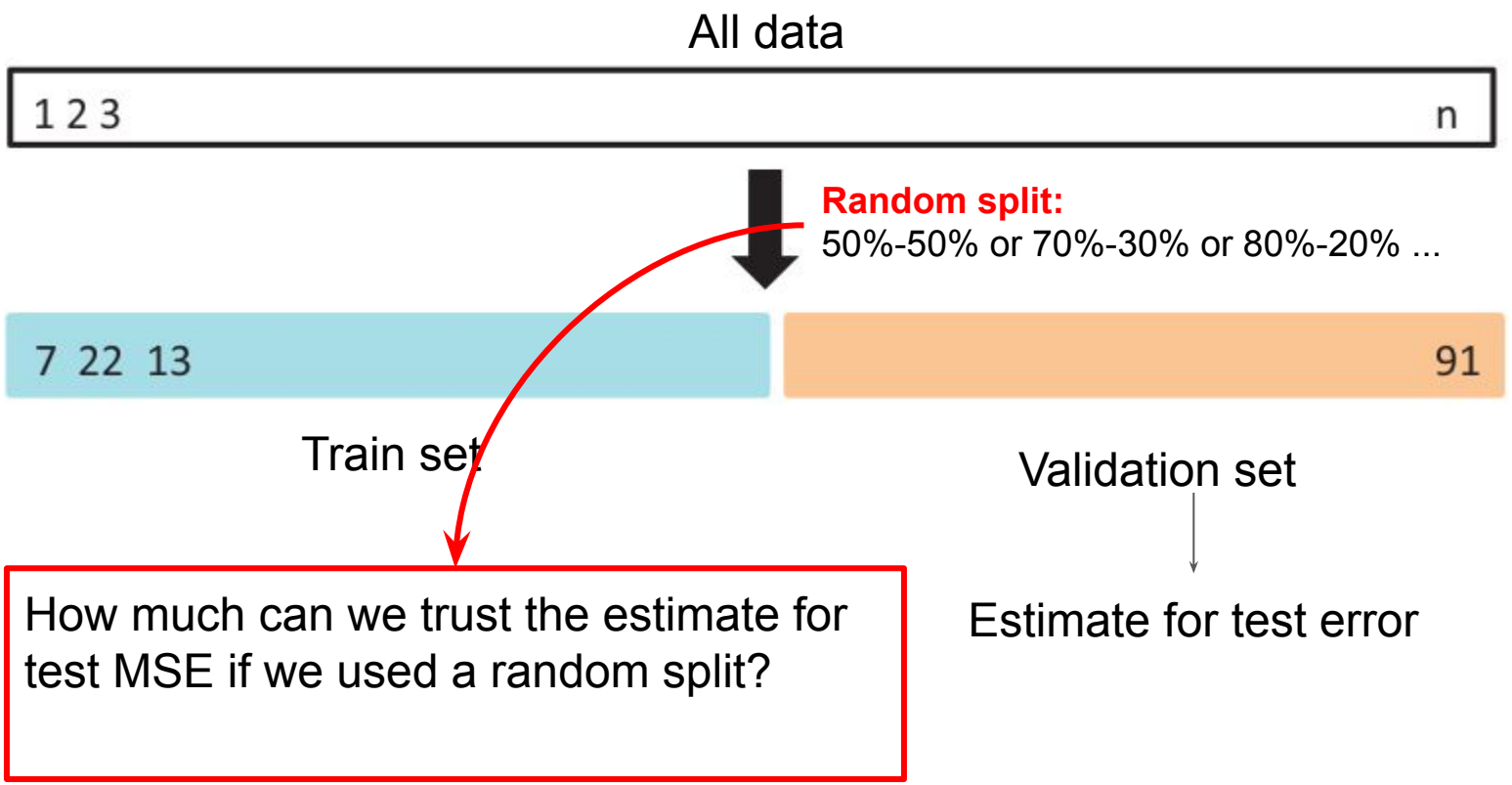
So we need unseen test data...

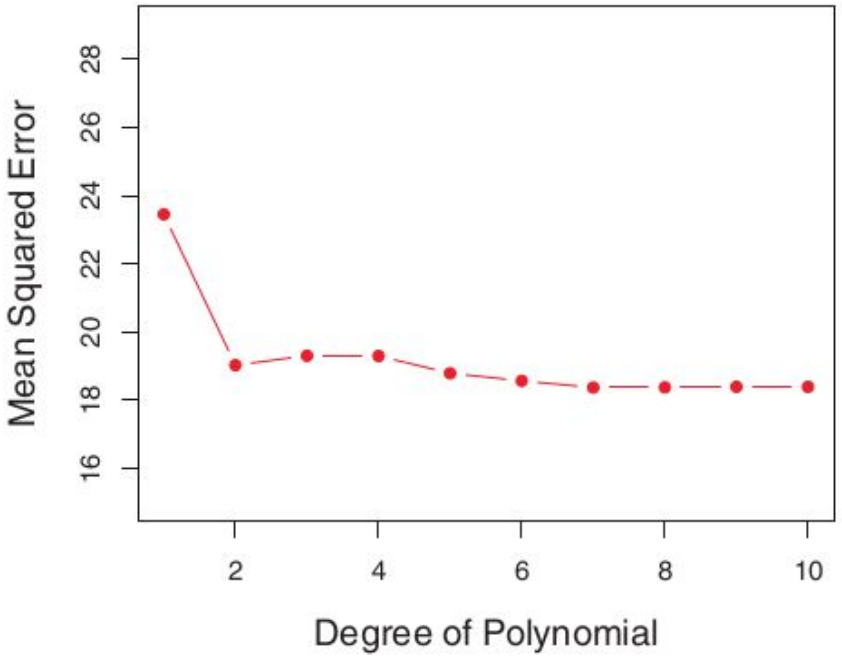
- Amount of available data is limited
- Have to make do with what we have...

So we need unseen test data...

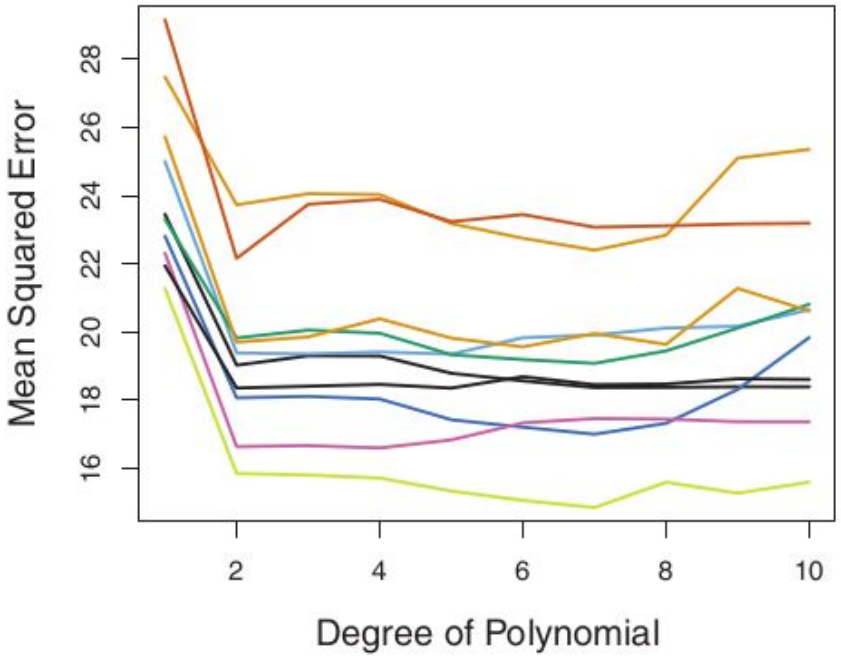
- Amount of available data is limited
- Have to make do with what we have...



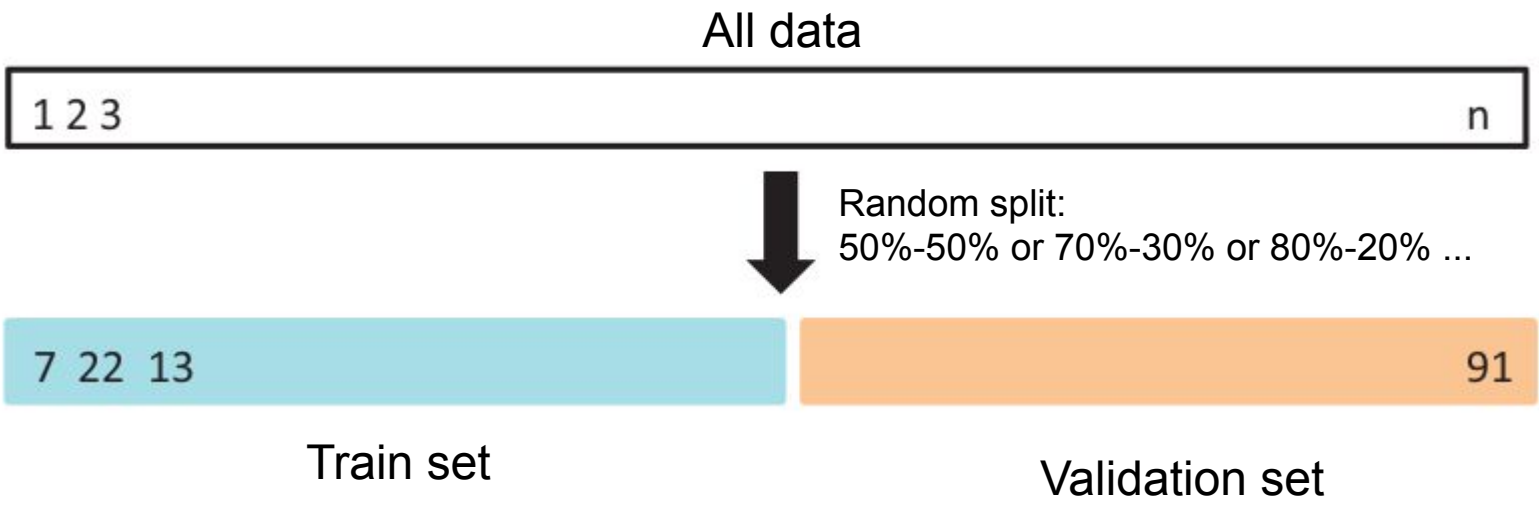

All data
1 2 3
n
Random split:
50%-50% or 70%-30% or 80%-20% ...
7 22 13
91
Train set
Validation set
How much can we trust the estimate for test MSE if we used a random split?
Estimate for test error



1 random data split



10 random data splits

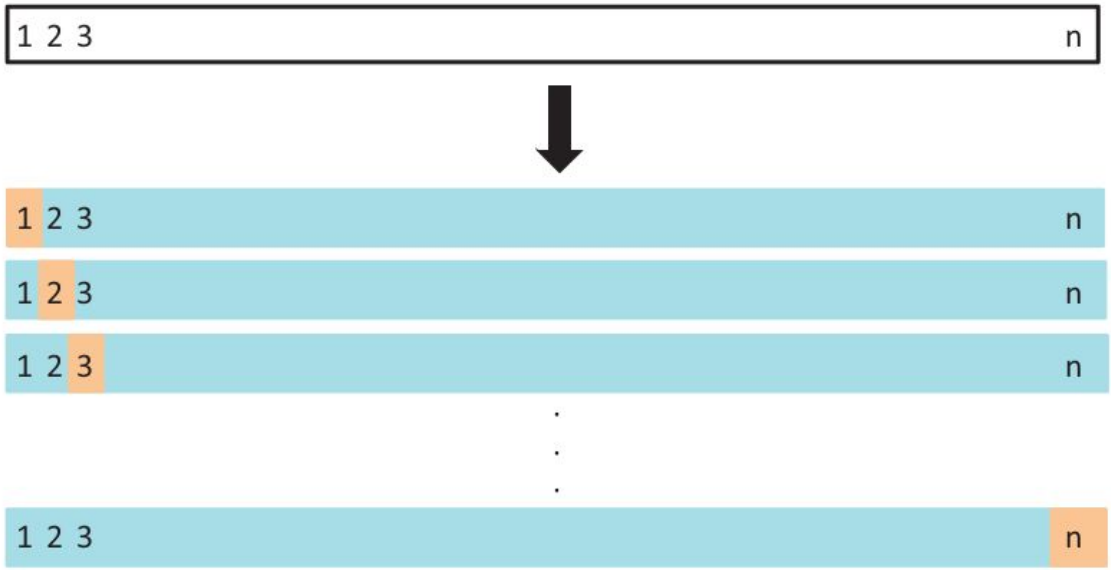


Drawbacks:

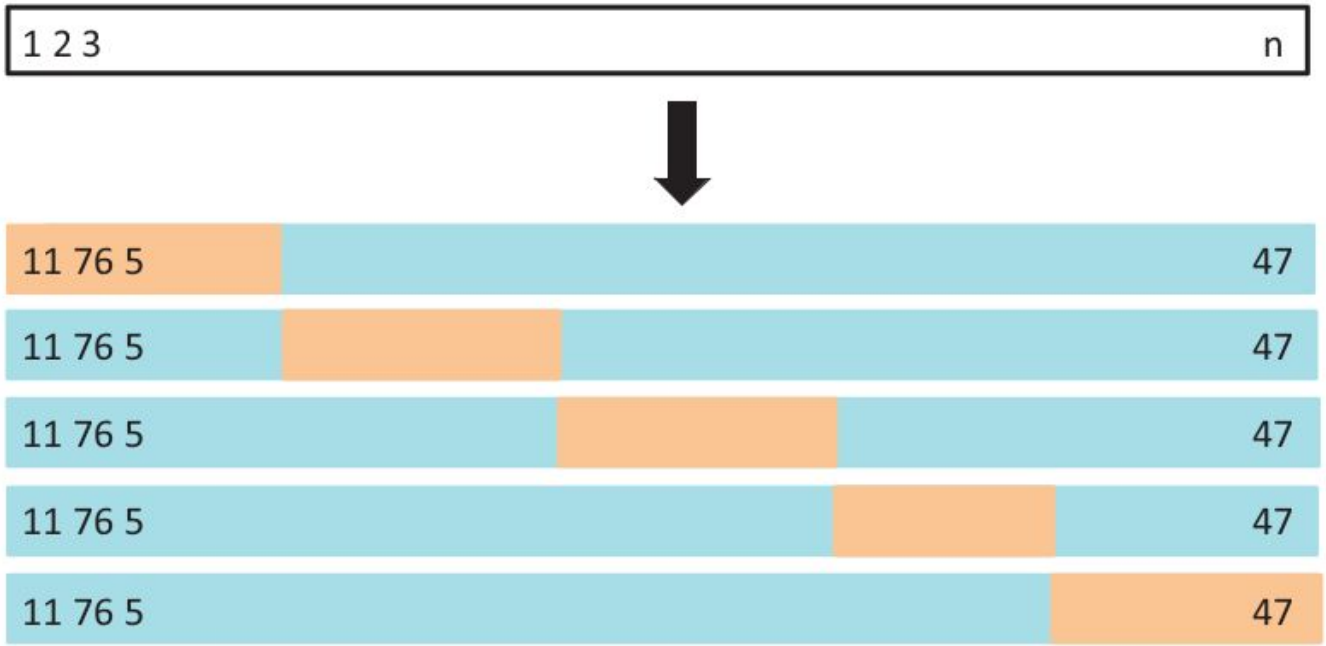
1) validation estimate of the test error rate can be highly variable
2) only a subset of the observations - are used to fit the model.
   statistical methods tend to perform worse when trained on fewer observations →

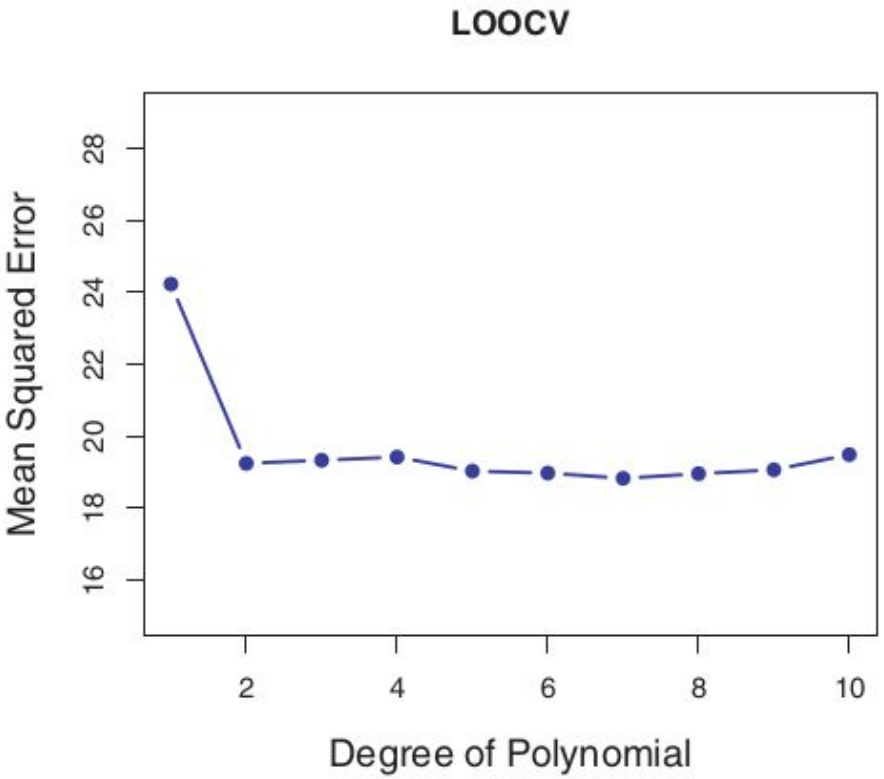
   validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set

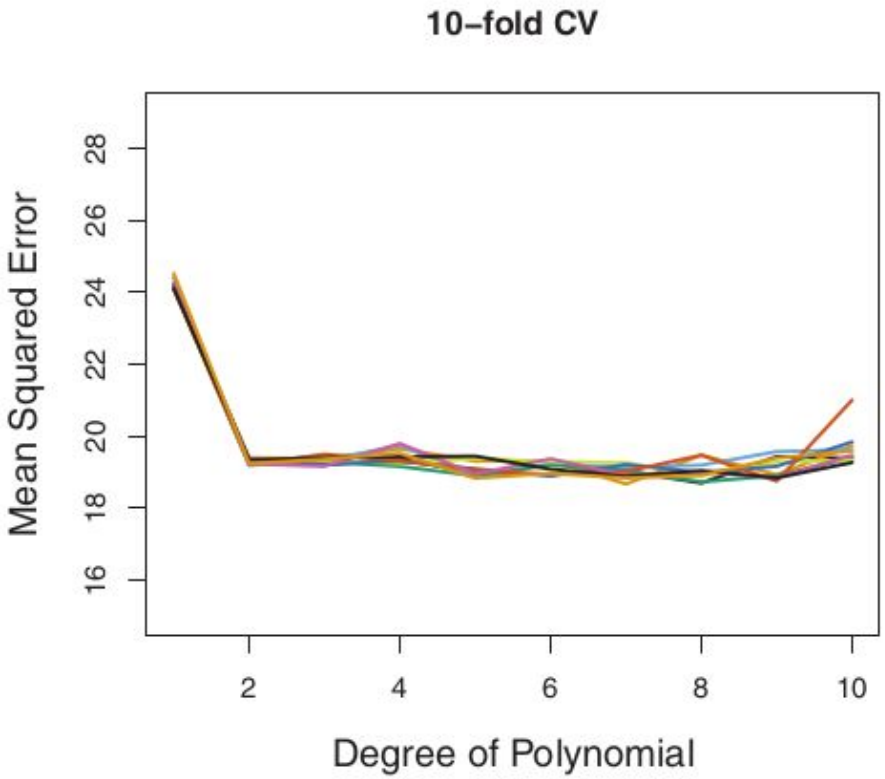
Leave one out cross validation (LOOCV)



K-fold cross validation

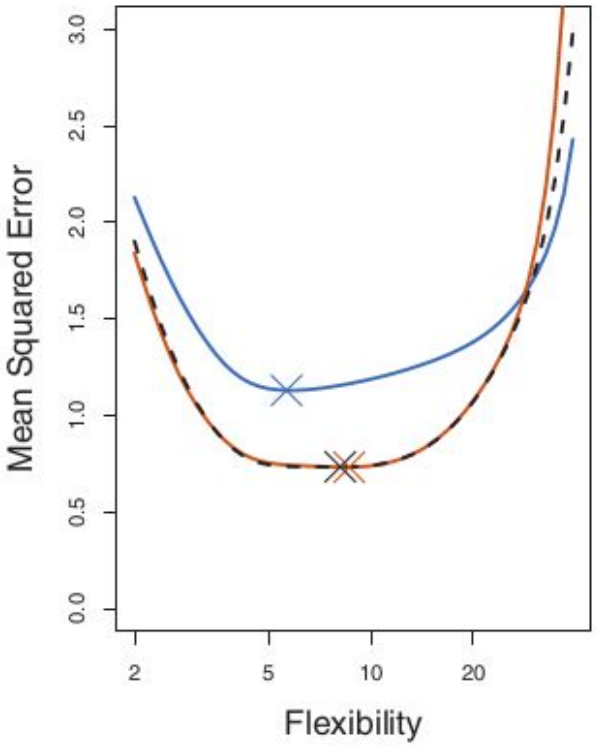


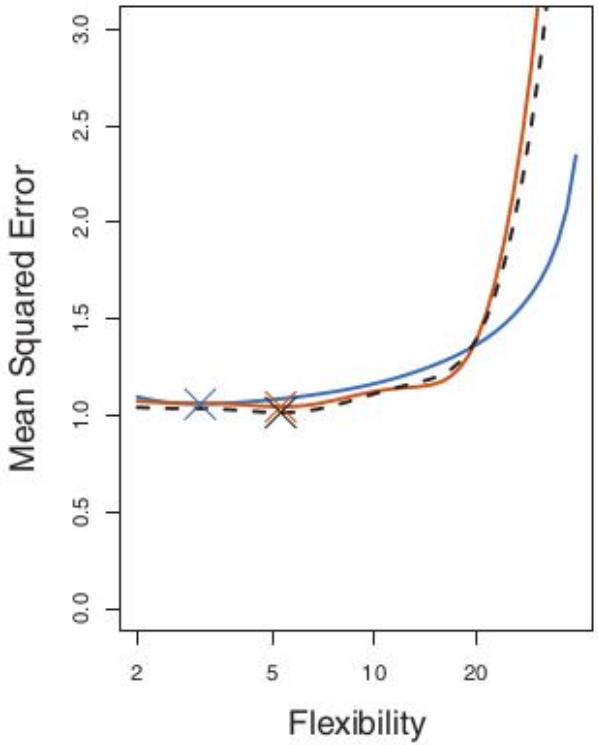


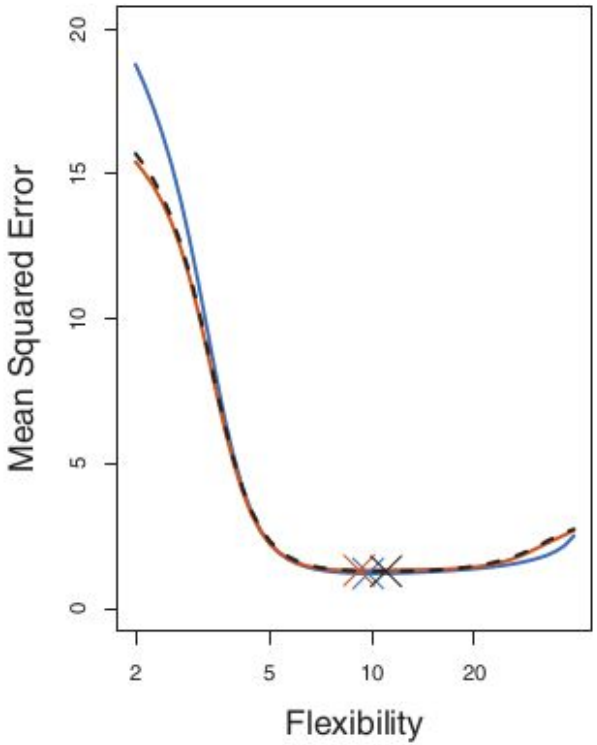


- LOOCV -- can be computationally expensive

True test MSE
LOOCV test MSE estimate
10-fold CV test MSE estimate







"True" *f* is non-linear

"True" *f* is linear

"True" *f* is very non-linear

Here we tried to find the best hyperparameter for our model e.g. number of smoothing splines, degree of polynomial function

What if we want to compare two different models?

Training, validation and testing sets

- If we like to be rigorous (like for example in scientific article), then the statisticians would say that beside training and validation sets we also need testing dataset.
- Testing dataset is held out from the beginning and not used in creating model at all. We use it only after we are satisfied with our model and are ready to report the final test error.
- We fit the model (select the best hyperparameters) using training and validation sets and some cross validation strategy.

Model assessment

Different assessment statistics for regression and classification

Regression:

$$trainMSE = \frac{1}{n}\sum_{i=n}^{n}(y_i - \hat{f}(x_i))^2$$

Where n is the number of data points, y_i is the actual value for data point i and hat $f(x_i)$ -- value returned by the model

$$testMSE = \frac{1}{n}\sum_{i=1}^{n}(y_0 - \hat{f}(x_0))^2$$

Where y_0 and x_0 belong to unseen data

Classification:

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

Error rate -- the proportion of mistakes that are made if we apply our estimated *f* function to data points

Can be train and test Error rate the same as MSE

Classification result representation: confusion table

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Confusion table

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

https://en.wikipedia.org/wiki/Confusion_matrix

Confusion table statistics

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

https://en.wikipedia.org/wiki/Confusion_matrix