DATA SCIENCE

MODEL EVALUATION

I. EVALUATION PROCEDURES

II. EVALUATION METRICS

I. EVALUATION PROCEDURES

Q: What's wrong with training error?

Thought experiment:

Suppose we train our model using the entire dataset.

Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

A: Down to zero!

Q: What's wrong with training error?

Thought experiment:

Suppose we train our model using the entire dataset.

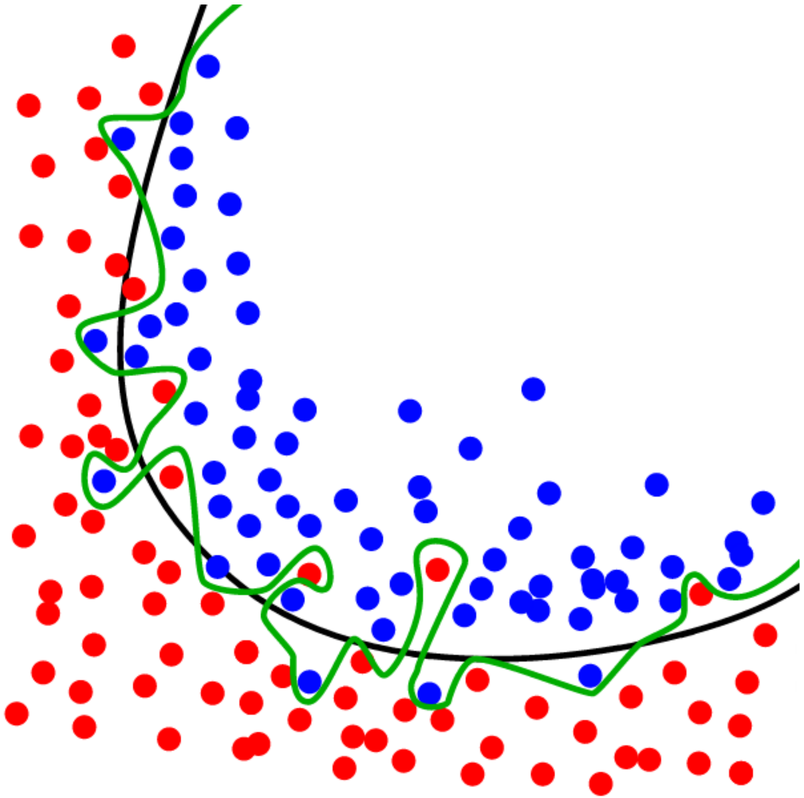
Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

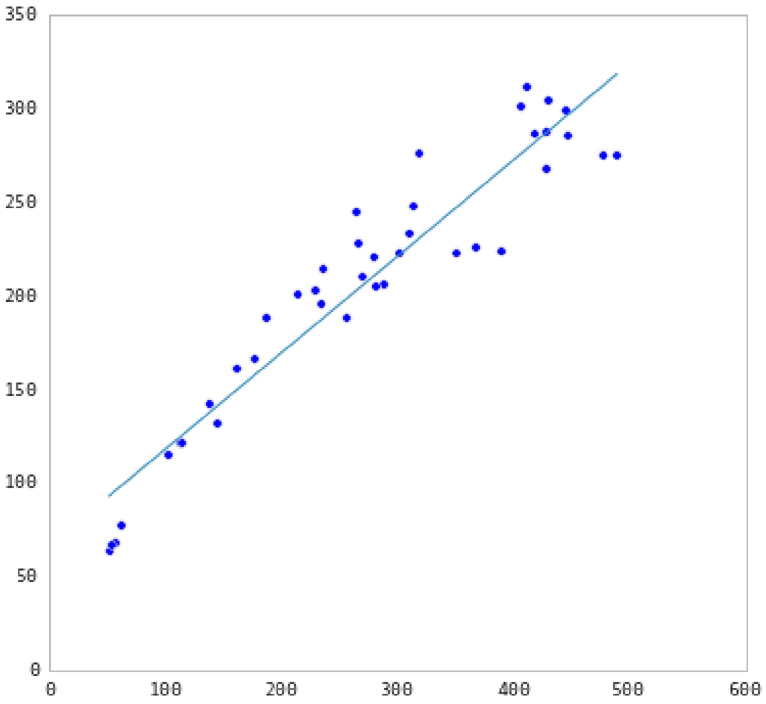
A: Down to zero!

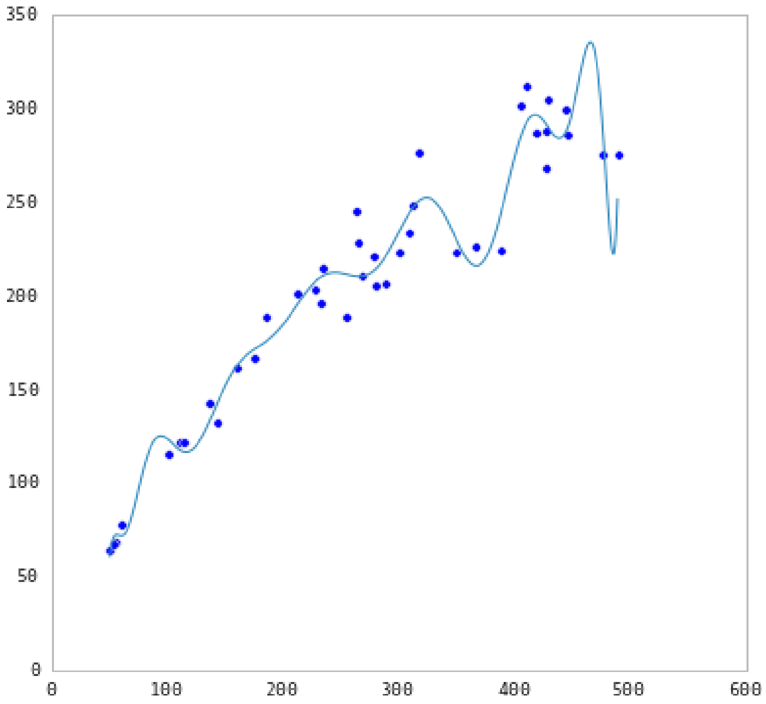
NOTE

This phenomenon is called *overfitting*.



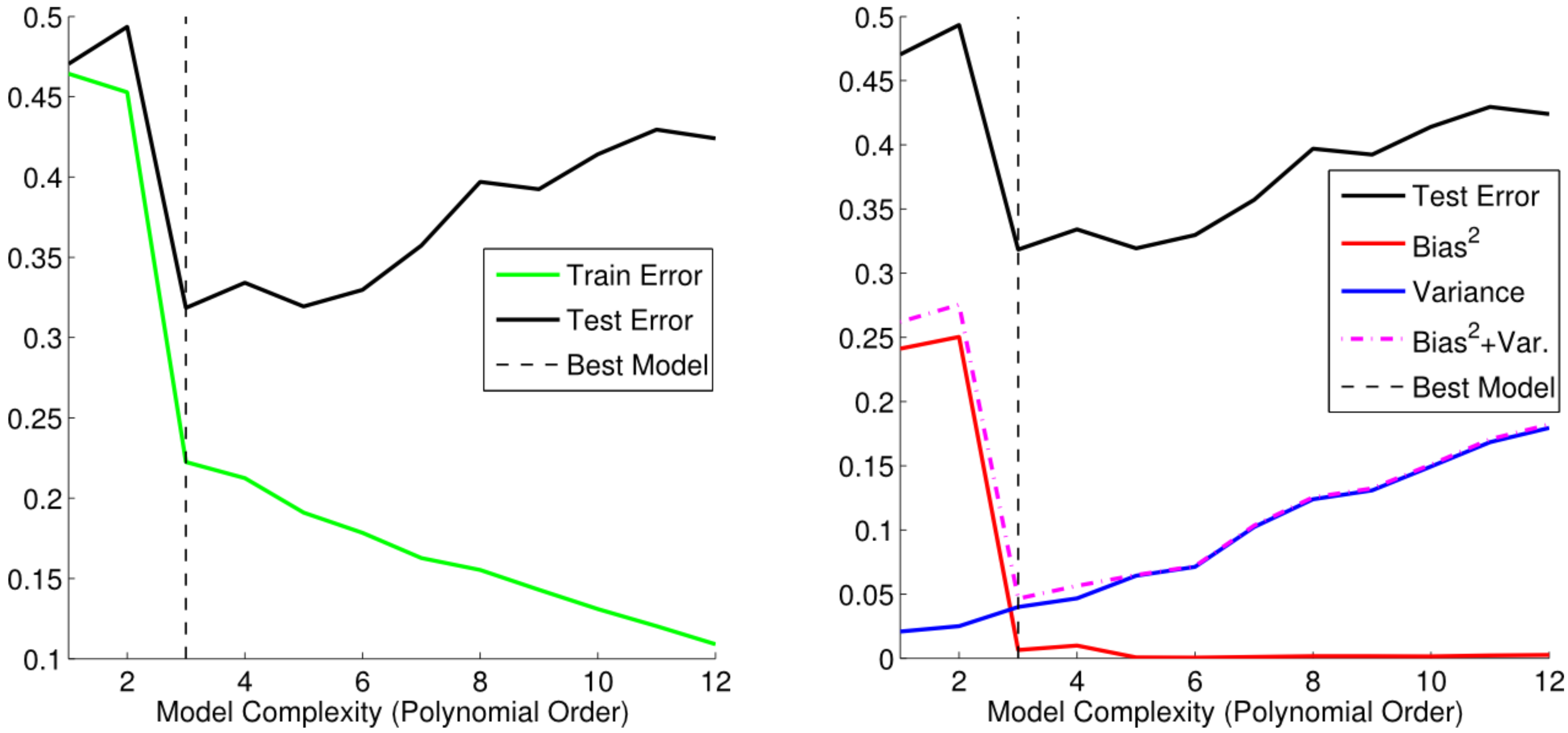
Source: http://www.dtreg.com





Source: http://nbviewer.ipython.org/github/fonnesbeck/Bios366/blob/master/notebooks/Section6_3-Model-Selection-and-Validation.ipynb

UNDERFITTING AND OVERFITTING



Source: http://theclevermachine.wordpress.com/2013/04/21/model-selection-underfitting-overfitting-and-the-bias-variance-tradeoff/

Q: What's wrong with training error?

Thought experiment:

Suppose we train our model using the entire dataset.

Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

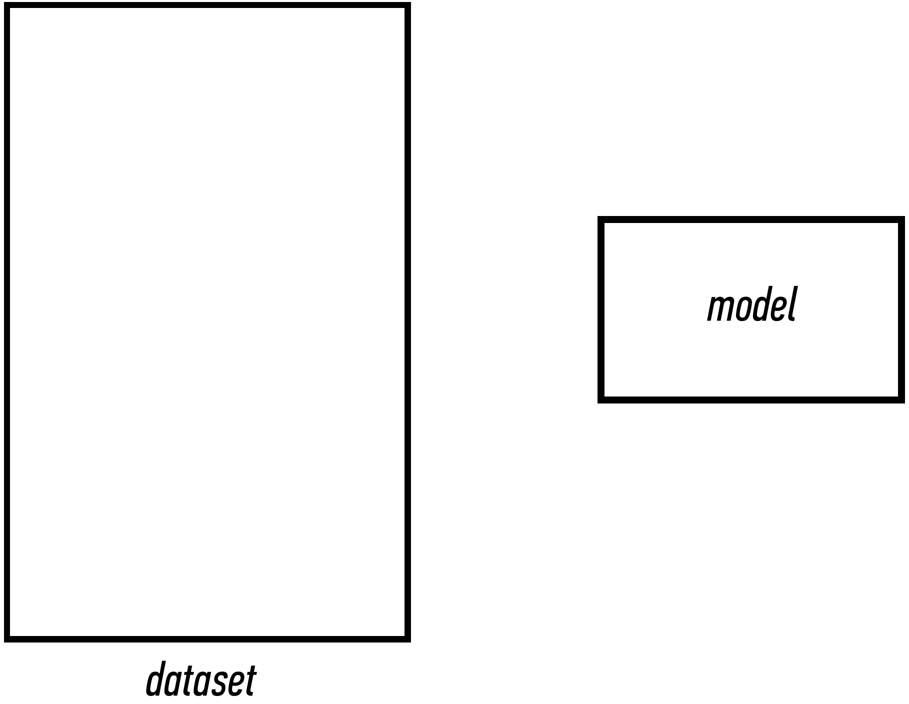
A: Down to zero!

NOTE

This phenomenon is called *overfitting*.

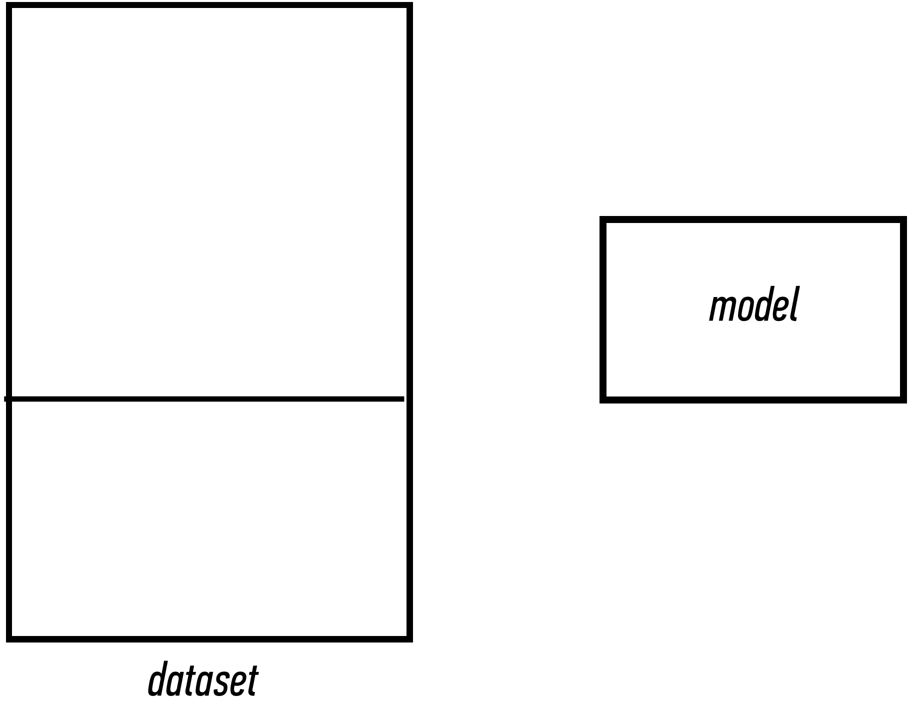
A: Training error is not a good estimate of accuracy beyond training data.

Q: How can we make a model that generalizes well?



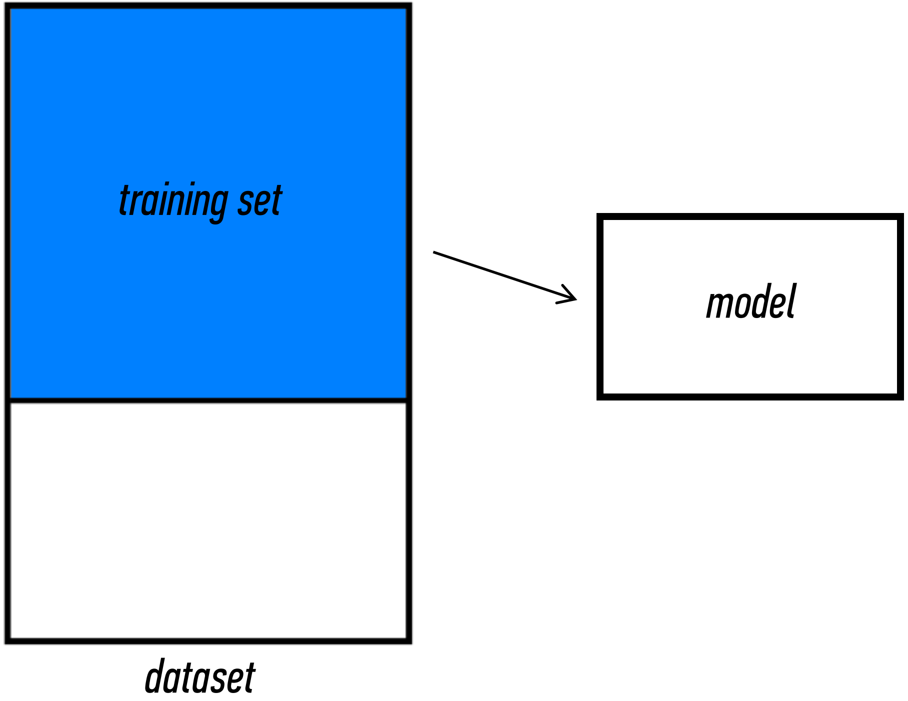
Q: How can we make a model that generalizes well?

1) *split dataset*



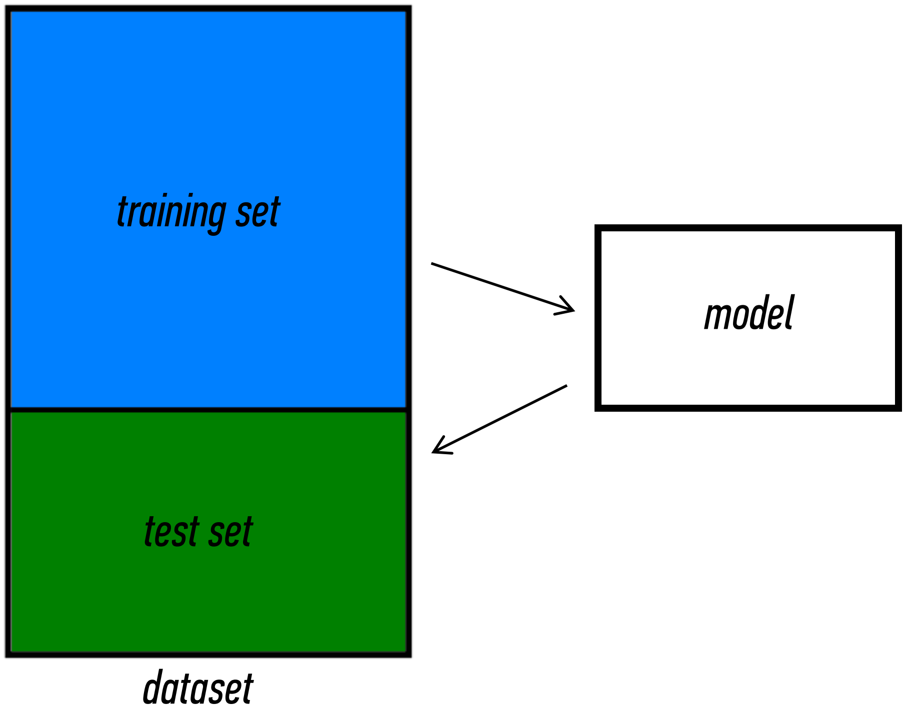
Q: How can we make a model that generalizes well?

1) split dataset
2) train model



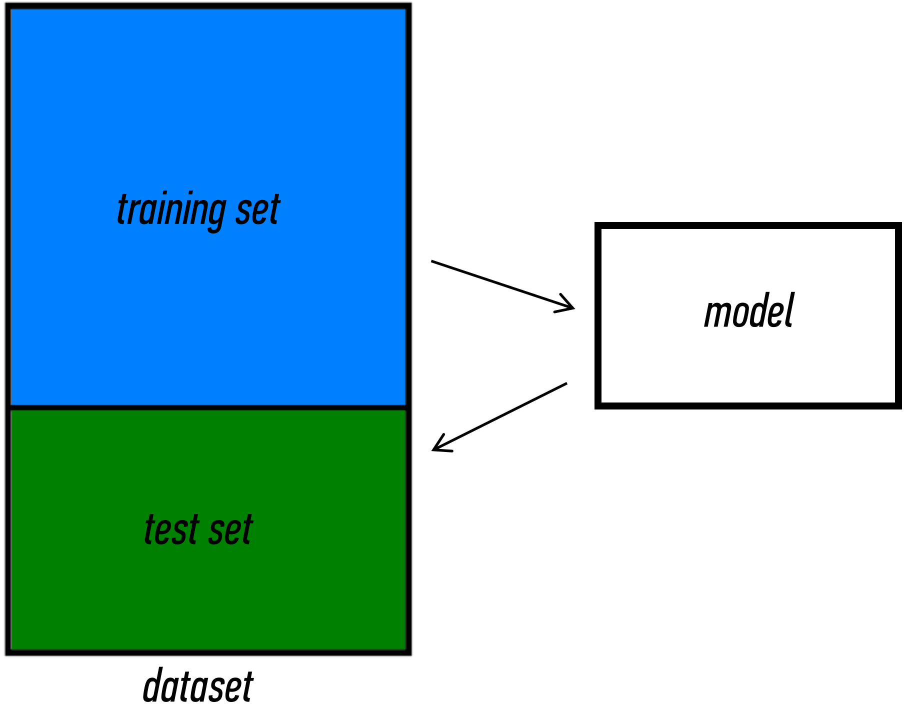
Q: How can we make a model that generalizes well?

1) split dataset
2) train model
3) test model



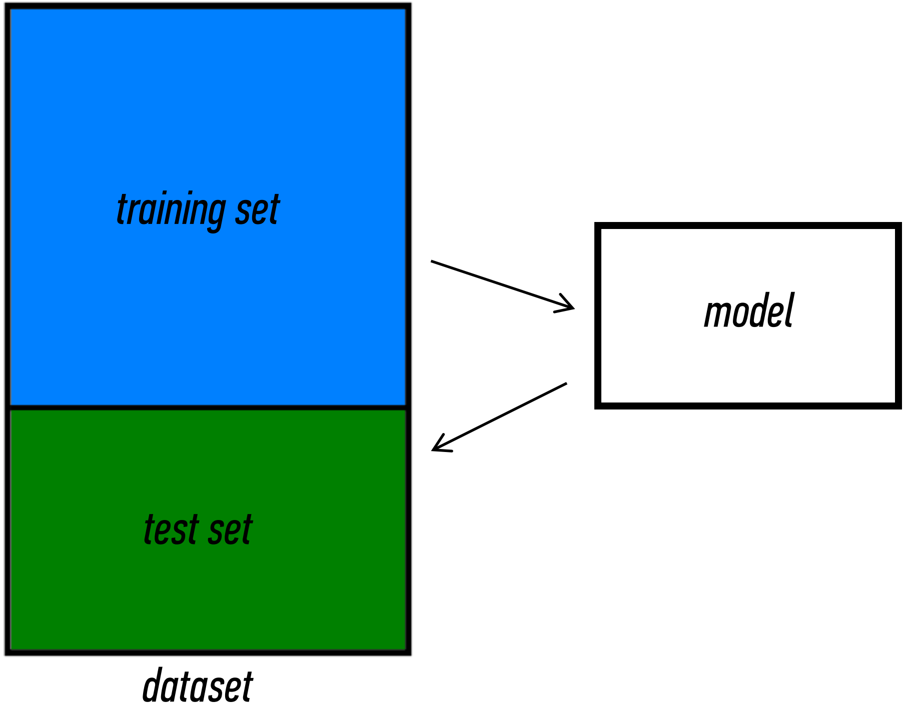
Q: How can we make a model that generalizes well?

1) split dataset
2) train model
3) test model
4) parameter tuning



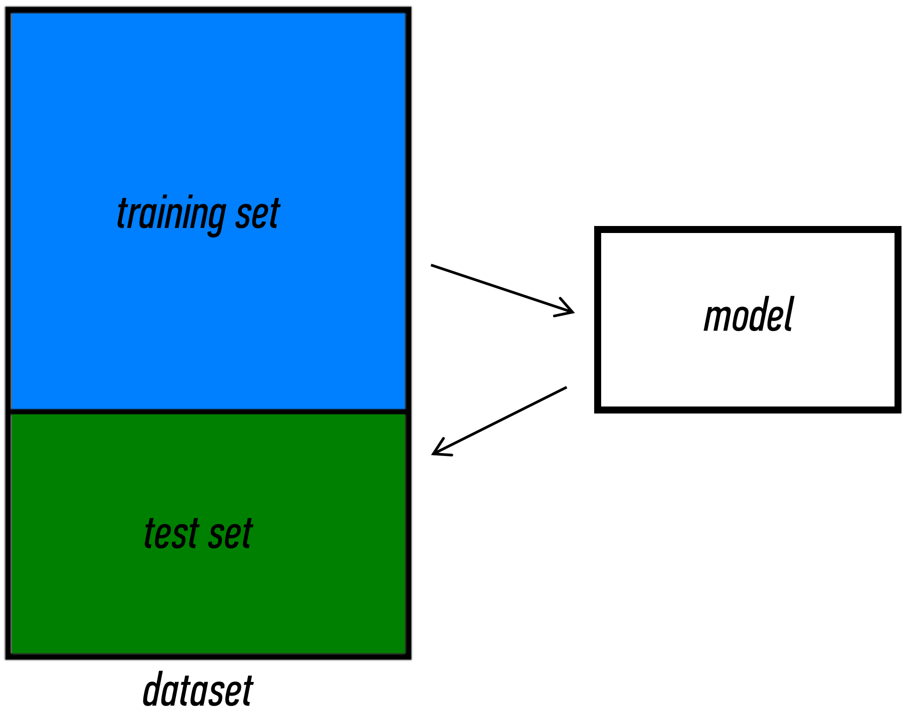
Q: How can we make a model that generalizes well?

1) split dataset
2) train model
3) test model
4) parameter tuning
5) choose final model



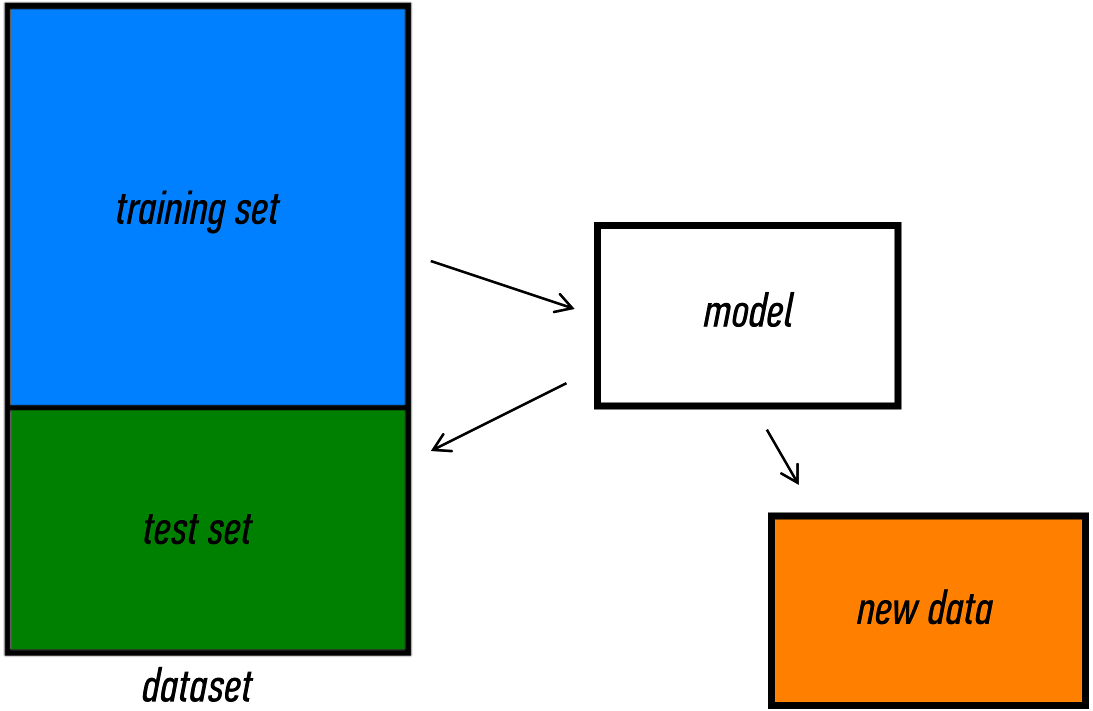
Q: How can we make a model that generalizes well?

1) *split dataset*
2) *train model*
3) *test model*
4) *parameter tuning*
5) *choose final model*
6) *train on all data*



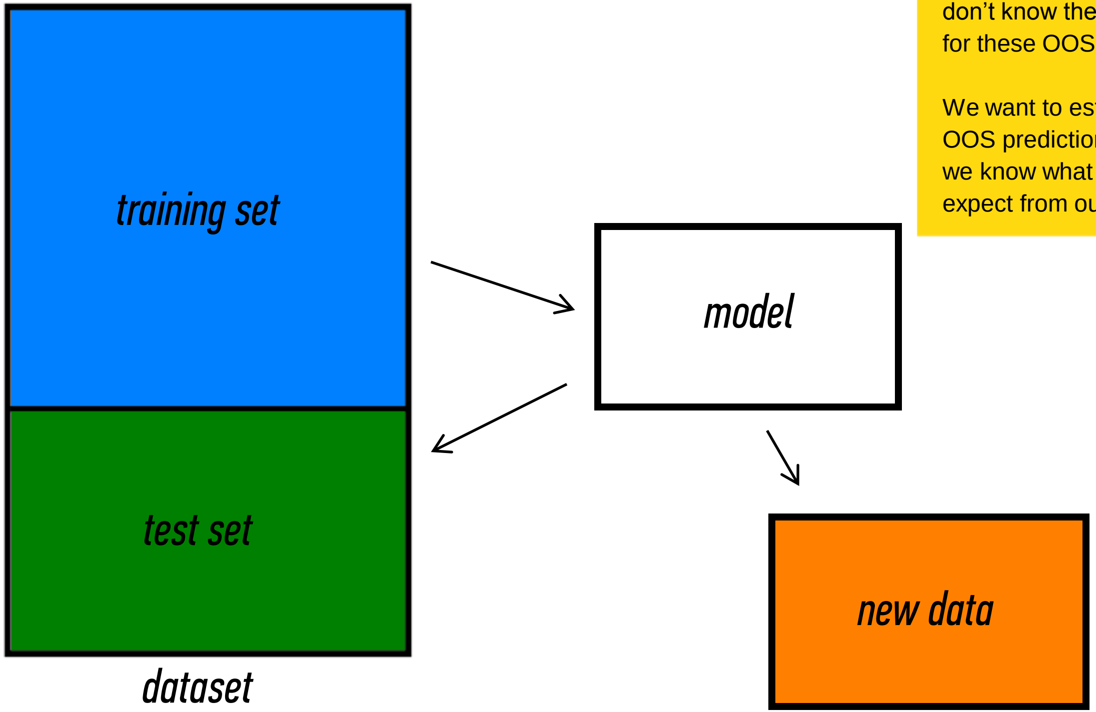
Q: How can we make a model that generalizes well?

1) *split dataset*
2) *train model*
3) *test model*
4) *parameter tuning*
5) *choose final model*
6) *train on all data*
7) *make predictions*



Q: How can we make a model that generalizes well?

1) *split dataset*
2) *train model*
3) *test model*
4) *parameter tuning*
5) *choose final model*
6) *train on all data*
7) *make predictions*



NOTE

This new data is called *out of sample* data. We don't know the labels for these OOS records!

We want to estimate OOS prediction error so we know what to expect from our model.

Suppose we do the train/test split.

Q: How well does test set error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the test set error remain the same?

A: Of course not!

A: On its own, not very well.

Suppose we do the train/test split.

Q: How well does test set error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the test set error remain the same?

A: Of course not!

A: On its own, not very well.

NOTE

The test set error gives a *high-variance estimate* of OOS accuracy.

Something is still missing!
Q: How can we do better?

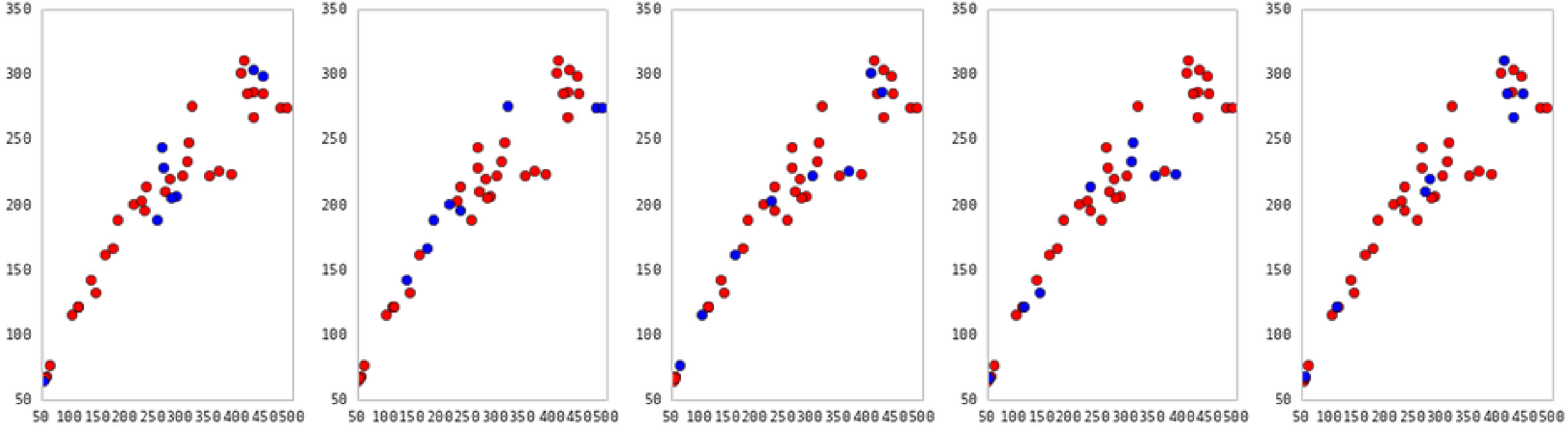
Thought experiment:
Different train/test splits will give us different test set errors.
Q: What if we did a bunch of these and took the average?
A: Now you're talking!

A: Cross-validation.

Steps for K-fold cross-validation:

1) *Randomly split the dataset into K equal partitions.*
2) *Use partition 1 as test set & union of other partitions as training set.*
3) *Calculate test set error.*
4) *Repeat steps 2-3 using a different partition as the test set at each iteration.*
5) *Take the average test set error as the estimate of OOS accuracy.*

CROSS-VALIDATION



5-fold cross-validation: red = training folds, blue = test fold

Source: http://nbviewer.ipython.org/github/fonnesbeck/Bios366/blob/master/notebooks/Section6_3-Model-Selection-and-Validation.ipynb

Features of K-fold cross-validation:

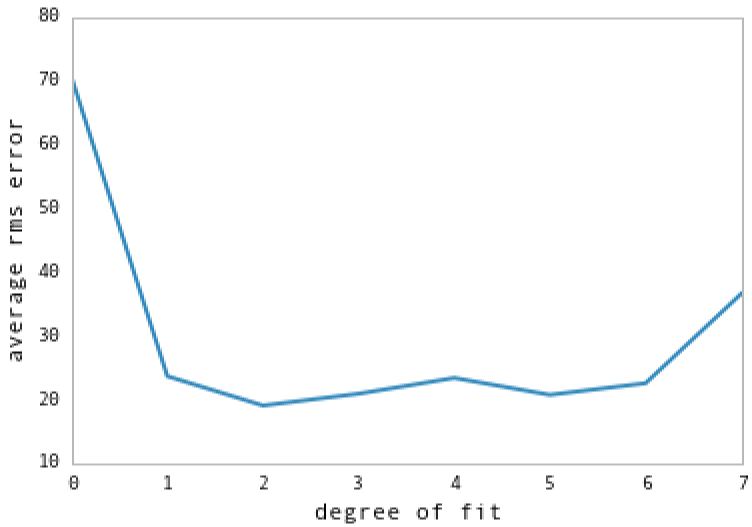
1) *More accurate estimate of OOS prediction error.*
2) *More efficient use of data than single train/test split.*
   - *Each record in our dataset is used for both training and testing.*
3) *Presents tradeoff between efficiency and computational expense.*
   - *10-fold CV is 10x more expensive than a single train/test split*
4) *Can be used for parameter tuning and model selection.*

Features of K-fold cross-validation:

1) *More accurate estimate of OOS prediction error.*
2) *More efficient use of data than single train/test split.*
   - *Each record in our dataset is used for both training and testing.*
3) *Presents tradeoff between efficiency and computational expense.*
   - *10-fold CV is 10x more expensive than a single train/test split*
4) *Can be used for parameter tuning and model selection.*

NOTE

Leave one out cross-validation (LOOCV) is a special case of K-fold cross-validation.



Model selection using cross-validation:
lowest predicted OOS error at degree = 2

Source: http://nbviewer.ipython.org/github/fonnesbeck/Bios366/blob/master/notebooks/Section6_3-Model-Selection-and-Validation.ipynb

II. EVALUATION METRICS

Classification:

- *Confusion Matrix*
- *ROC Curve (and AUC)*

Regression:

- *Root Mean Squared Error*

Confusion Matrix: table to describe the performance of a classifier

n=165	Actual: YES	Actual: NO
Predicted: YES	100	10
Predicted: NO	5	50

Example: Test for presence of disease
YES = positive test = True = 1
NO = negative test = False = 0

- *How many classes are there?*
- *How many patients?*
- *How many predictions of disease?*
- *How many patients actually have the disease?*

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

Basic Terminology:

- *True Positives (TP)*
- *True Negatives (TN)*
- *False Positives (FP)*
- *False Negatives (FN)*

Accuracy:

- *Overall, how often is it correct?*
- *(TP + TN) / total = 150/165 = 0.91*

Misclassification Rate (Error Rate):

- *Overall, how often is it wrong?*
- *1 - accuracy = 1 - 0.91 = 0.09*

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

True Positive Rate:

- *TP / actual yes = 100/105 = 0.95*
- *"sensitivity" or "recall"*

False Positive Rate:

- *FP / actual no = 10/60 = 0.17*

Precision:

- *TP / predicted yes = 100/110 = 0.91*

Specificity:

- *1 - FPR = 1 - 0.17 = 0.83*

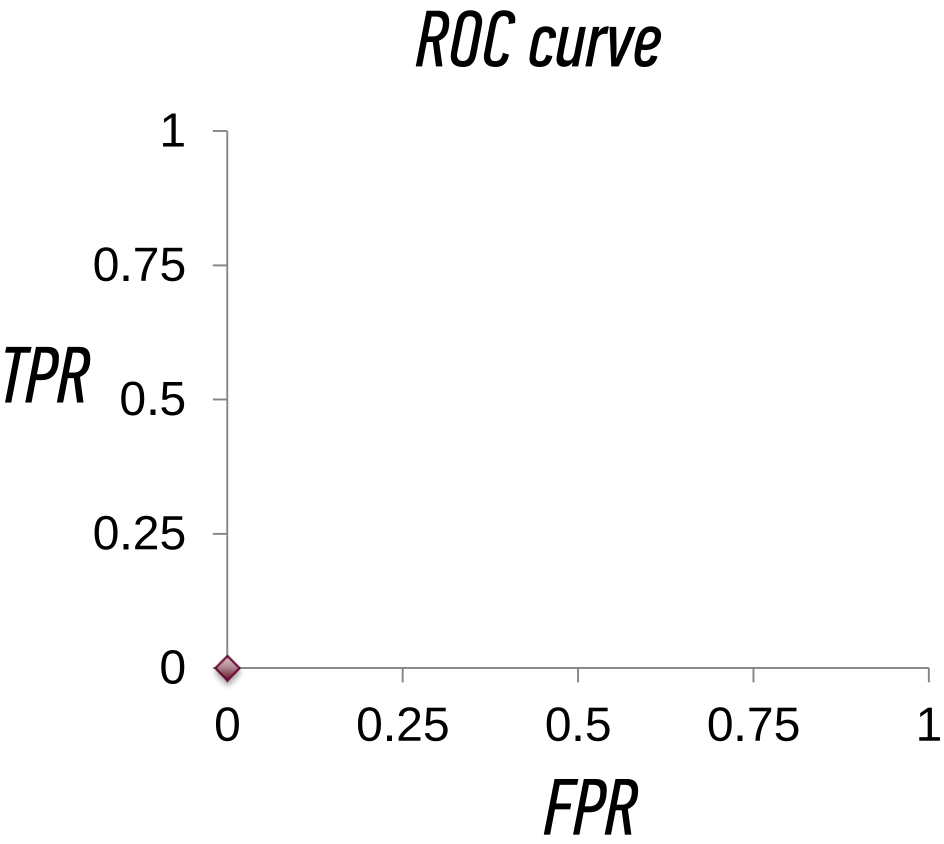
Email Number	Score	True Label
5	0.93	Spam
8	0.91	Spam
2	0.84	Spam
1	0.6	Ham
7	0.54	Spam
3	0.22	Ham
4	0.10	Ham
6	0.02	Ham

Every email gets a spamminess score.

Choosing a cut-off, this becomes a classification.

How do we choose a cut-off?
How do we evaluate the ranking without choosing a cut-off?

Email Number	Score	True Label
5	0.93	Spam
8	0.91	Spam
2	0.84	Spam
1	0.6	Ham
7	0.54	Spam
3	0.22	Ham
4	0.10	Ham
6	0.02	Ham



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

- *Used for regression problems*
- *Square root of the mean of the squared errors*
- *Easily interpretable (in the "y" units)*
- *"Punishes" larger errors*

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

Example:

y_true = [100, 50, 30]

y_preds = [90, 50, 50]

*RMSE = np.sqrt((10**2 + 0**2 + 20**2) / 3) = 12.88*

DATA SCIENCE