Machine Learning in R

Cross-Validation and Nested Cross-Validation

John Kornak – Epidemiology and Biostatistics

john.kornak@ucsf.edu

Estimating and evaluating "best" models without overfitting

- How to choose between different models/algorithms?
- Idea is to use "prediction quality" (e.g. MSE) to assess what level of complexity is best given a particular dataset
- But we can't use the same data we fit the model to in order to assess how well it did – could be perfect fit but not generalize to other data – e.g. fitting a n-degree polynomial to n observations
- So we deliberately leave data aside for "testing"…

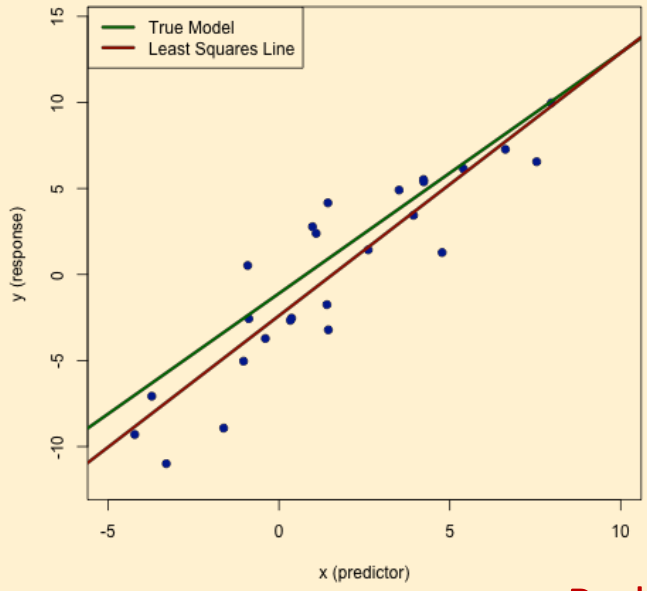
Training/test

- Randomly split data into *training* and *test* sets (often ~2/3 for train, 1/3 for test)
- E.g., fit a linear regression model using the training set and evaluate it using Mean Squared Error (MSE) on the test set

Training/test

- Fit model to *training* data
- Evaluate on *test* data

E.g. Linear regression

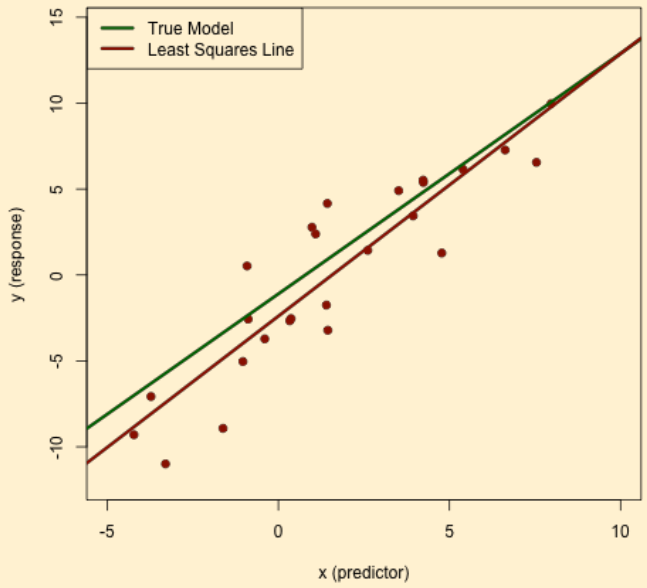


Red line is *overfitting* to this specific dataset

Training/test

- Fit model to *training* data
- Evaluate on *test* data

E.g. Linear regression

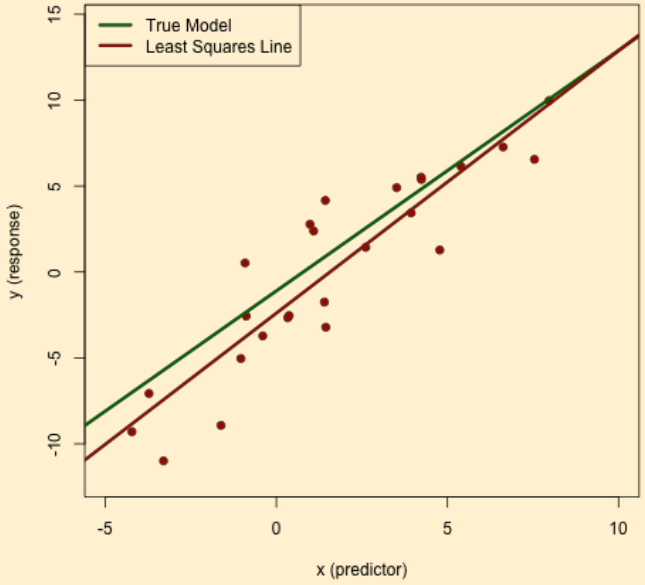


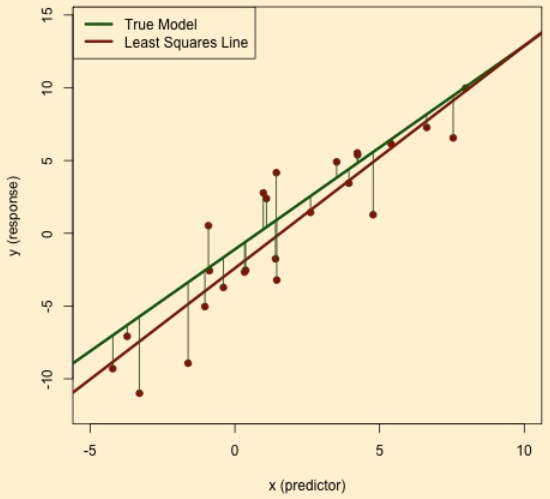
Red line is *overfitting* to the training dataset

Training/test

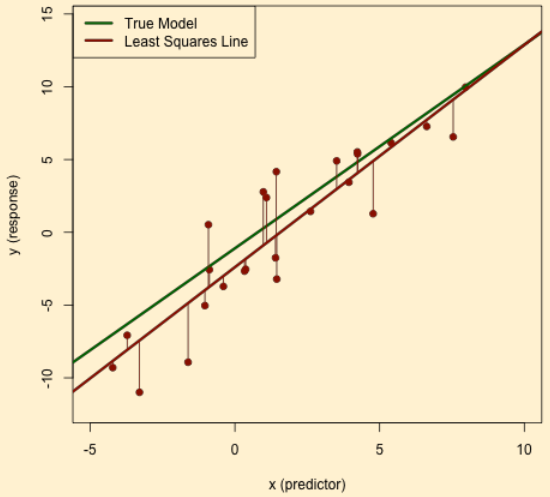
- Fit model to *training* data
- Evaluate on *test* data

E.g. Linear regression





True noise Var is 9.0. MSE to true line is 8.25



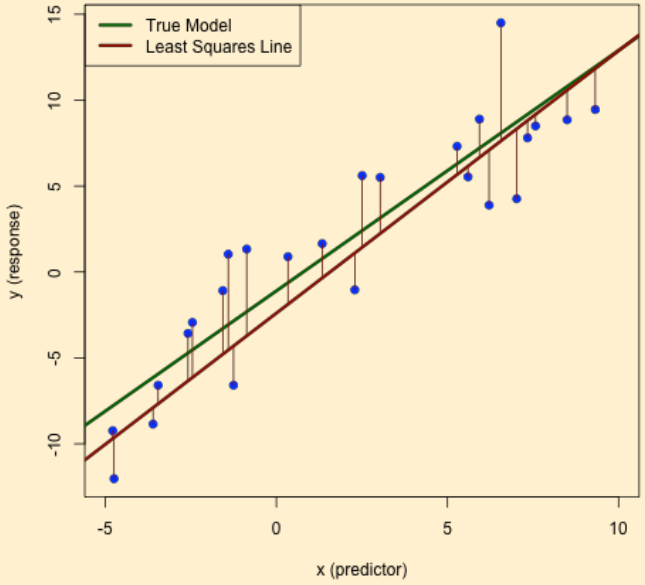
Train MSE is 6.62 (underesti mated – i.e. overfitted)

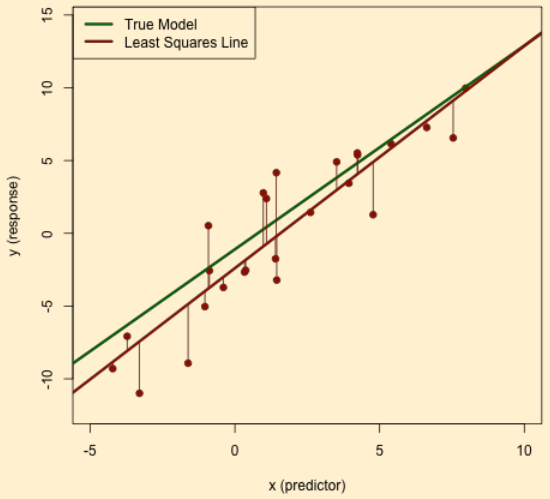
Red line is *overfitting* to the training dataset

Training/test

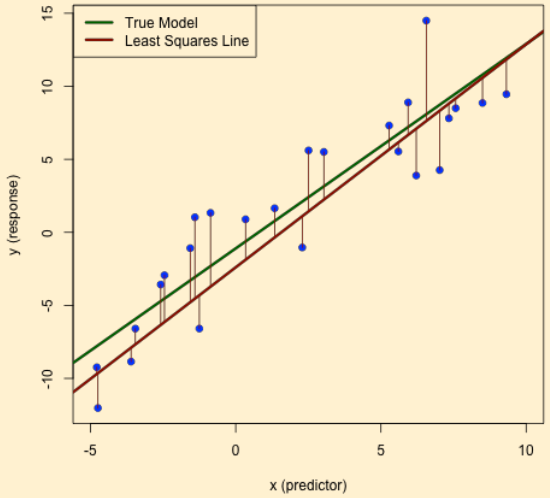
- Fit model to *training* data
- Evaluate on *test* data

E.g. Linear regression





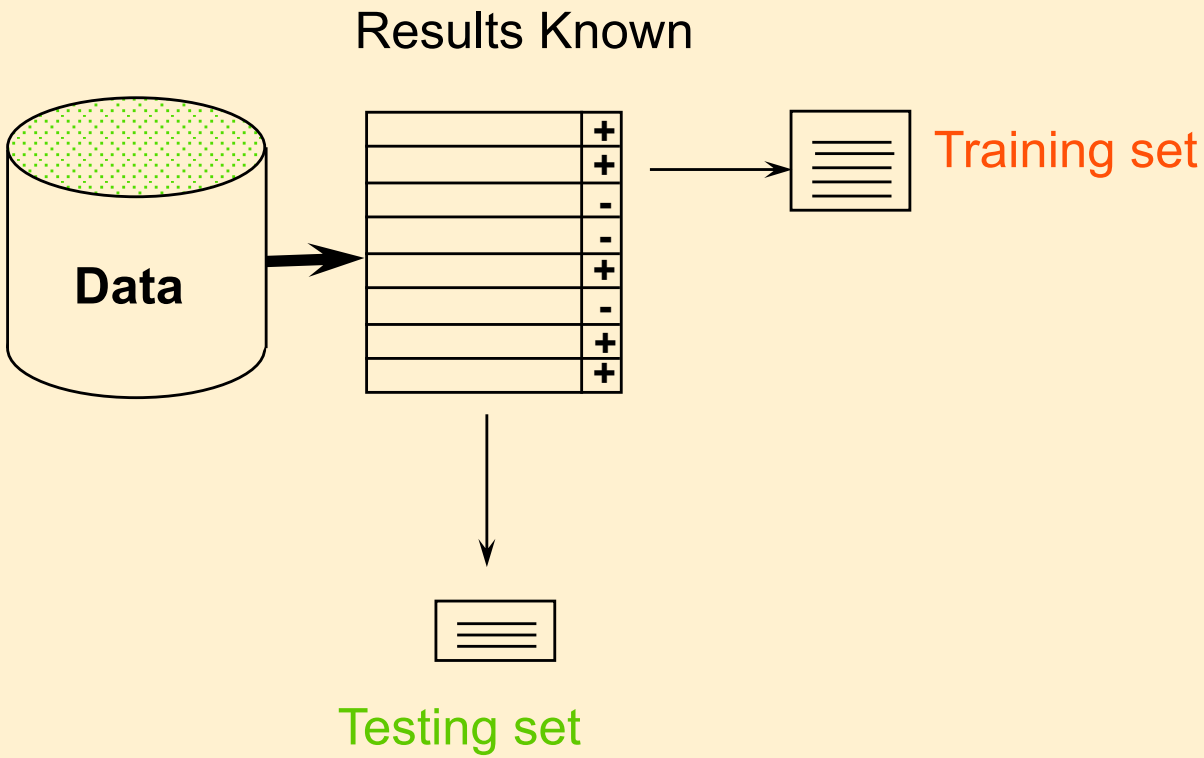
Train MSE is 6.62 (underesti mated – i.e. overfitted)



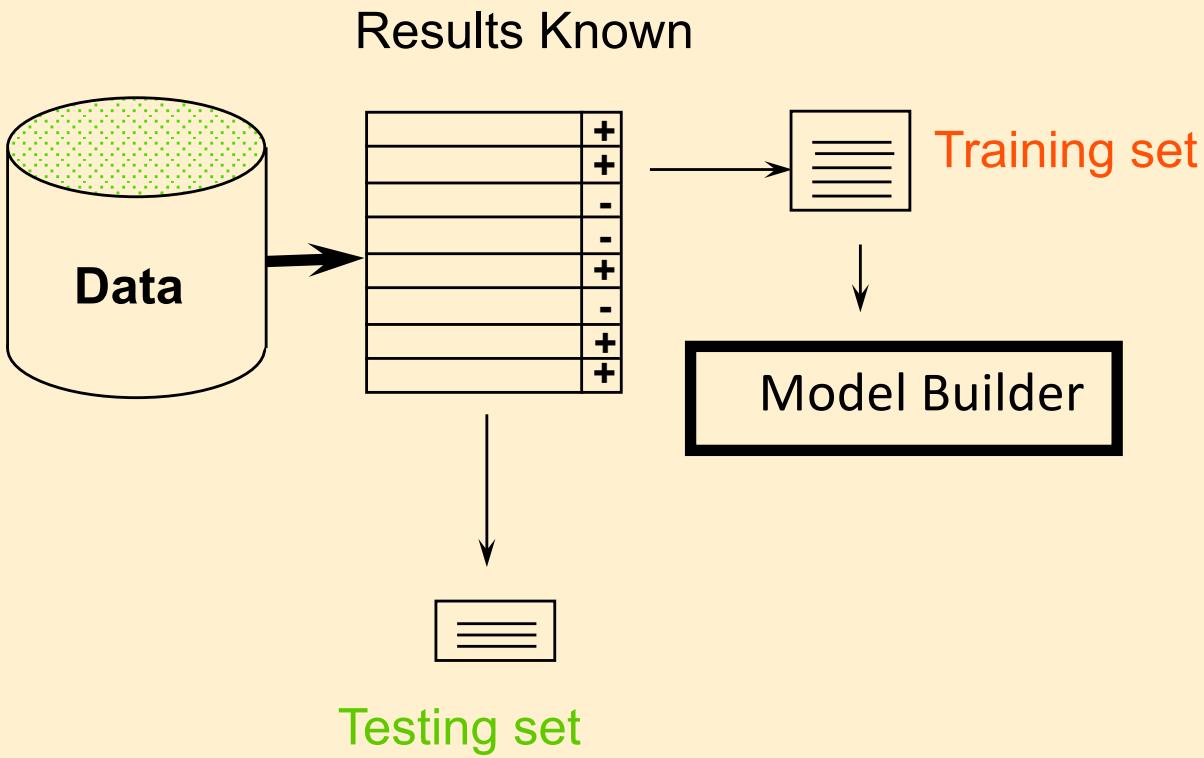
Test MSE is 9.52 (not underesti mated i.e. not overfitted)

Red line not *overfitting* to the test dataset

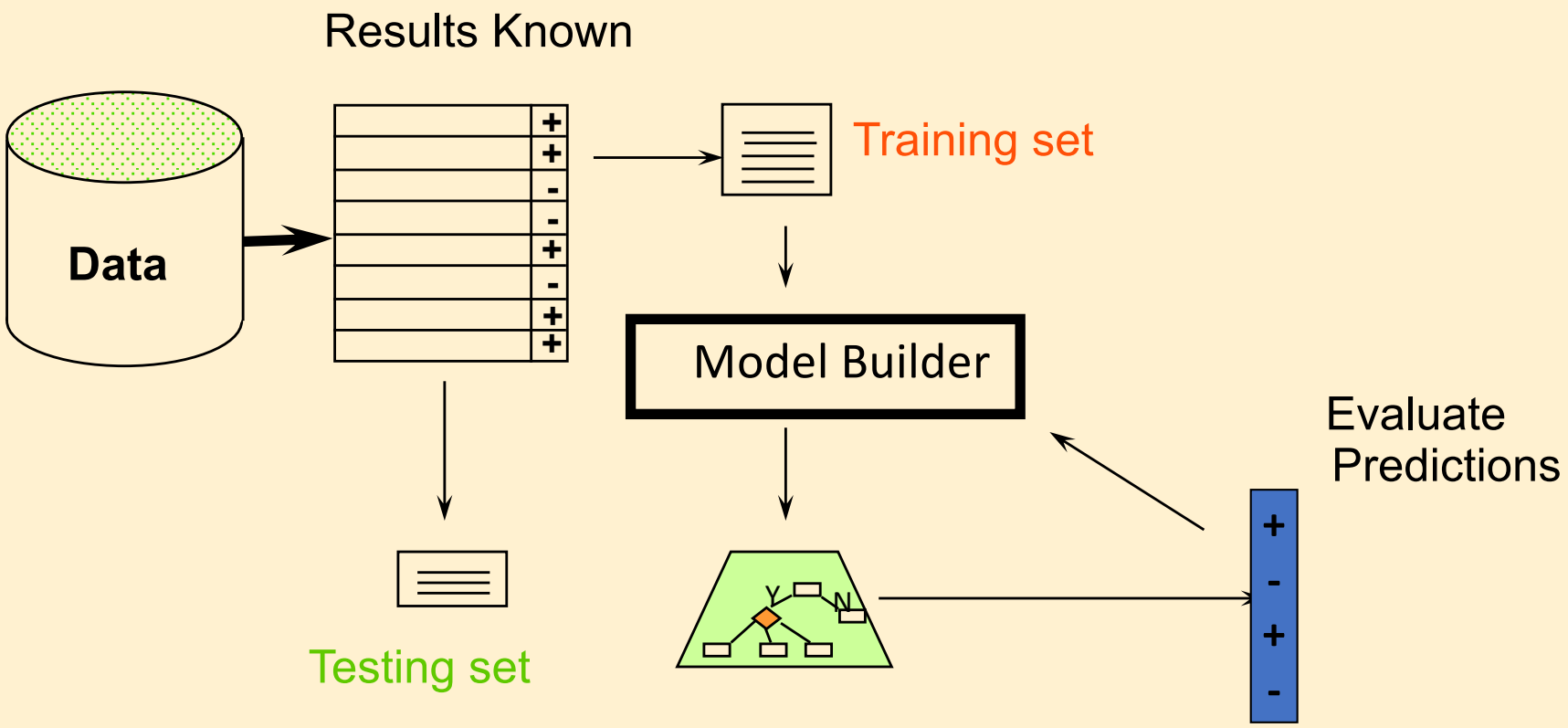
Training/test



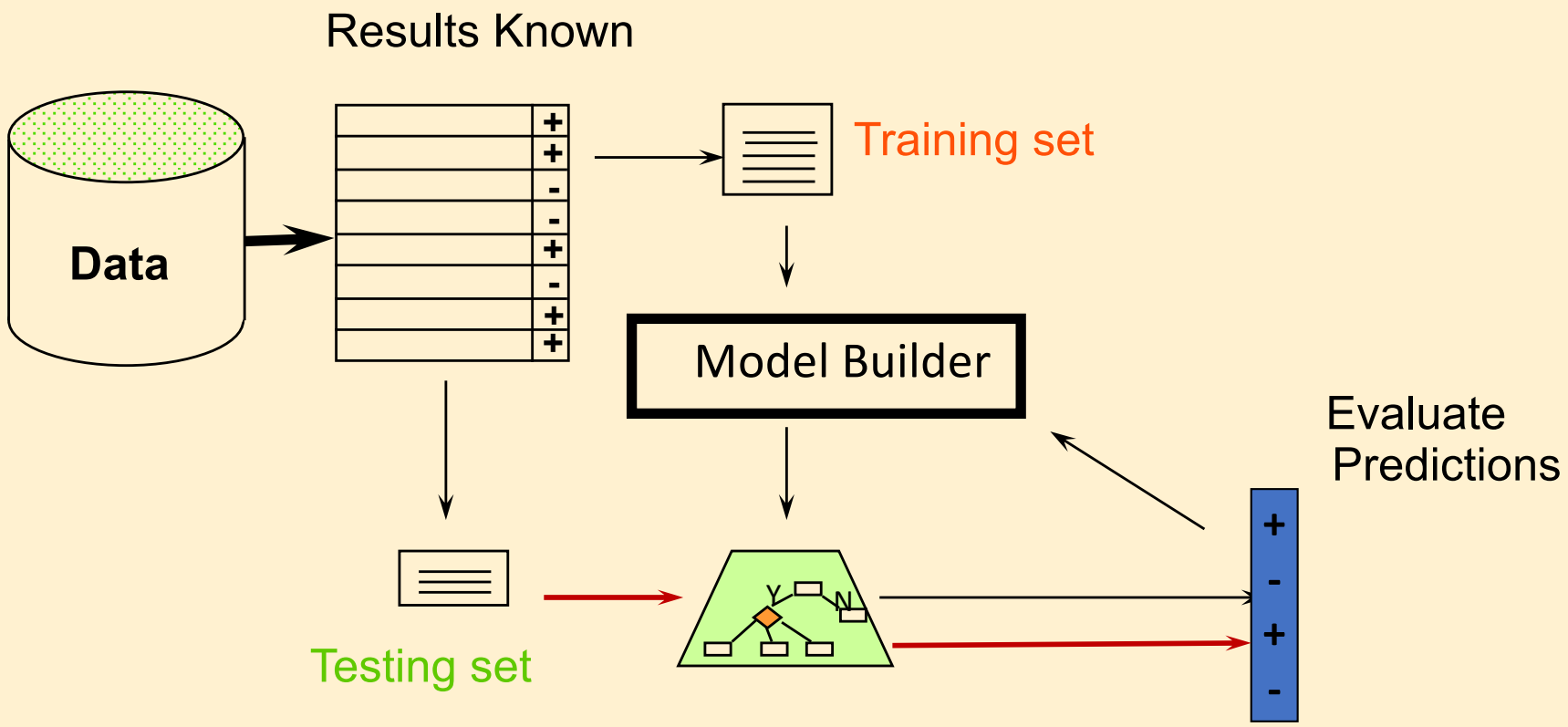
Training/test



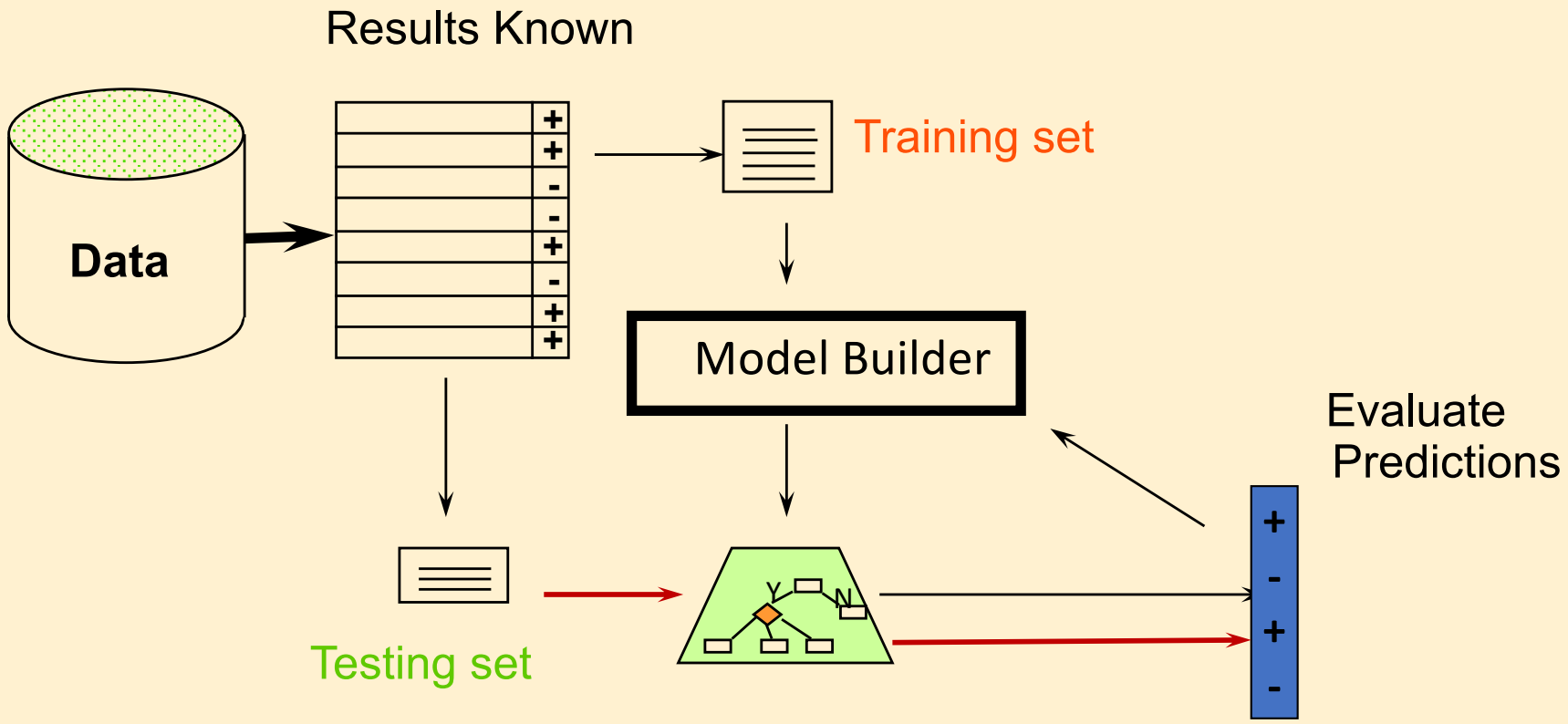
Training/test



Training/test



Training/test



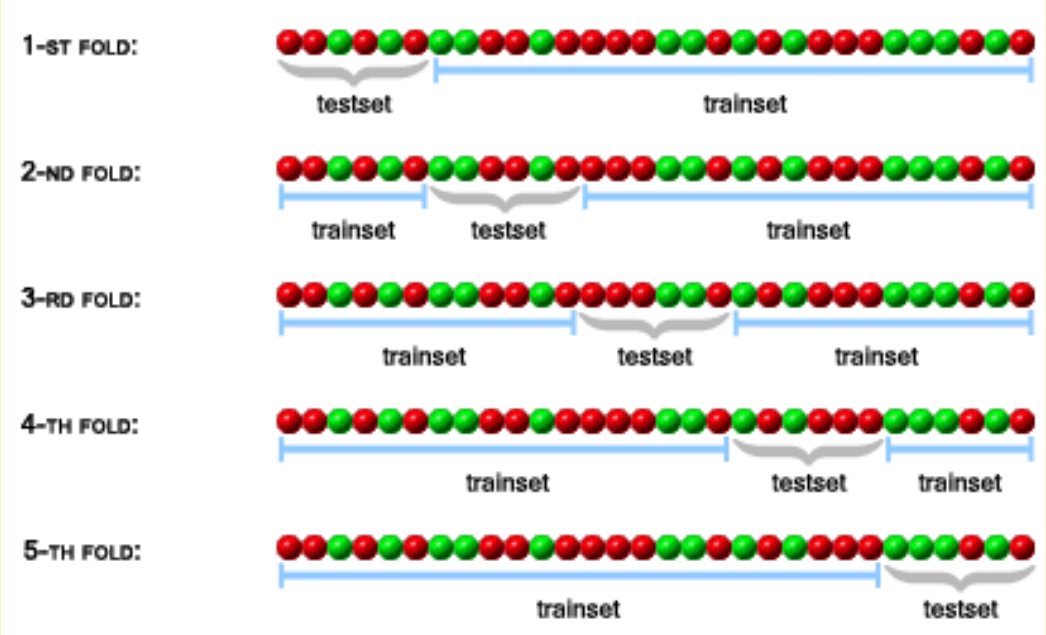
This is great if you have a lot of data to be able to reserve a testing set, but what if you don't?

Cross-validation for training/test

- *Cross-validation*
  - First step: data is split into *k* subsets of equal size
  - Second step: each subset is used in turn for validation and the remainder for training
- This is called *k-fold cross-validation* (10-fold is common choice, also 5-fold for smaller datasets approx. <200 and even *N-fold* for very small datasets, called leave-one-out)
- Often the subsets are stratified before the cross-validation is performed – e.g. same proportion of each diagnostic class in each fold
- The error estimates are averaged across folds to yield an overall error estimate

E.g. 5-fold cross-validation

- Break up data into 5 groups of the same size
- Hold aside one group for testing and use the rest to build model
- Repeat



- When every group has had a turn at being the test set then you can estimate the accuracy (MSE or classification) based on the combined Test data (i.e. the 5 test-sets)

Cross-validation for training/test

- Notice that at no time is the classification accuracy based on data that was used to fit or choose the model – so there is no over-fitting
- And all of the data gets used in the process unlike training/test
- As a final step you can then rebuild the classifier using all the data for predicting future observations – this model will (on average) be better than any of the individual models in the cross-validation because it uses more data

The issue of tuning parameters

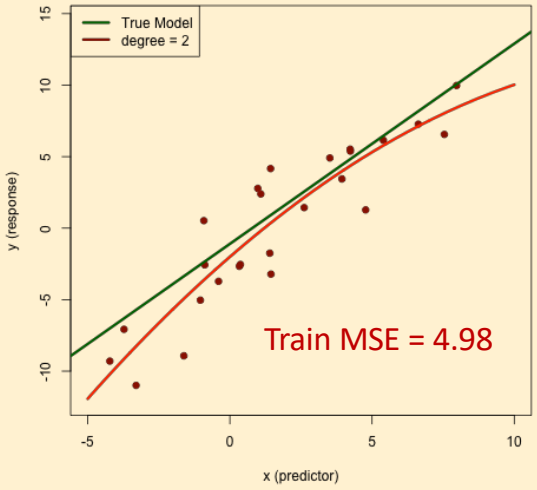
- But linear regression is a special case – there are no tuning parameters – most methods have them – e.g. order for polynomials, cost for support vector machines, “mtry” for random forests, shrinkage penalty λ for penalized regression etc. etc.
- How does training/test work in the presence of tuning parameters?

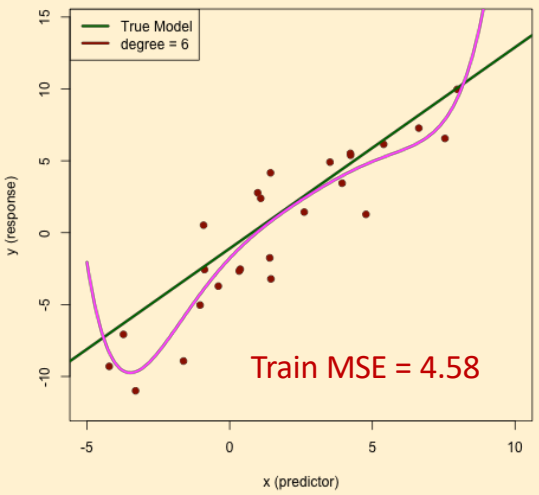
Polynomial example

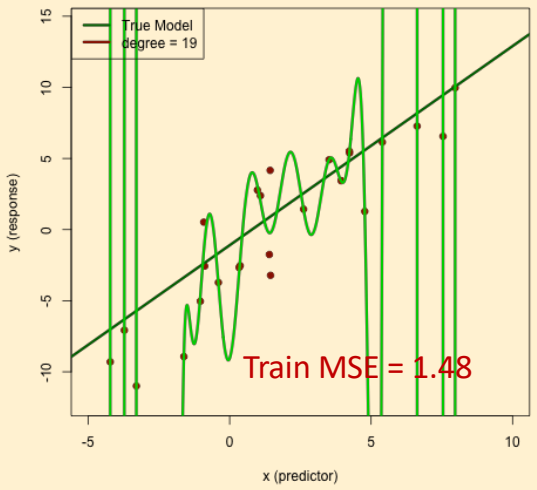
- Assume we have training set and test set as before (i.e. generated as linear model with noise)
- But we don’t know that and we want to find the optimal polynomial fit
- We want to reserve the test data to independently evaluate fit
- So we work on finding the best polynomial in the training data based on MSE to evaluate in the test set

Polynomial fits to training data

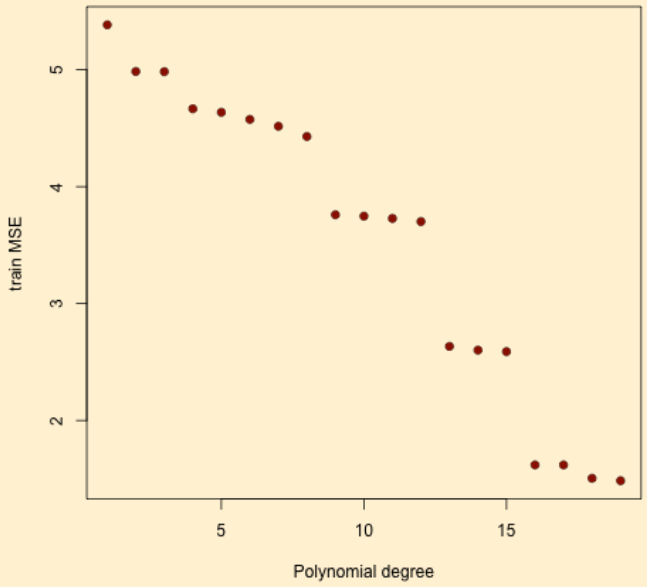
Varying the degree of the fitted polynomial (the degree is the tuning parameter)





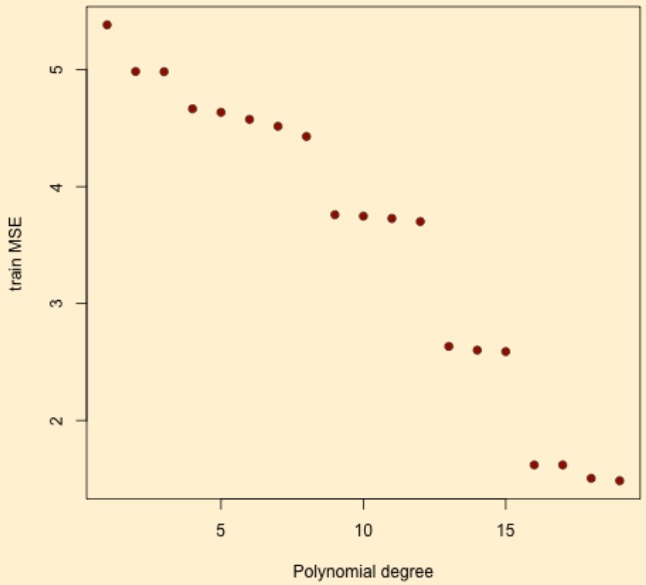


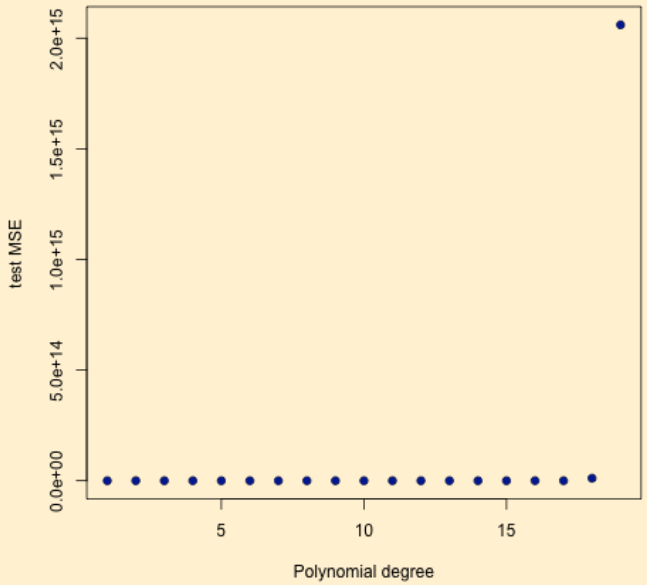
Training MSE



Choose polynomial degree 19?

Training MSE – and – test MSE



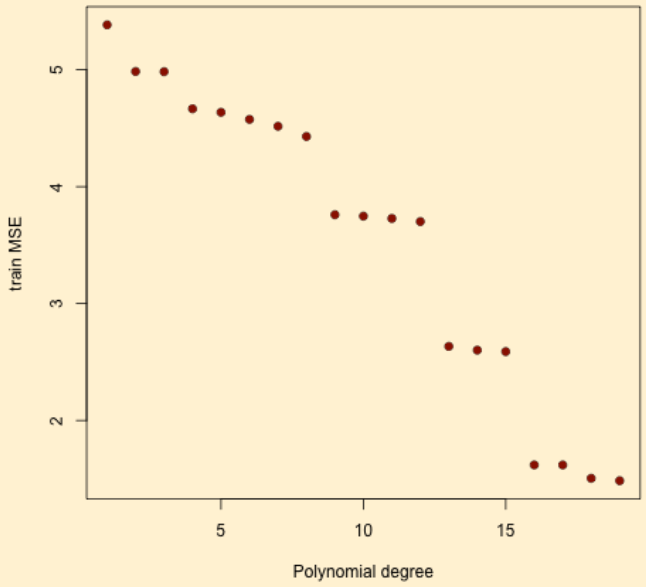


Choose polynomial degree 19?

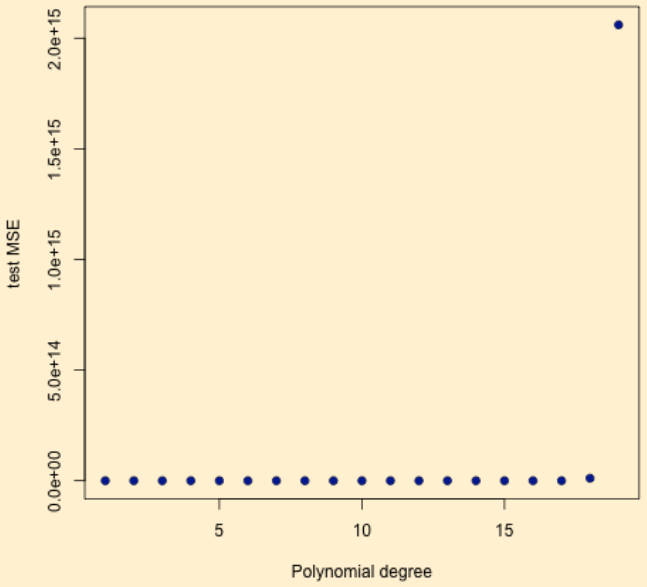
Degree 19 does really bad in test set – overfitting in training

Can we then try again? NO!

Training MSE – and – test MSE

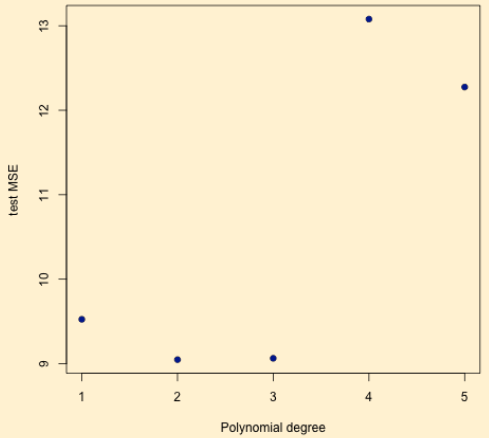


Choose polynomial degree 19?



Degree 19 does really bad in test set – overfitting in training

Can we then try again? NO!



Test error would pick degree 2

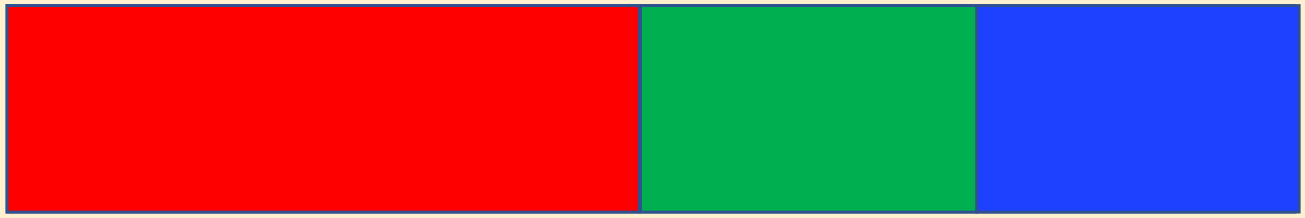
But we can't use the test error based on degree 2 because we used the test data to select the model: model could be overfitted with respect to degree (or whatever parameter your optimization method uses)

Solution train/validate/test

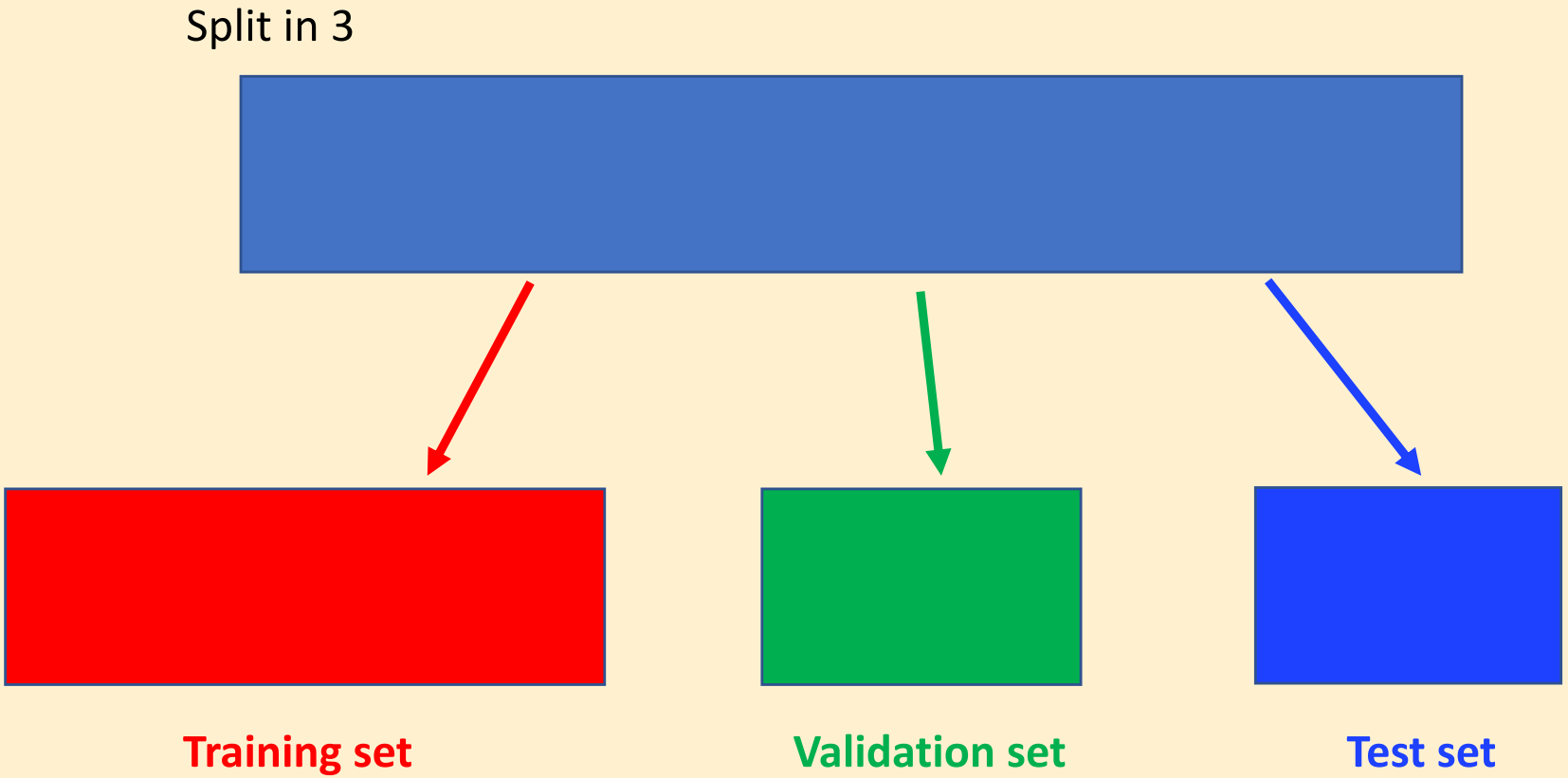
Full dataset

Solution train/validate/test

Split in 3

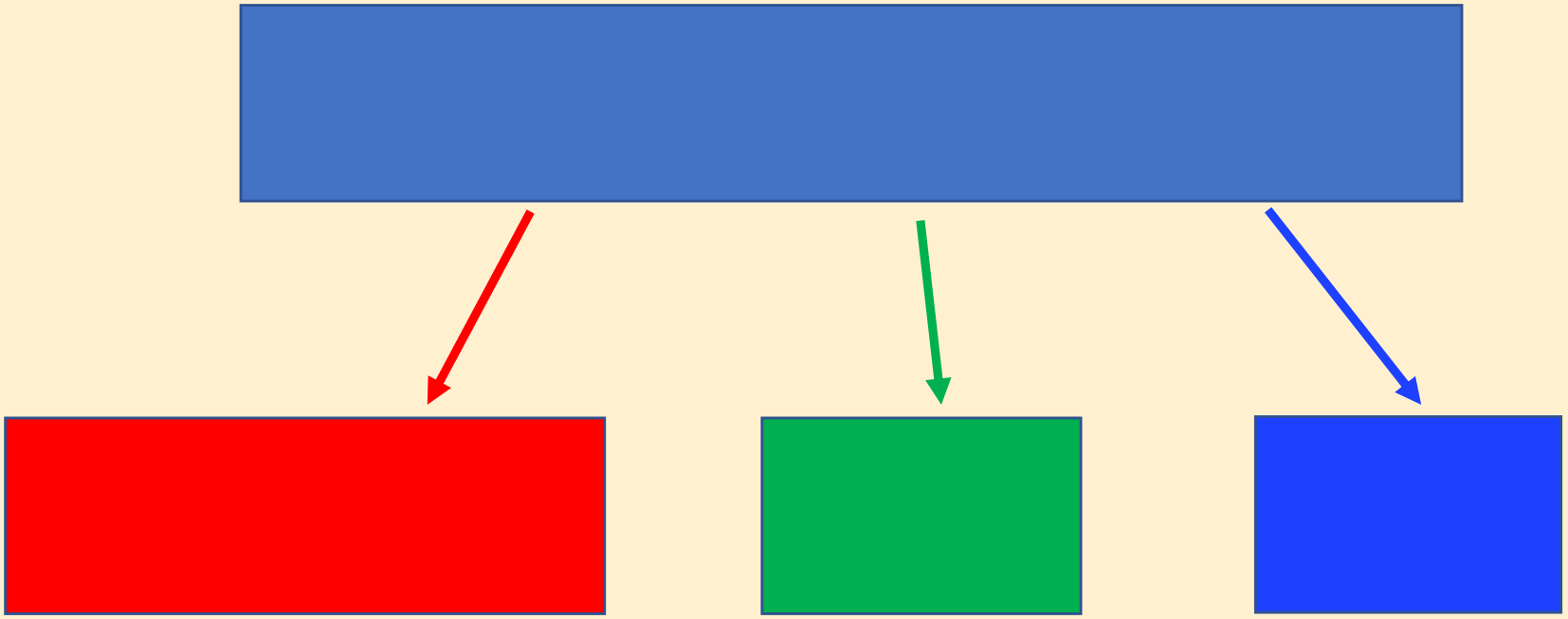


Solution train/validate/test



Solution train/validate/test

Split in 3



Training set

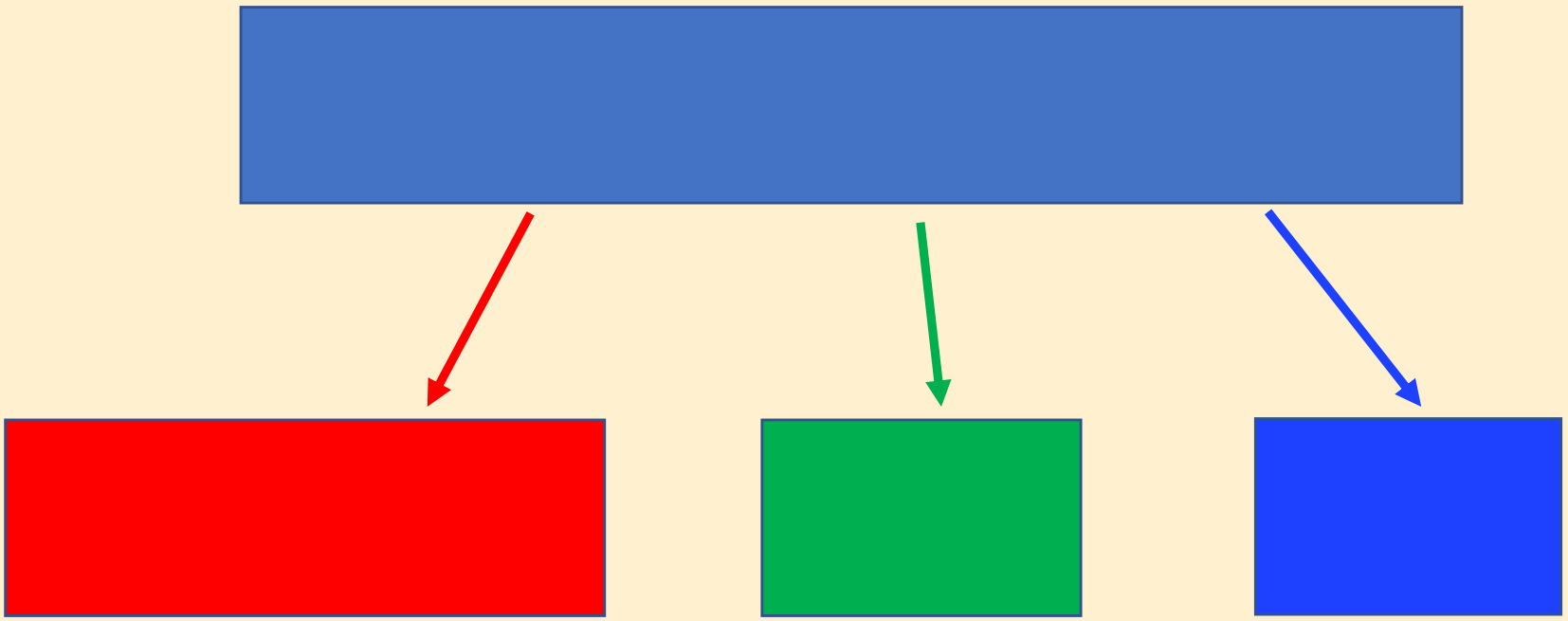
Fit all models here with all values of tuning parameters

Validation set

Test set

Solution train/validate/test

Split in 3



Training set

Fit all models here with all values of tuning parameters

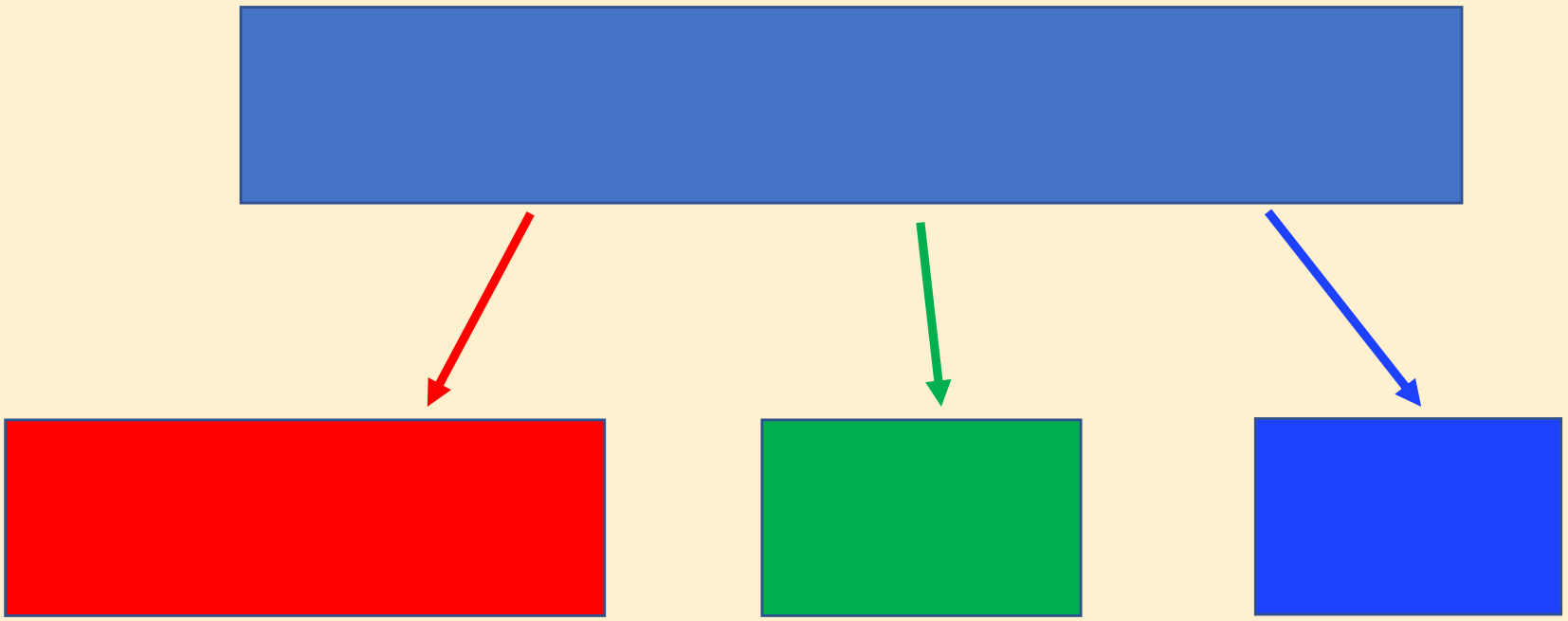
Validation set

Choose the best model here among the different values of tuning parameters

Test set

Solution train/validate/test

Split in 3



Training set

Fit all models here with all values of tuning parameters

Validation set

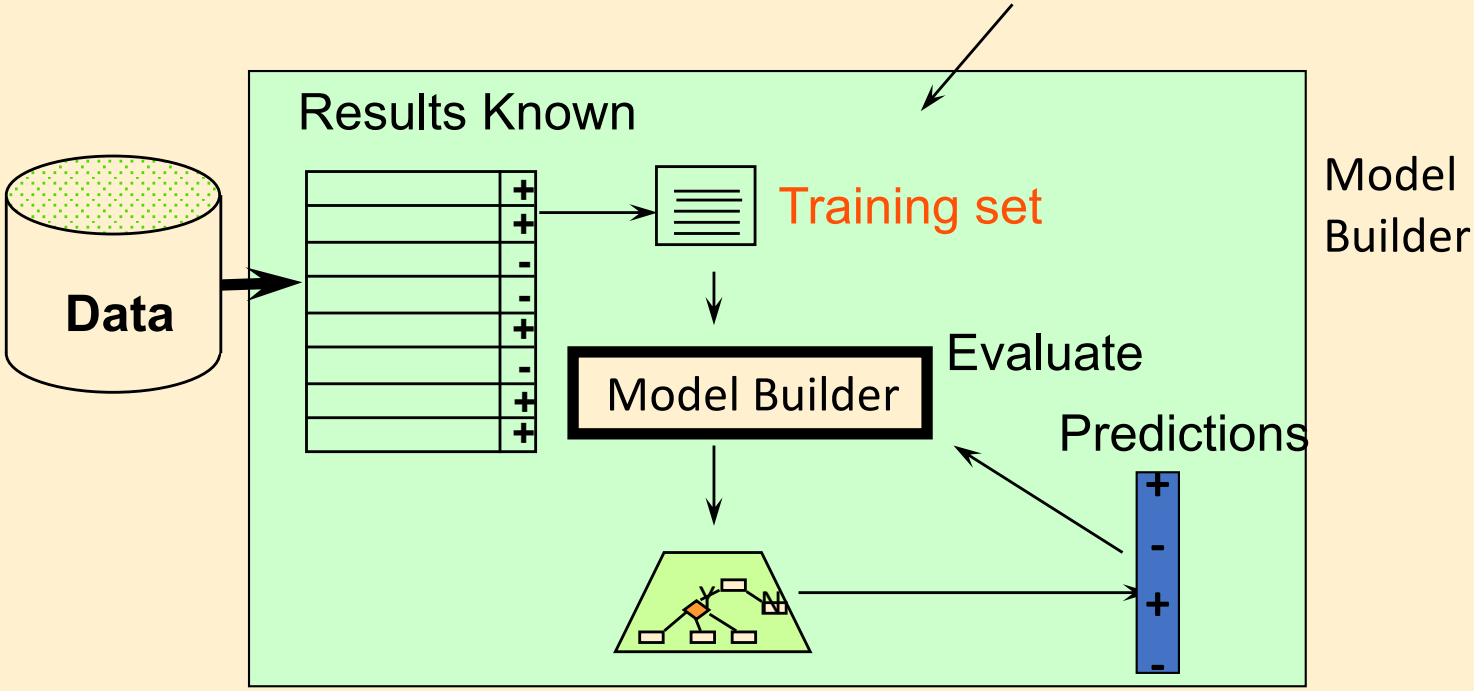
Choose the best model here among the different values of tuning parameters

Test set

Evaluate how well your chosen model performed here

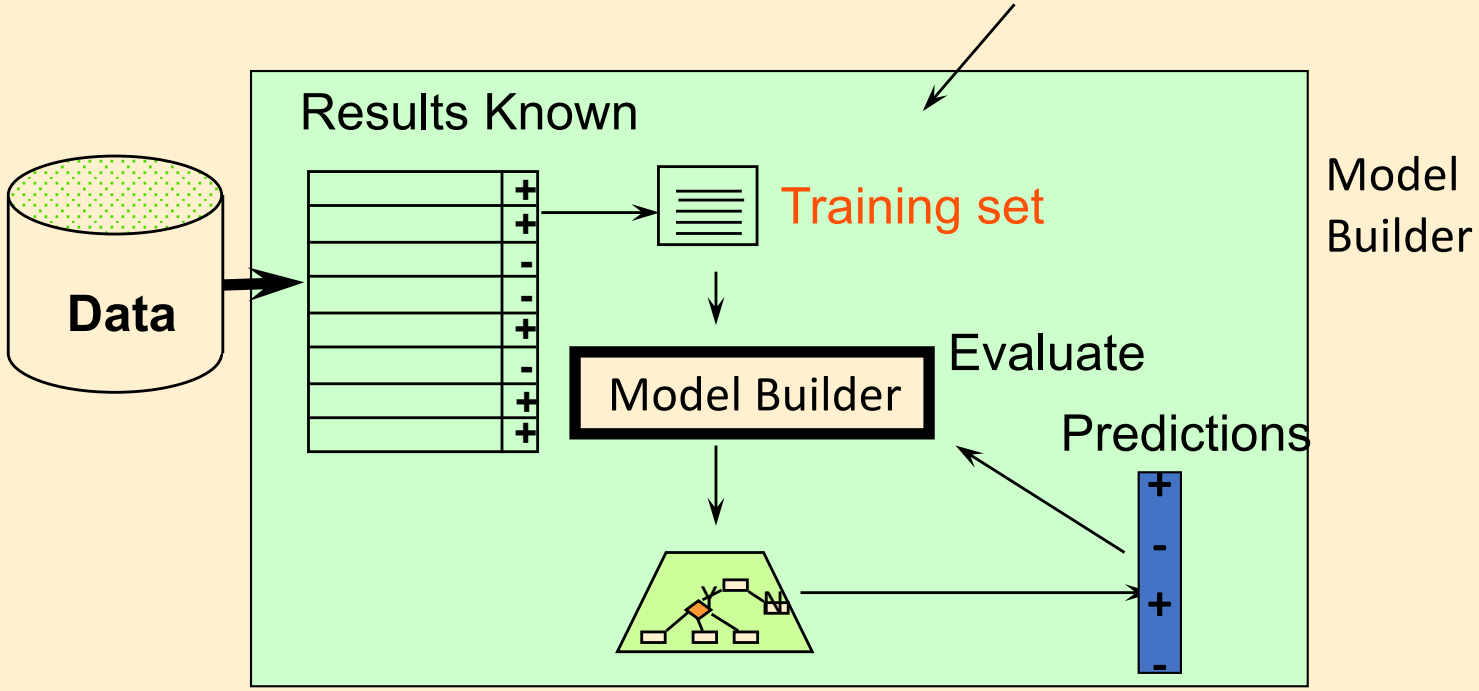
Train/validate/test

Do this green box for multiple values of the procedures parameter(s), e.g. d = 3, 4, 5, 6 for polynomial degree



Train/validate/test

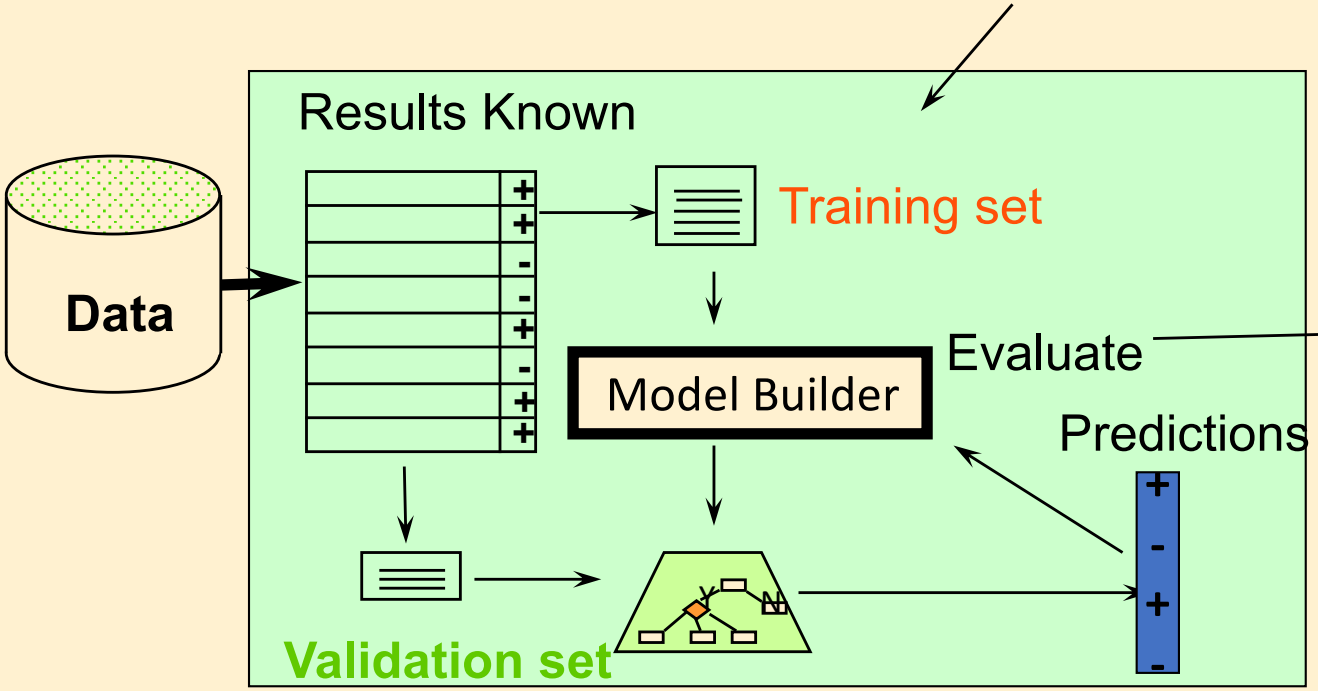
Do this green box for multiple values of the procedures parameter(s), e.g. d = 3, 4, 5, 6 for polynomial degree



Each model based on training set has potentially been over-fitted… so utilize a validation set to determine best fitting model

Train/validate/test

Do this green box for multiple values of the procedures parameter(s), e.g. d = 3, 4, 5, 6 for polynomial degree

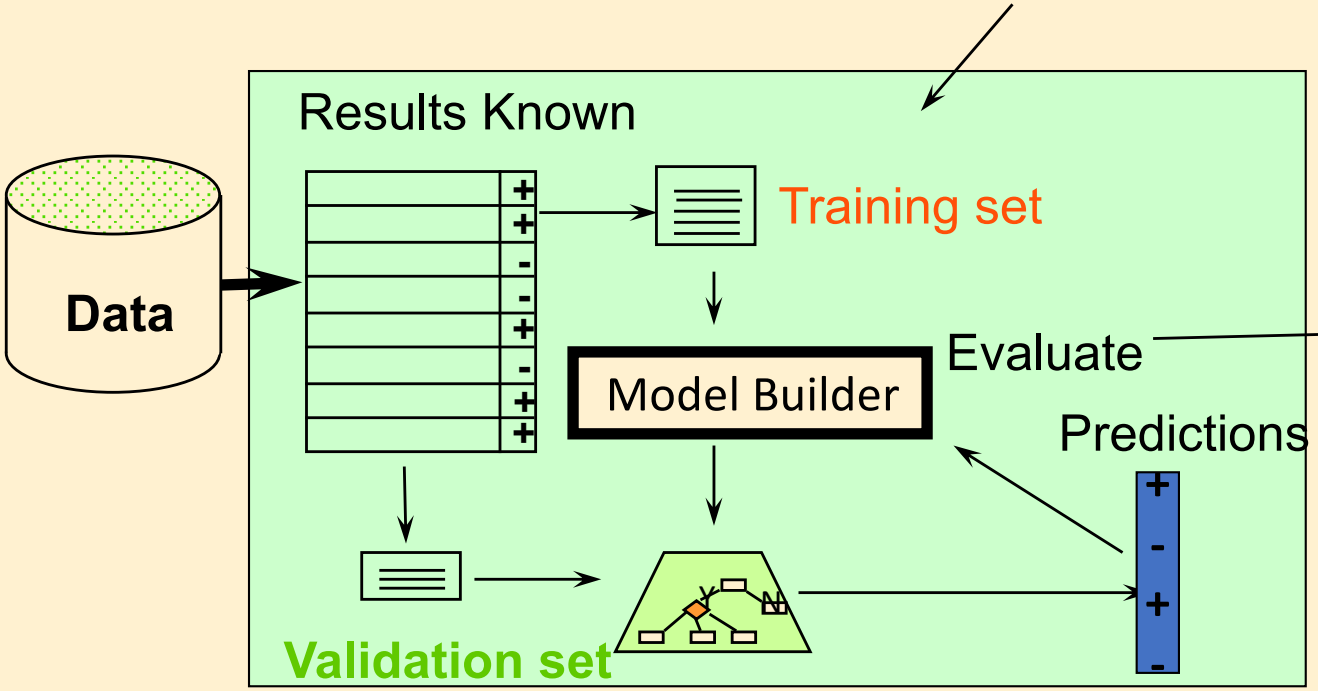


Model Builder

Choose best parameter value based on **validation set** results – evaluate model for each parameter value and choose the best

Train/validate/test

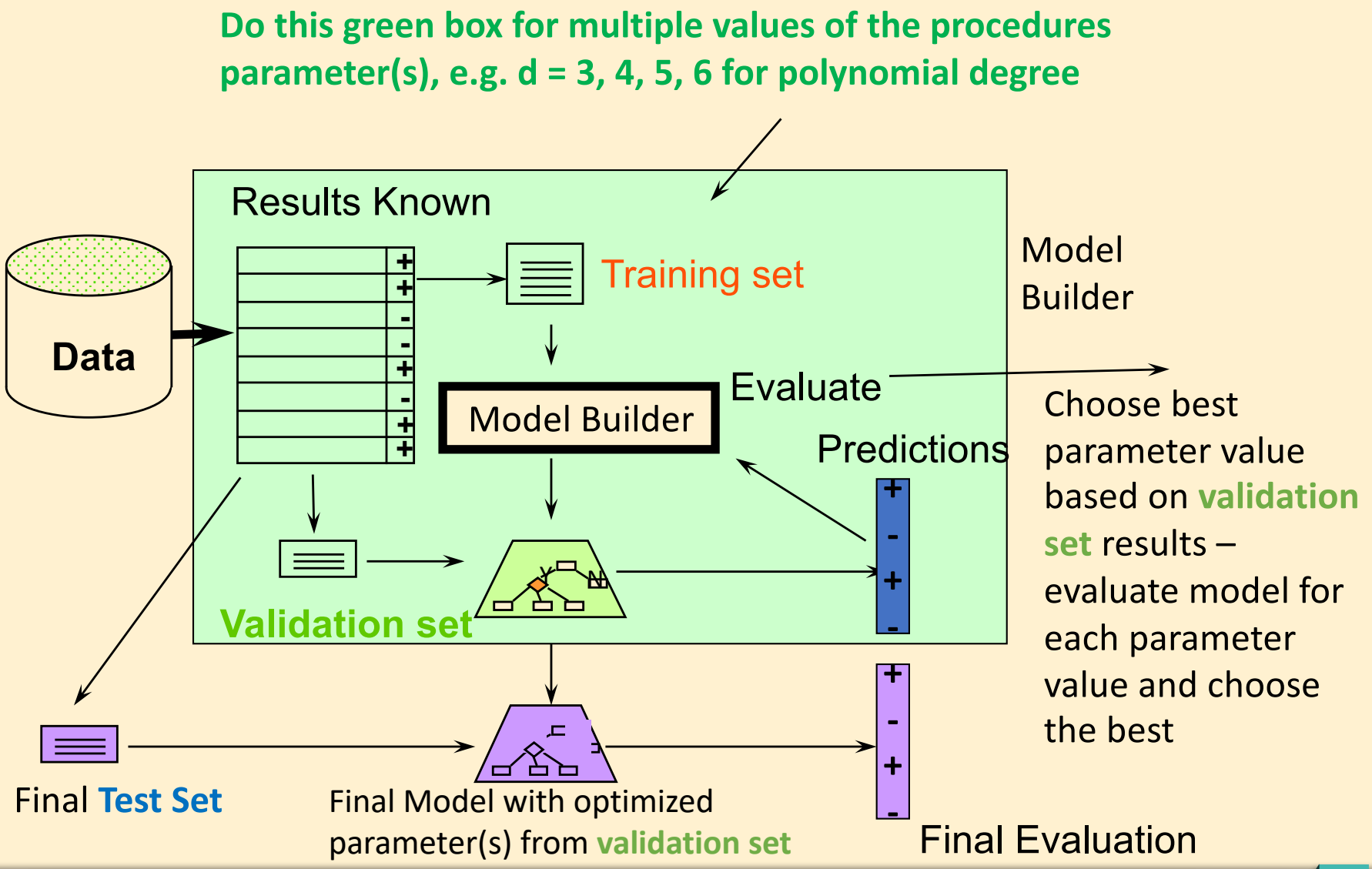
Do this green box for multiple values of the procedures parameter(s), e.g. d = 3, 4, 5, 6 for polynomial degree



Choose best parameter value based on validation set results – evaluate model for each parameter value and choose the best

Because we are choosing from many models we have again potentially over-fitted and the classification accuracy may be optimistic -- so utilize a test set to determine accuracy of best fitting model

Train/validate/test

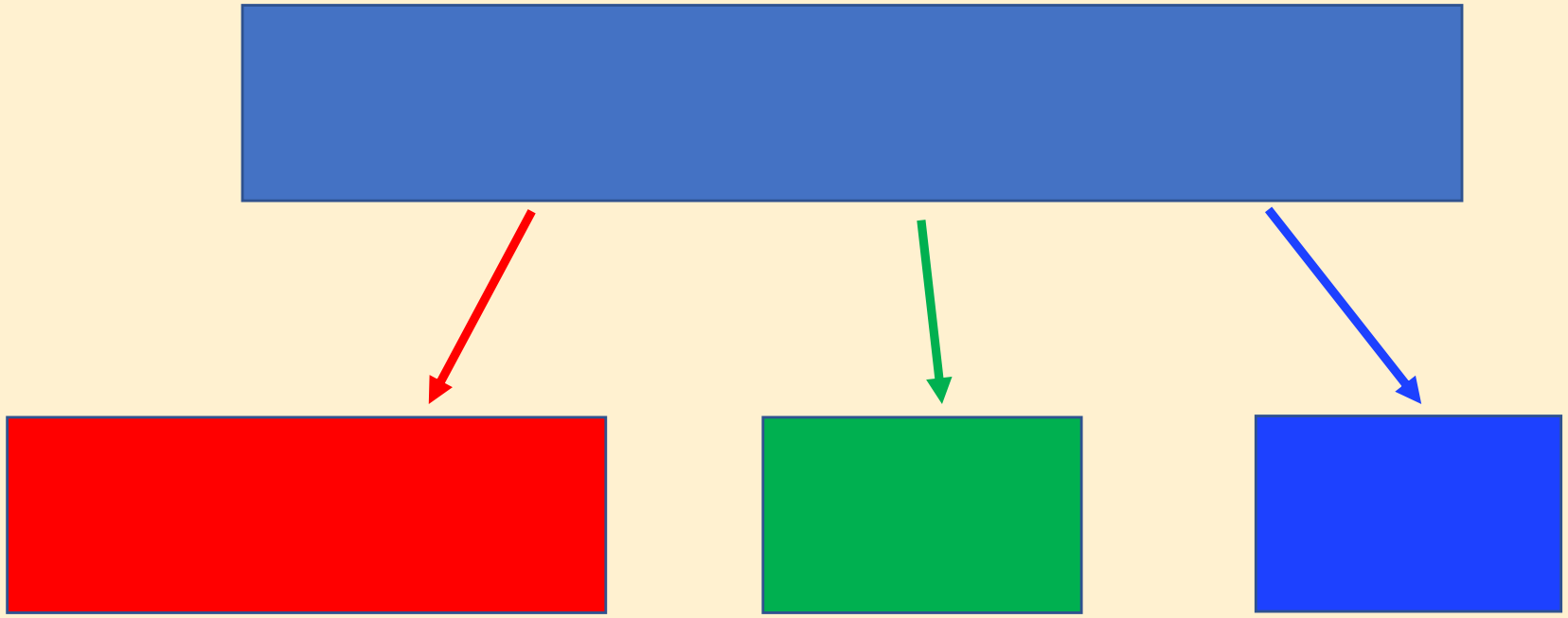


Train/validate/test

- But this is expensive in terms of data
- We need to reserve a good part of the data for each of validation and testing
- Can we use cross-validation again to help us not be so wasteful?
- Yes, but we have to be careful…

Incorporating cross-validation

Split in 3



Training set

Fit all models here with all values of tuning parameters

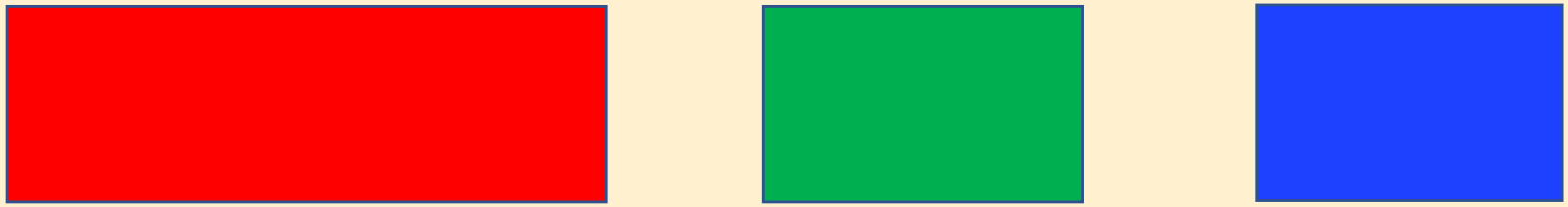
Validation set

Choose the best model here among the different values of tuning parameters

Test set

Evaluate how well your chosen model performed here

Incorporating cross-validation



Training set

Fit all models here with all values of tuning parameters

Validation set

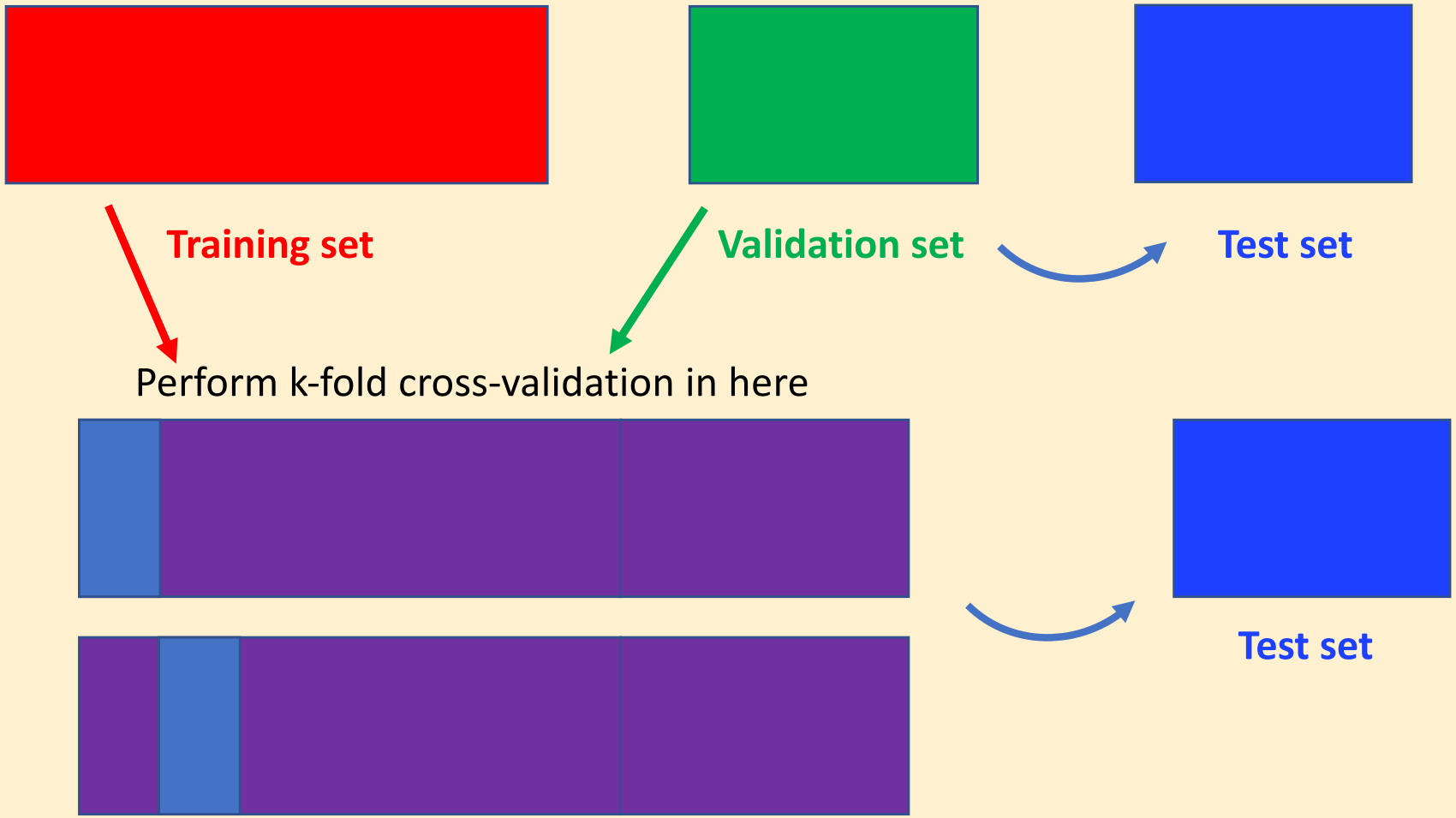
Choose the best model here among the different values of tuning parameters

Test set

Evaluate how well your chosen model performed here

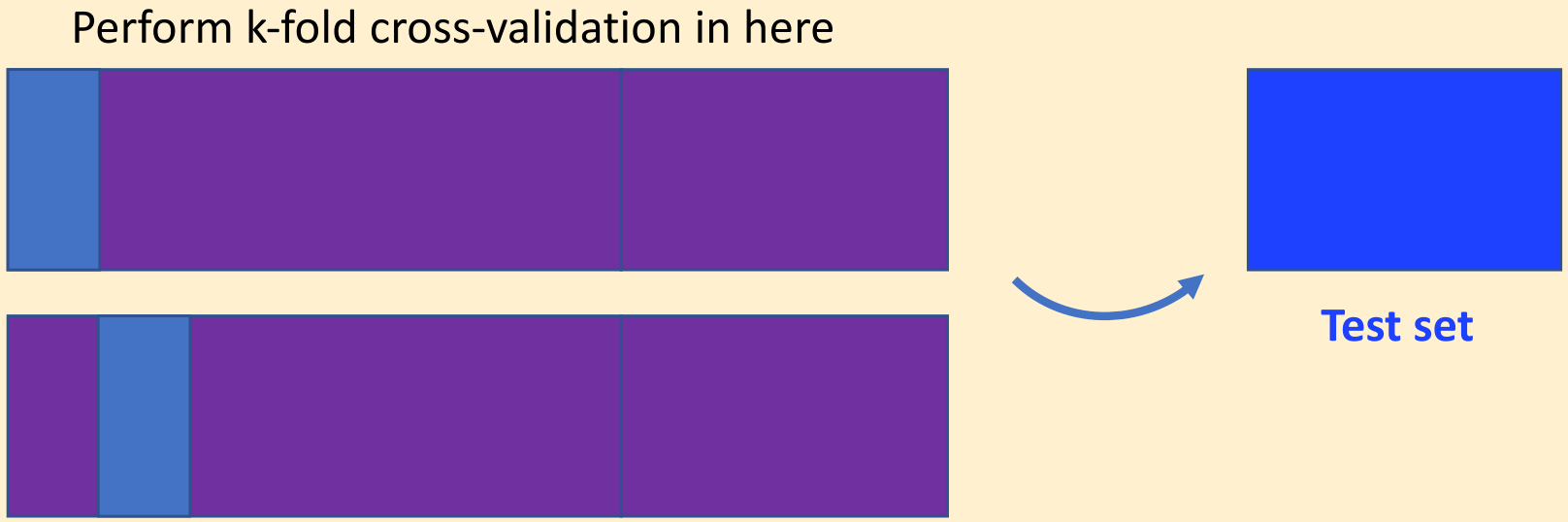
Step 1: break down the problem. Let's combine the training and validation into a single set that we will cross-validate over to get parameter estimates…

Incorporating cross-validation



Gives k estimates of parameters – take the average and refit to the combined training/validation set – note that this would not pick d = 19 for the polynomial -- why not?

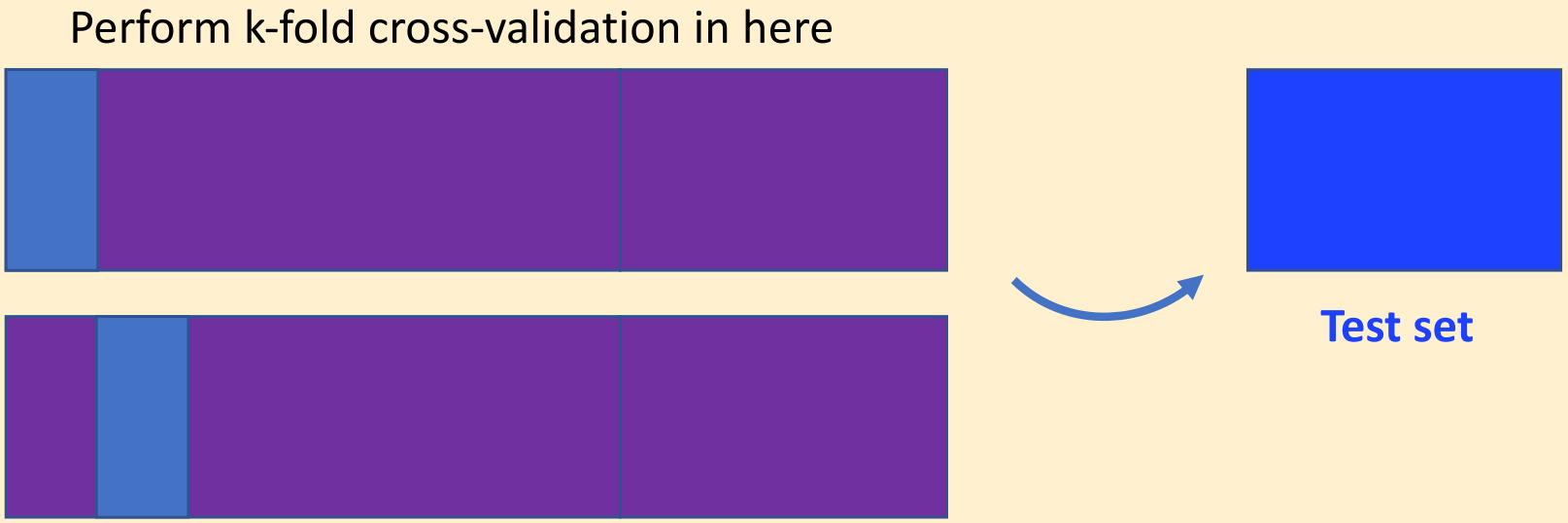
Incorporating cross-validation



Gives k estimates of parameters – take the average (or most common) and refit to the combined training/validation set – note that this would not pick d = 19 for the polynomial -- why not?

This approach is commonly used in medical research where the Test Set ideally comprises of a completely independent dataset – perhaps from another institution

Incorporating cross-validation



Gives k estimates of parameters – take the average and refit to the combined training/validation set – note that this would not pick d = 19 for the polynomial -- why not?

This approach is commonly used in medical research where the Test Set ideally comprises of a completely independent dataset – perhaps from another institution

But what about if we really have a single data set that is not particularly big. Can we avoid reserving the test dataset for a single use? The answer is yes, but it takes some computer muscle. We need to perform an outer cross-validation around an inner cross-validation – *nested cross-validation*

Nested cross-validation

Full data

Split in k-folds

Nested cross-validation

Full data

Split in k-folds

Set the first test-fold

Nested cross-validation

Full data

Split in k-folds

Set the first test-fold

Use the k-1 training folds to fit the model and parameters

Nested cross-validation

Full data

Split in k-folds

Set the first test-fold

Use the k-1 training folds to fit the model and parameters with m-fold cross-validation

Nested cross-validation

Full data

Split in k-folds

Set the first test-fold

Use the k-1 training folds to fit the model and parameters with m-fold cross-validation

Nested cross-validation

Full data

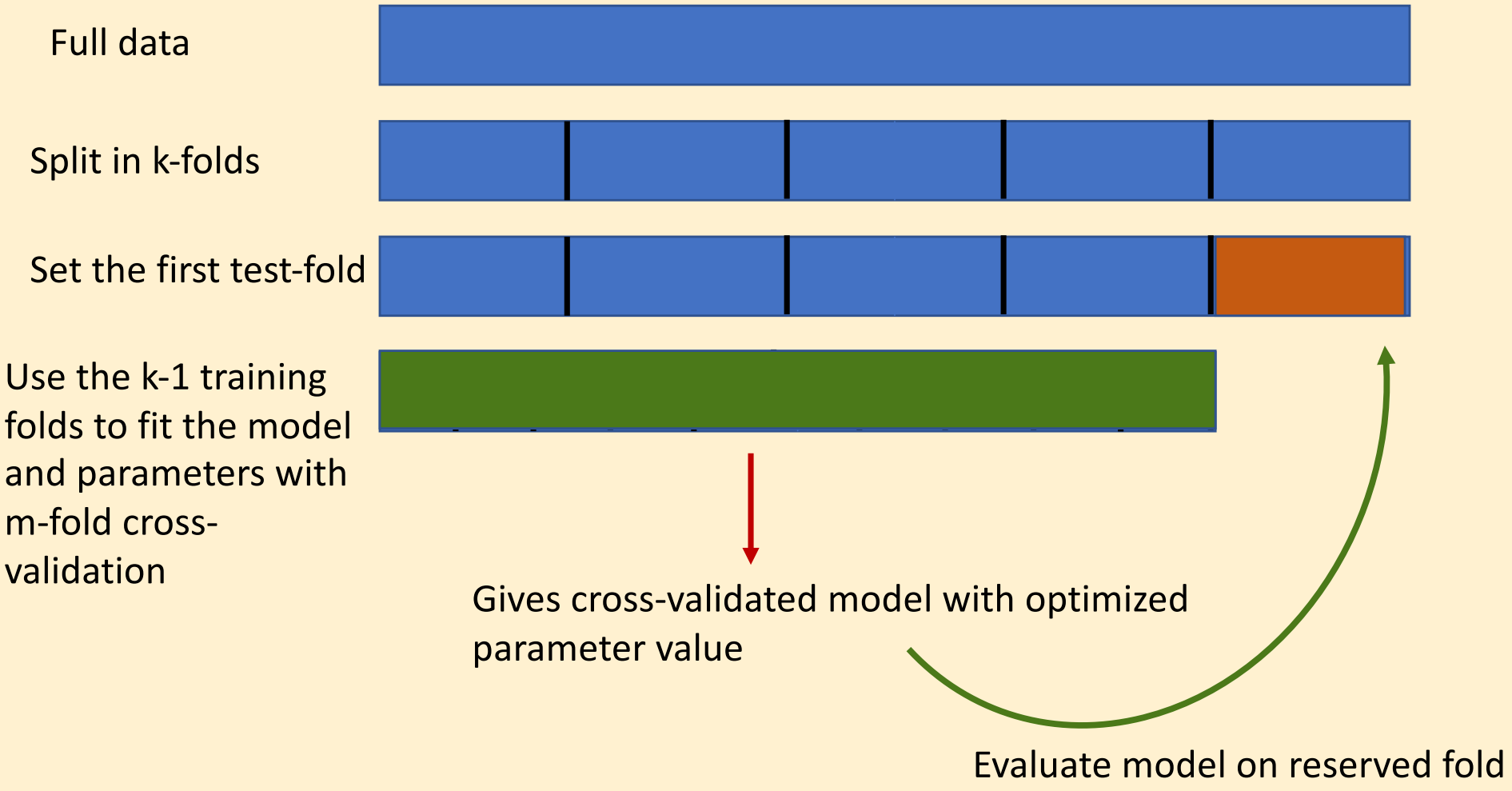
Split in k-folds

Set the first test-fold

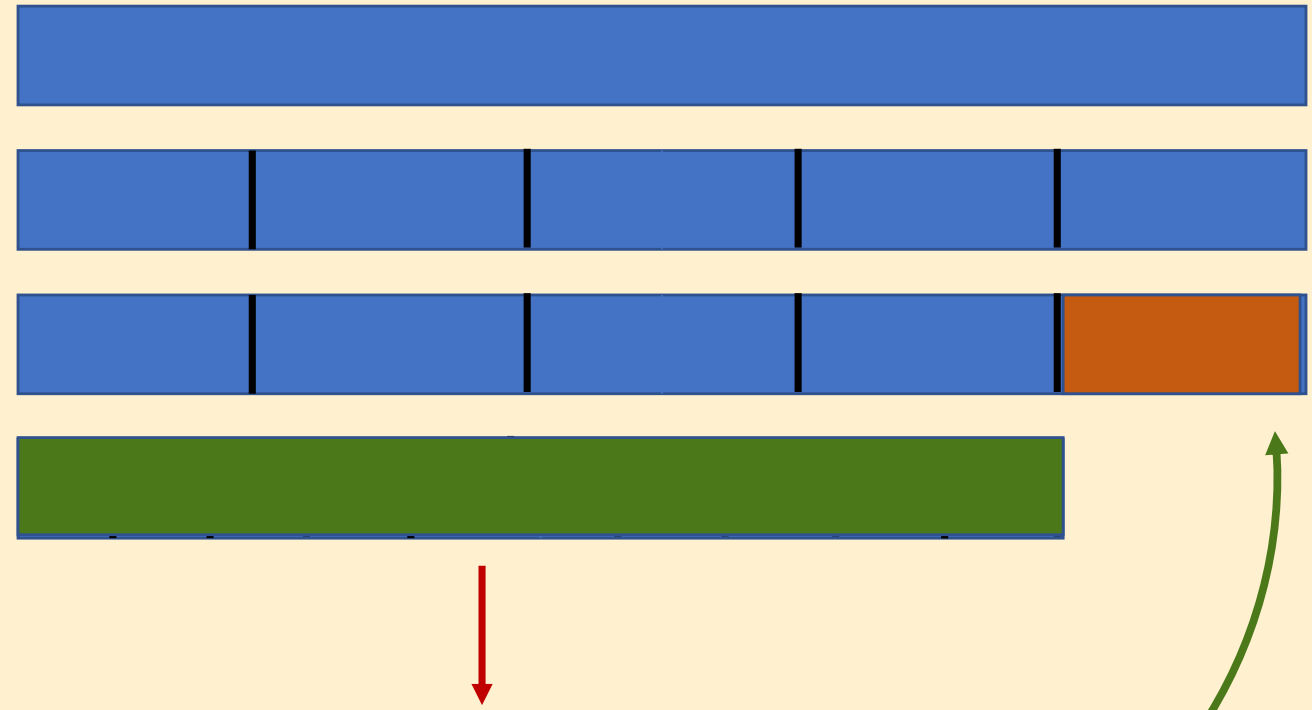
Use the k-1 training folds to fit the model and parameters with m-fold cross-validation

Nested cross-validation

Nested cross-validation

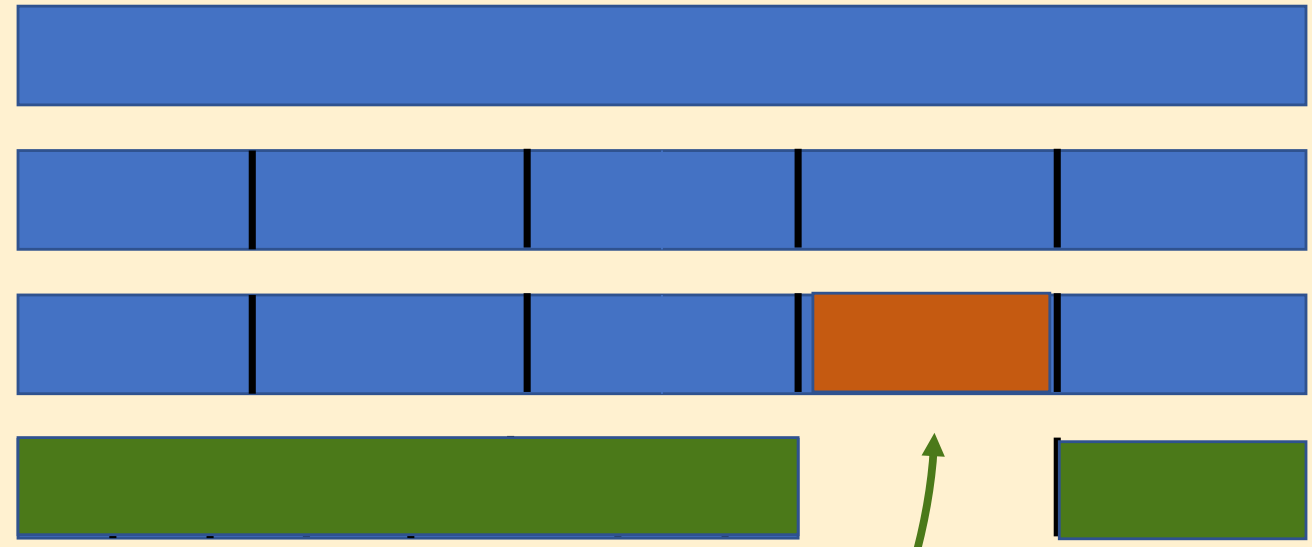


Nested cross-validation



Repeat through all k-folds such that within each of the k-folds, m-fold cross-validation is performed on the remaining k-1 training folds

Nested cross-validation



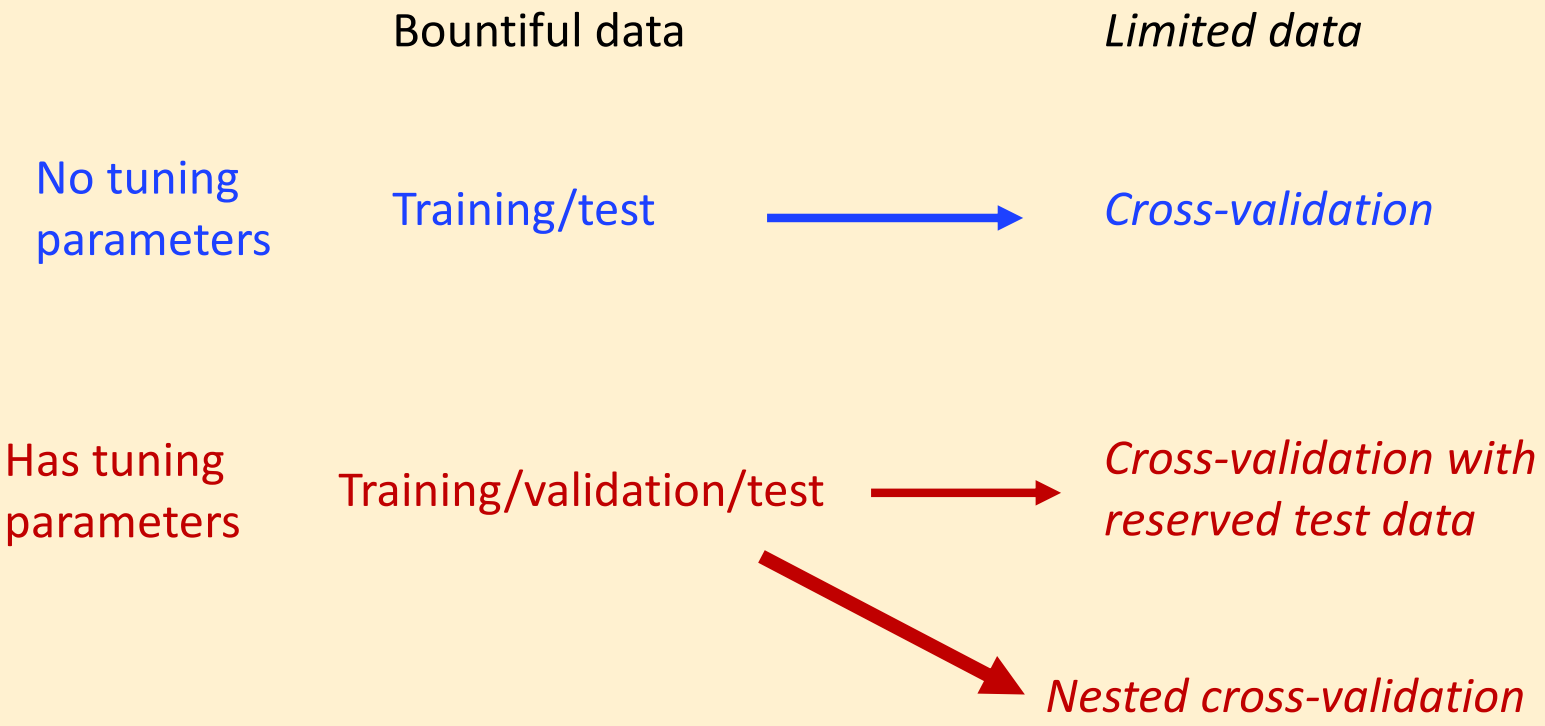
Use the k-1 training folds to fit the model and parameters with m-fold cross-validation

Gives estimated cross-validated model

Repeat through all k-folds such that within each of the k-folds, m-fold cross-validation is performed on the remaining k-1 training folds

Evaluate model accuracy on reserved fold

Summary


Bountiful data
Limited data
No tuning parameters
Training/test
Cross-validation
Has tuning parameters
Training/validation/test
Cross-validation with reserved test data
Nested cross-validation