Data Mining: Classification: Model Evaluation

CS4821 – CS5831

Laura Brown

Some slides adapted from: A. Moore, E. Alpaydin, G. Piatetsky-Shapiro; Han, Kamber, & Pei; C.F. Aliferis; S. Russell; D. Klein; L. Kaebling; A. Mueller; P. Smyth; C. Volinsky; Tan, Steinbach, & Kumar; J. Taylor; G. Dong;

Model Evaluation

- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

Classification Process

- Given a collection of records (training set)

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \text{ where}$$

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$$

- Each record contains a vector of attributes, and a class label, $y \in \mathcal{Y}$
- Use the data, $\mathcal{D}$, to find a model for the class label as a function of the attributes

$$\hat{f}(\boldsymbol{x}) \colon \mathbb{R}^p \longmapsto \mathcal{Y}$$

- Use the model, $\hat{f}$, to predict class for new data

$$\hat{y} = \hat{f}(\boldsymbol{x_n})$$

Generalization and Overfitting

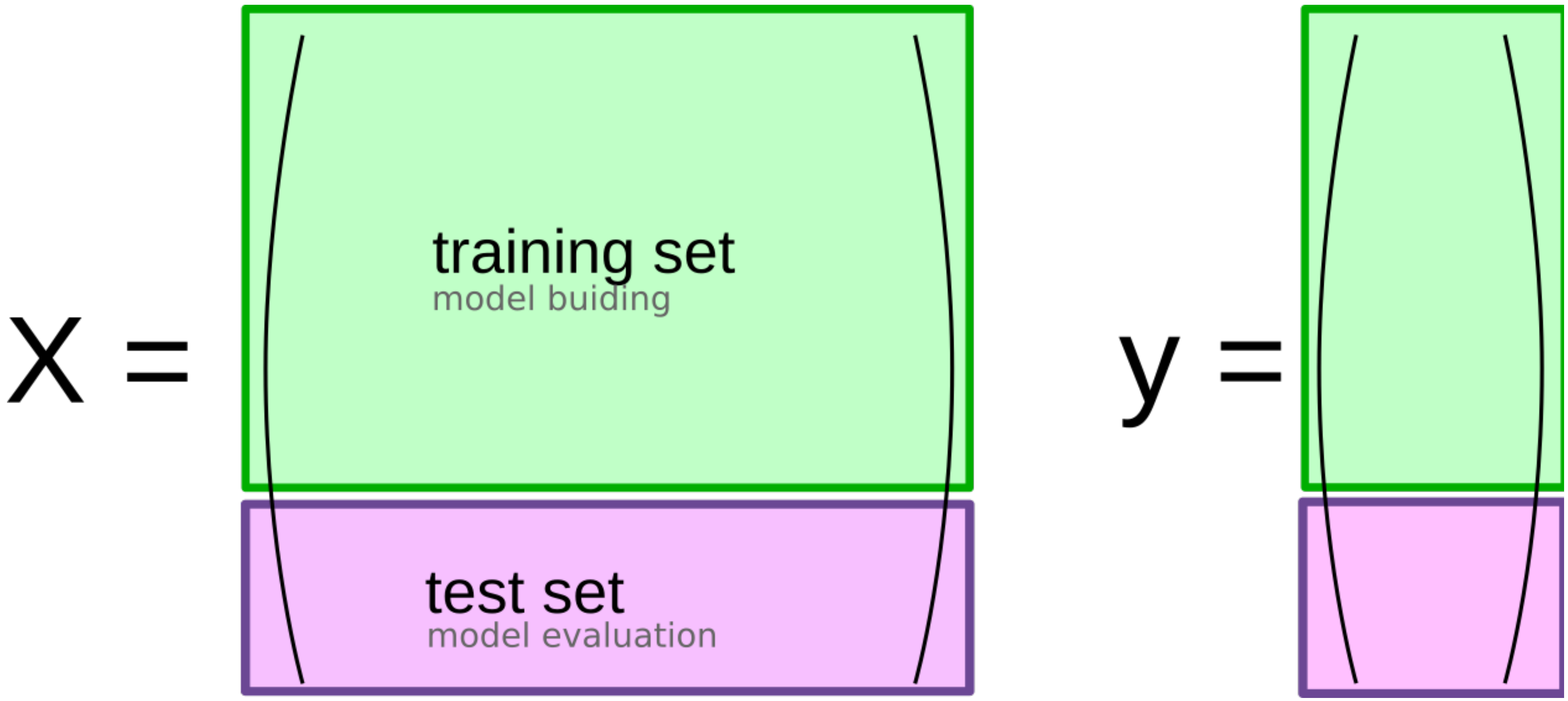
- A model should learn something about the data beyond the specific examples it has been presented (training data, the data to build a model)

- The model should be able to predict the correct output for a new samples (not only previously seen examples) – this is the property of generalization

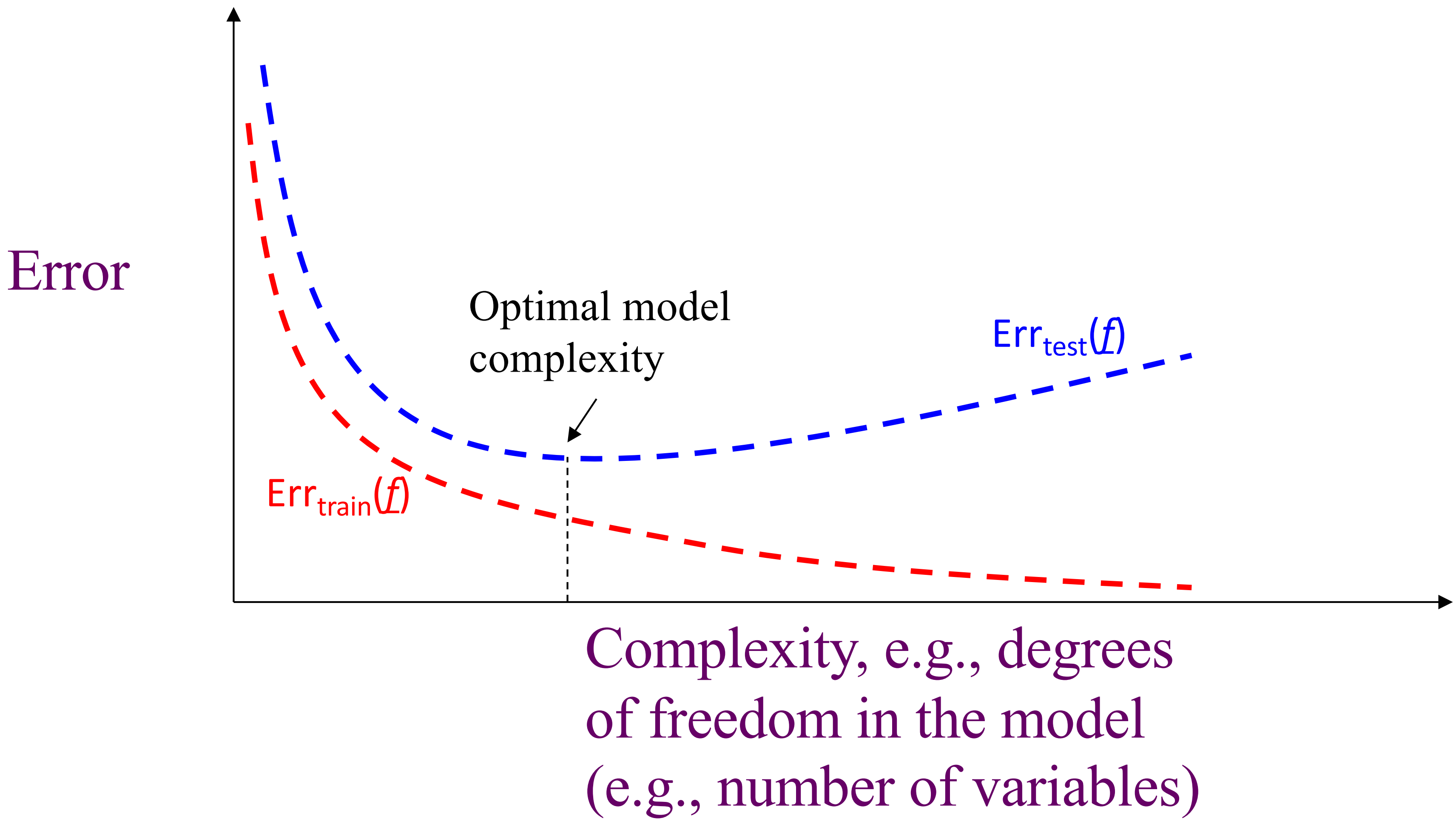
- Overfitting: Model is too complex and matches training data well, but not on new data

- Underfitting: Model is too simple and performs poorly overall

So Far: Train-Test Split



Complexity and Generalization



Conditions for Generalization

1. Information for predictions needs to be encoded in the training data

2. The training data should be large and varied to capture the variability in the underlying process/distribution

3. The new data run through the model should be generated from the same process as the training data

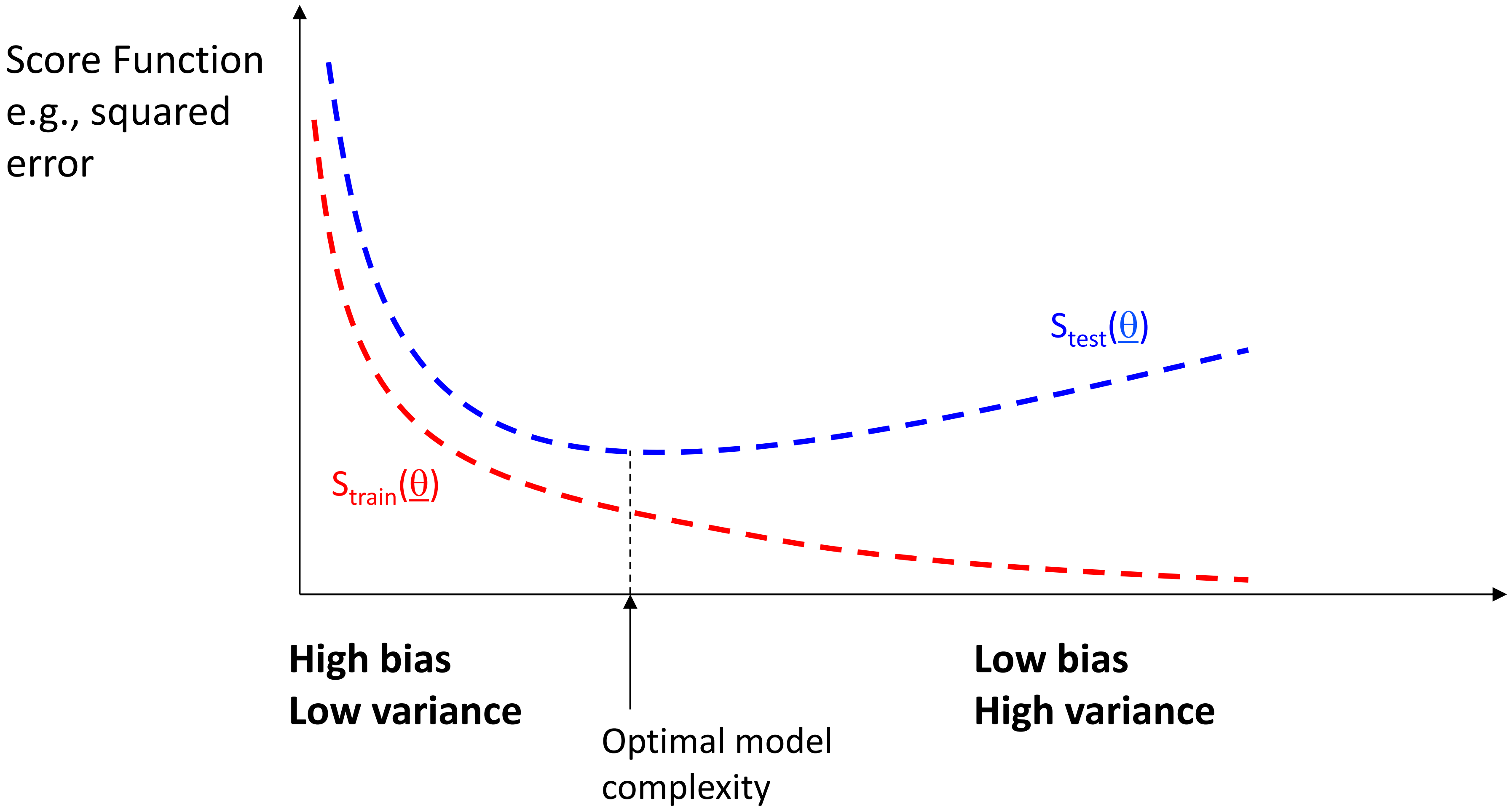
Bias – Variance Decomposition

- The generalization error can be broken down into two parts:

  Gen. Error as MSE = Variance + Bias^2

- Bias: error from the difference between the model's predictions and the true targets
  - High bias -> model with few parameters
  - Low bias -> model with many parameters
- Variance: measure of spread, how much does the estimator vary with a new data set?
  - High variance -> model with many parameters
  - Low variance -> model with few parameters

Complexity and Generalization

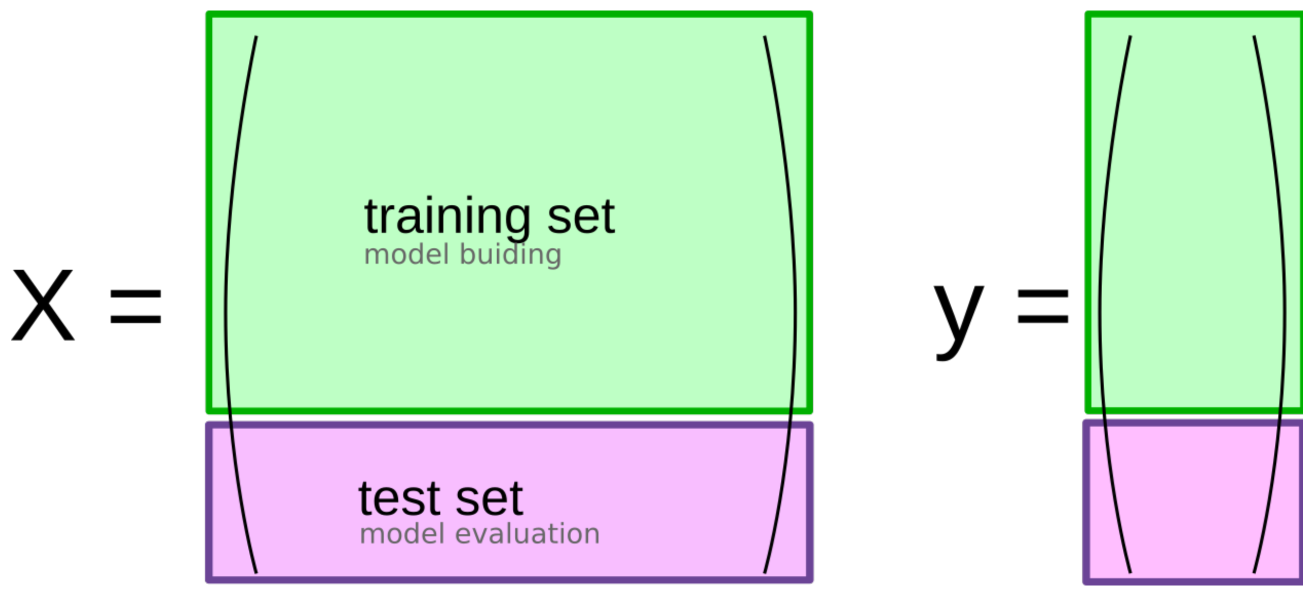


Generalization and Overfitting

- How to avoid overfitting?
  - Use analysis methods that intrinsically generalize well
  - Pursue simple models / classifiers for small samples
  - Fit parameters in data separate from data used to estimate generalization error
  - Add a penalty term that corrects for optimism
    - regularization

Data Partitioning – Holdout Set

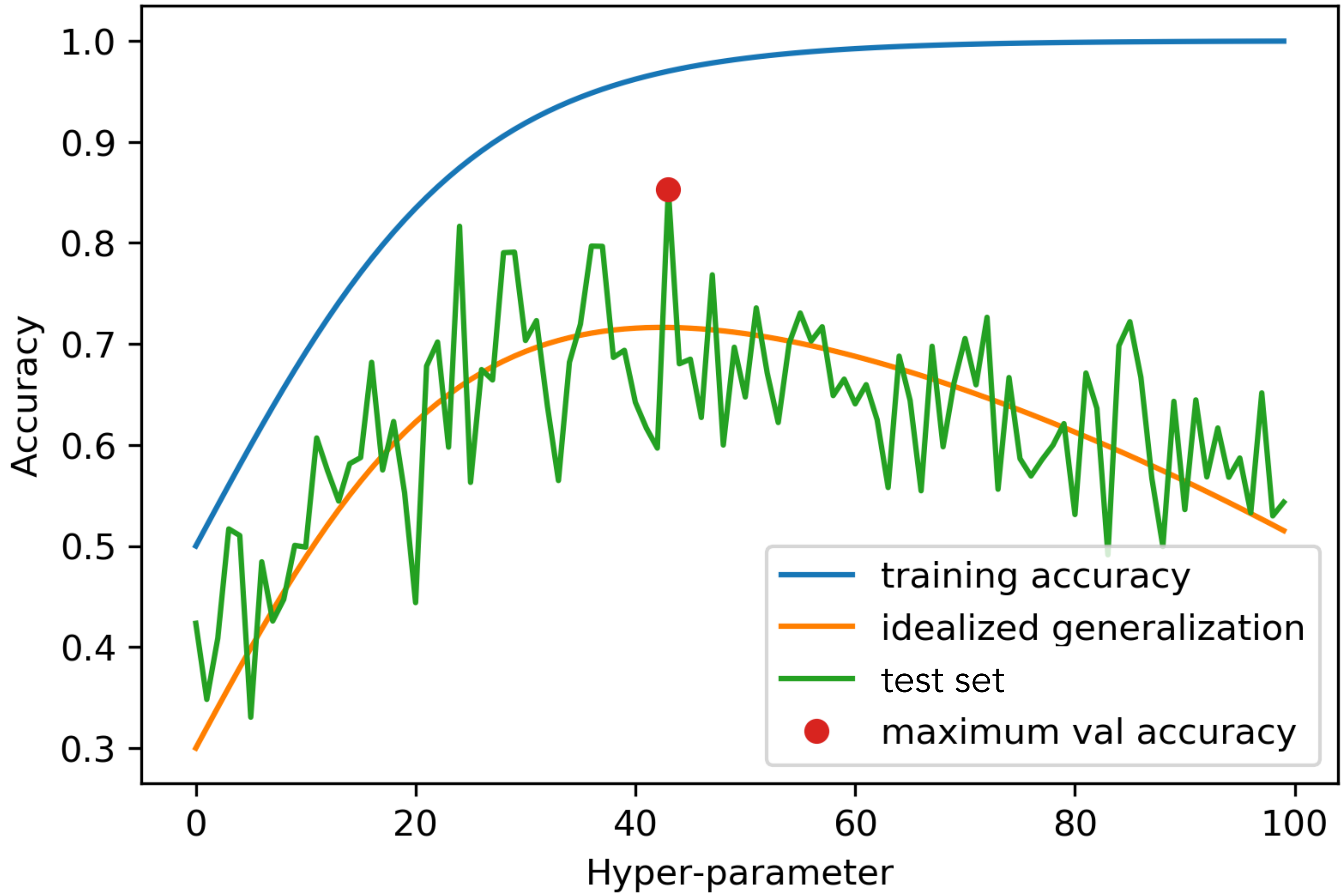
- randomly split data into training and test sets
  - Many splits can be used: 60/40, 70/30, 75/25, 80/20
  - Ensure split is **stratified** for unbalanced data sets
- build a model on *train* set
  - Find the model parameters, build a decision tree, learn the artificial neural network
- evaluate on *test* set



Limitations of Holdout Set

- Many models have hyper-parameters to select, e.g., k for KNN, or we want to choose between a KNN and DT model
- If we want to select the *best k*, learn a model for each value of k on the training set, and evaluate it on the test set
- What's the issue?

Limitations on Holdout Set

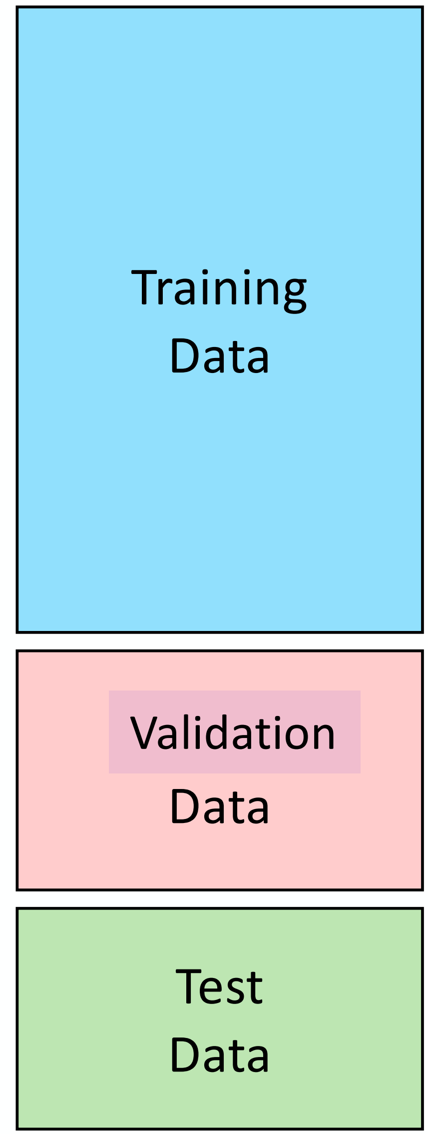


- Assumes we know “real” generalization; instead we have noisy estimate from test set

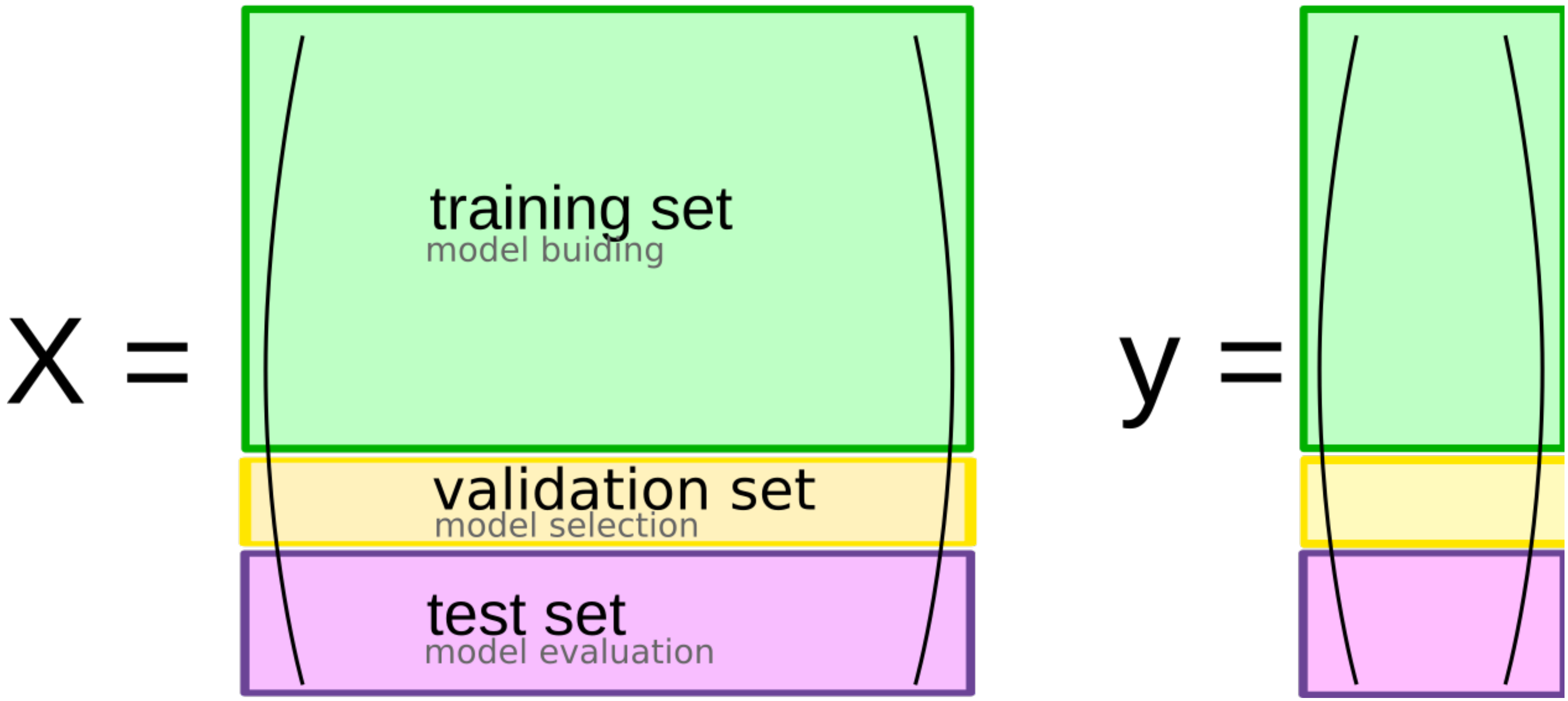
Image from Muller, AMLP

Threefold Split –Validation Set

- For many models we have:
  - Parameters – to specify a particular model
  - Hyperparameters – options for the model, learning process
- What should we learn where?
  - Learn parameters on training data
  - Tune hyperparameters on validation set
  - Estimate generalization performance on test data



Threefold Split – **Best Practice**



- Use Three Sets:
  - Training set – model building
  - Validation set – model selection
  - Test set – model evaluation

Threefold Split

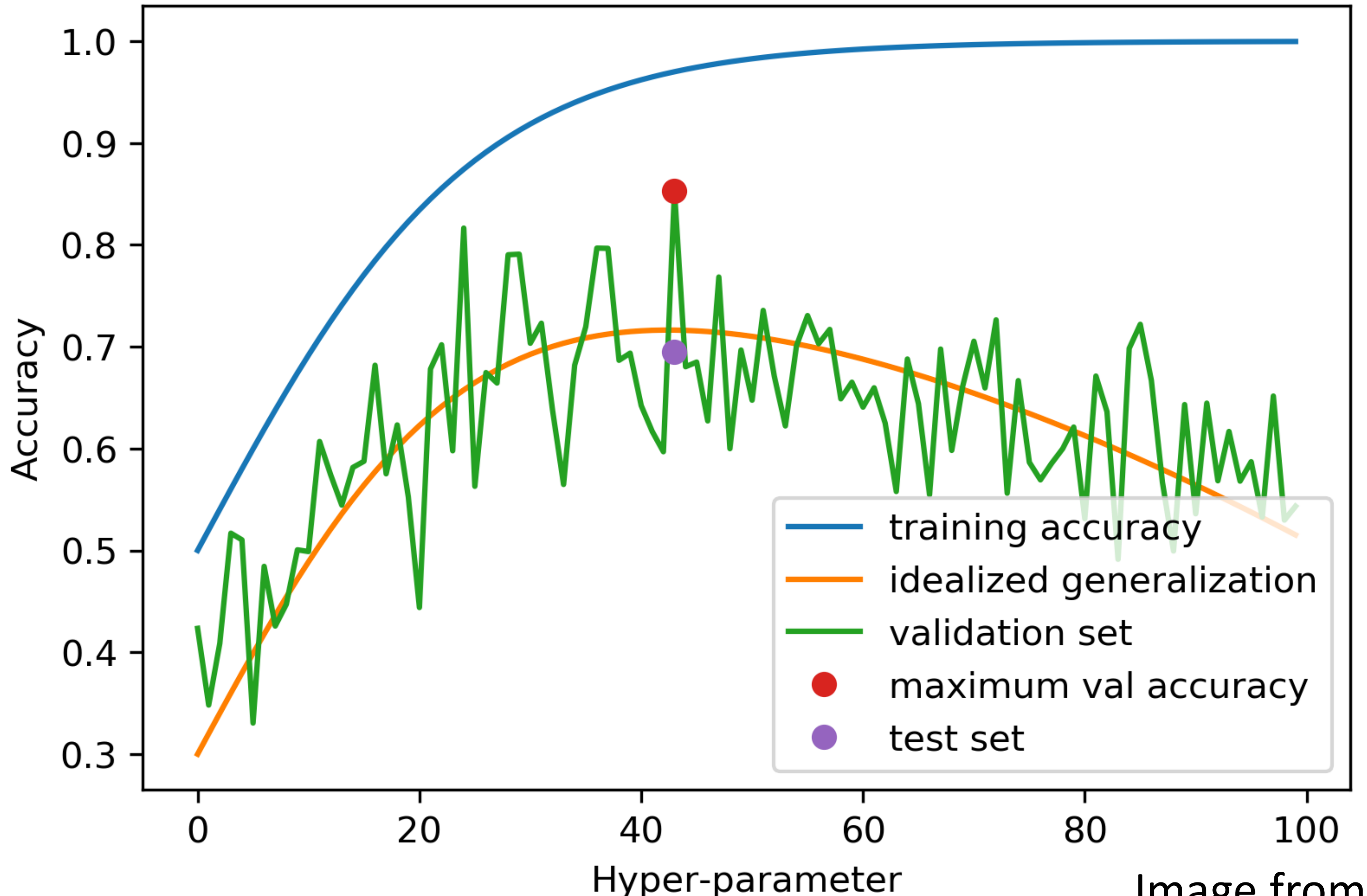


Image from Muller, AMLP

- Use validation set to select hyper-parameter; test set provides unbiased estimate of generalization performance

Only use the test set once!

Threefold Split

- Whenever evaluating more than one model, you want to use a threefold split (or a variant)
- Still aspects to improve?
  - What if we change the split (use a different random state)? We can end up with different results
  - Ideally, we want hyperparameter selection and estimate of generalization not to be impacted by the initial split
    - A large variance among data splits is a bad sign

K-fold Cross-Validation

- Cross-validation replaces a split of data with multiple different splits.
  - Splitting data into parts repeatedly
- Most common type of cross-validation is k-fold cross-validation.
  - Split data into k disjoint parts (k=5, k=10) of about equal size
- Often applied to training/validation split

K-fold Cross-Validation

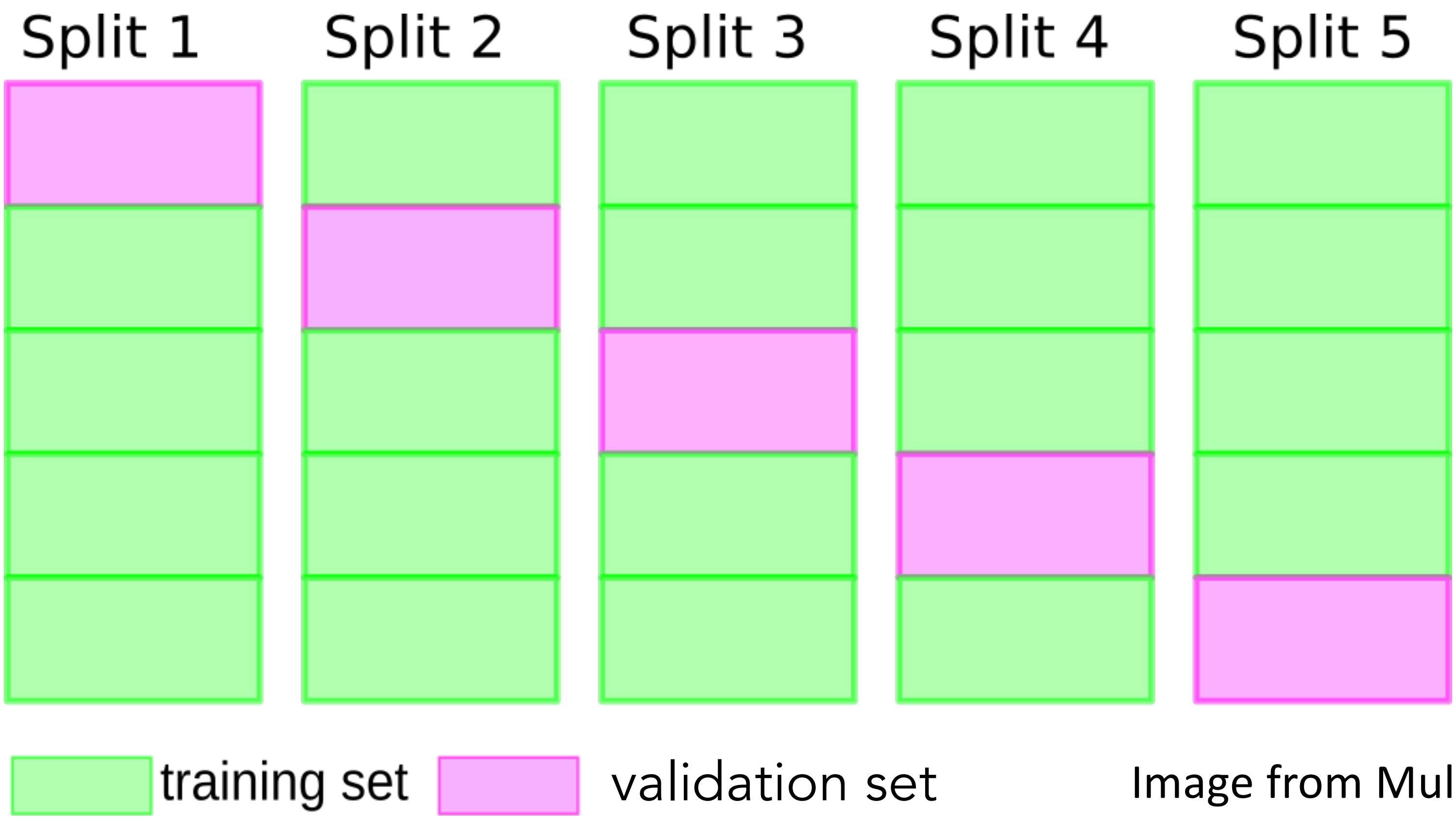


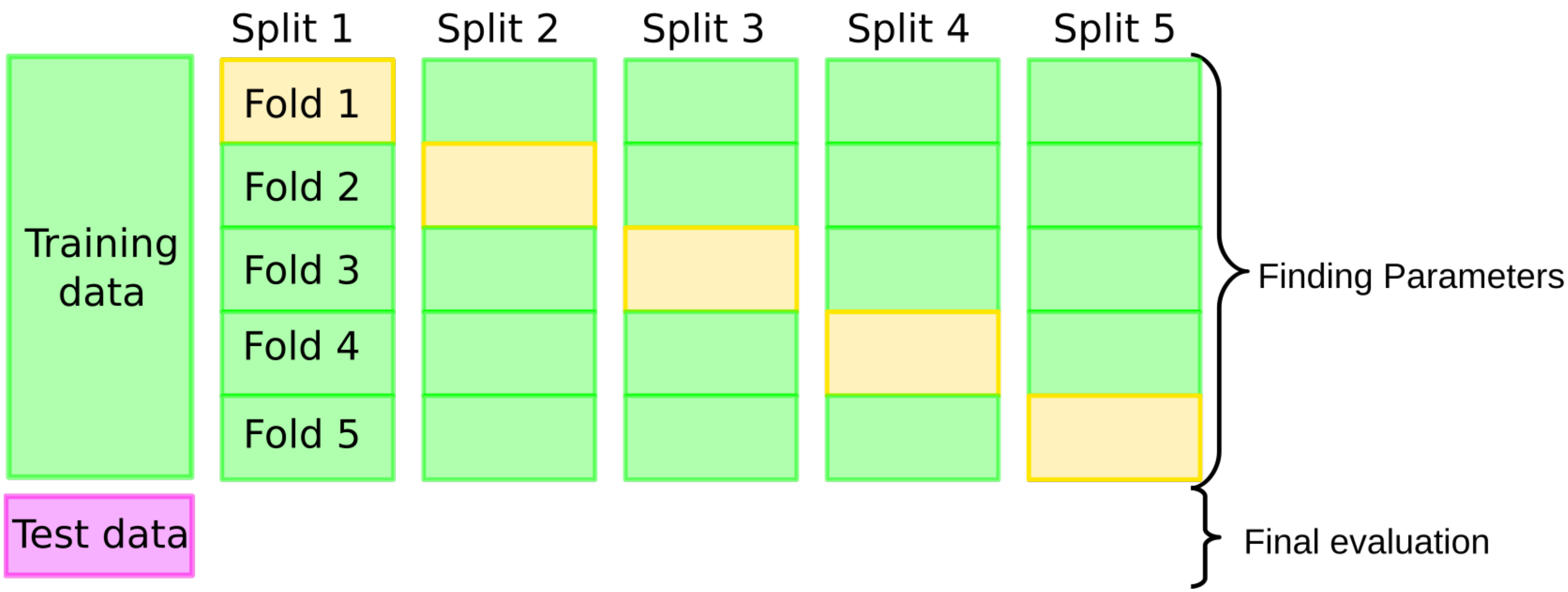
Image from Muller, AMLP

- For each split:
  - Train on training set (green)
  - Validation to evaluate the model
- Aggregate performance (mean / median)

K-Fold Cross-Validation

- Benefits:
  - Robust estimates
  - Each sample is used exactly once in a validation set
- Disadvantages:
  - Computational cost
    Building models multiple times
  - Result is not a single model, this produces *k* models

Cross-Validation + test set



- First, split into train/test sets
- Then, run cross-validation
  - For each combination of hyperparameters, train model, evaluate it on validation set
- Select best hyper-parameters on average*
- Build a new model using best hyper-parameters on full training set
- Evaluate on test set

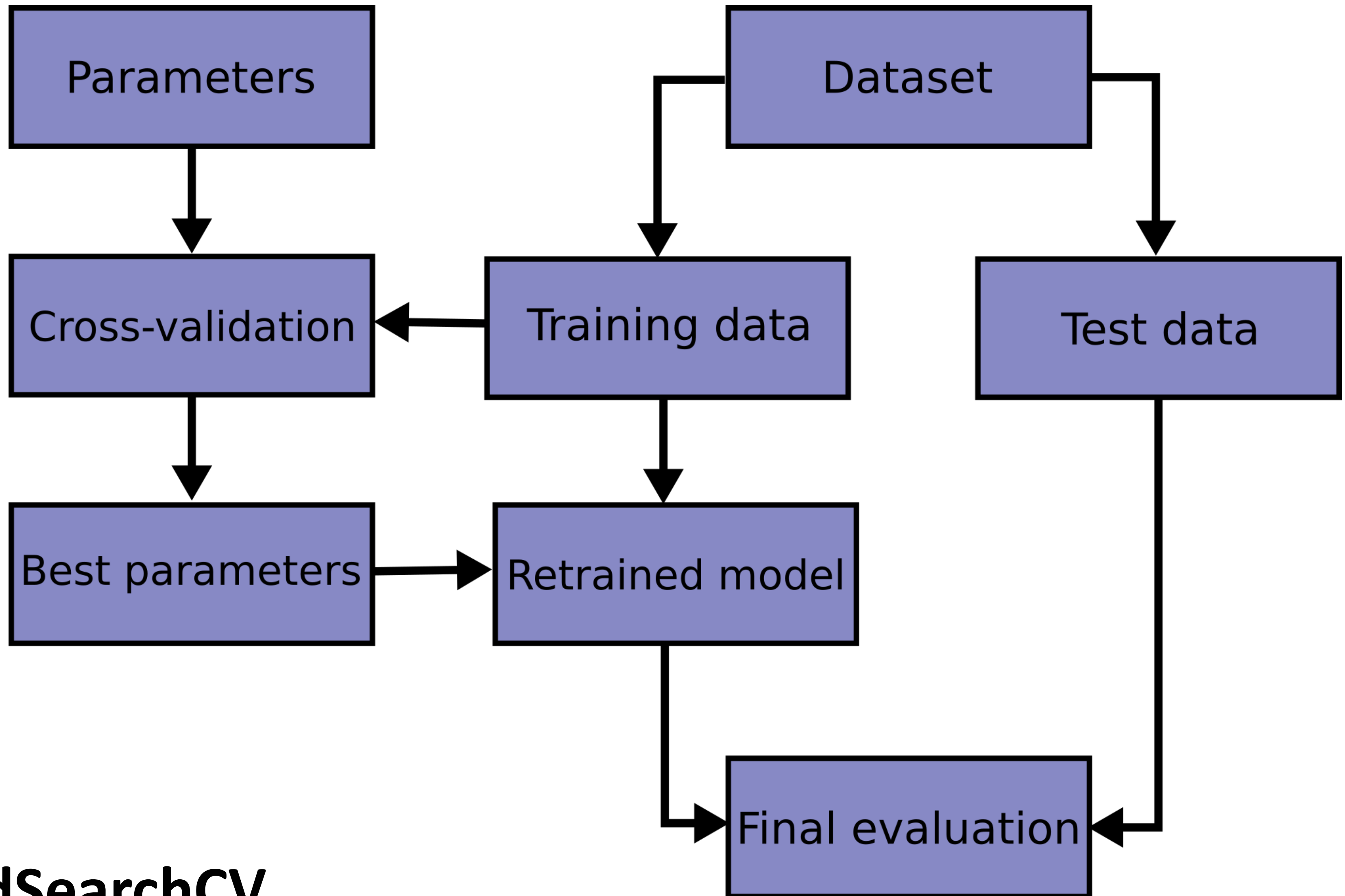
Image from Muller, AMLP

Grid Search vs. Cross-Validation

- Cross-validation is method to robustly evaluate a particular model

- Grid Search is a technique to tune hyper-parameters of a particular model (brute-force search)

Overall **BEST PRACTICE** Process



GridSearchCV

Image from scikit-learn

LOOCV – Leave-one-out-Cross-Validation

- LOOCV
  - number of folds = n, # of training instances
- Properties
  - gives good performance estimate
  - computationally expensive
  - non-stratified sample
- Limitations
  - Should only be run on very small data sets

k_1 x k_2 cross validation

- Repeated cross validation
  - replicate k_2 – fold cross validation, k_1 times
  - gives reliable estimates of performance
- Examples:
  - 5 x 2 c.v.
  - 10 x 10 c.v.
- Limitations:
  - computational cost

Nested Cross-Validation

- Nested Cross-validation performs cross-validation for both the train/validation split but also the train/test split
- Not commonly used:
  - Computational expense (another for loop added)
  - It doesn’t result in a single model
    - End up with different models for each split, that can have different “best” hyper-parameters
- Result is an estimate of how well a given model generalizes if the hyper-parameters are found using the inner method (e.g., grid search)
  - Is Method A better than B if both are properly tuned?

Summary

- We need to be able to select hyper-parameters and among different models.
- Use a three-stop process of training, validation, and testing on separate subsets of the data.
- Cross-validation adds robustness (and computational cost)
- Typically, use cross-validation + grid search to tune hyper-parameters.

Model Evaluation

- Supervised Learning Problems

- Methods for Performance Evaluation
  - How to obtain reliable estimates?

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

Metrics for Performance Evaluation

- Predictive capability of binary classification model
- Example Data

Input				Actual Class	Predicted Class	Correct?
0	1	45	small	1	1	Yes
1	3	24	medium	0	0	Yes
1	2	31	large	0	1	No
0	1	48	medium	1	0	No
…	…	…	…	…	…	…

Classification Evaluation: Confusion Matrix

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Actual Class

Predicted Class

	Fraud	**No Fraud**
Fraud	**0**	**0**
No Fraud	**10**	**990**

Classification Evaluation: Accuracy

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Accuracy

$$\frac{Num.\ of\ correct\ classifications}{Num.\ of\ total\ classifications} = \frac{TP + TN}{TP + FP + FN + TN}$$

Limits of Accuracy

- Consider a binary classification problem
  - number of 0 class examples = 9990
  - number of 1 class examples = 10

- Model may always predict class 0,
  - Accuracy = 9990 / 10000 = 99.9%

Classification Evaluation: Error Rate

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Error Rate

$$\frac{Num.\ of\ errors}{Num.\ of\ total\ classifications} = 1 - accuracy = \frac{FP + FN}{TP + FP + FN + TN}$$

Classification Evaluation: Precision

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Precision – num. samples predicted positive that really are?

$$Precision = \frac{TP}{TP + FP}$$

Classification Evaluation: Recall

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Recall – num. samples really +, that you predicted?

$$Recall = \frac{TP}{TP + FN}$$

Classification Evaluation: Sensitivity

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Sensitivity – true positive rate

$$Sensitivity = \frac{TP}{TP + FN}$$

Classification Evaluation: Specificity

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Specificity – true negative rate, proportion of TN found

$$Specificity = \frac{TN}{FP + TN}$$

Classification Evaluation: PPV

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Positive Predictive Value (PPV) – precision

$$PPV = Precision = \frac{TP}{TP + FP}$$

Classification Evaluation: NPV

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

Negative Predictive Value (NPV)

$$NPV = \frac{TN}{TN + FN}$$

Classification Evaluation: F_1-measure

- M – model predicted class / outcome
- A – actual class / outcome

		Actual Outcome		
		A+	A-	
Model Outcome	M+	TP	FP	TP + FP
	M-	FN	TN	FN + TN
		TP + FN	FP + TN	*n*

- TP-true positive
- TN-true negative
- FP-false positive
- FN-false negative

F_1-measure – harmonic mean of precision and recall

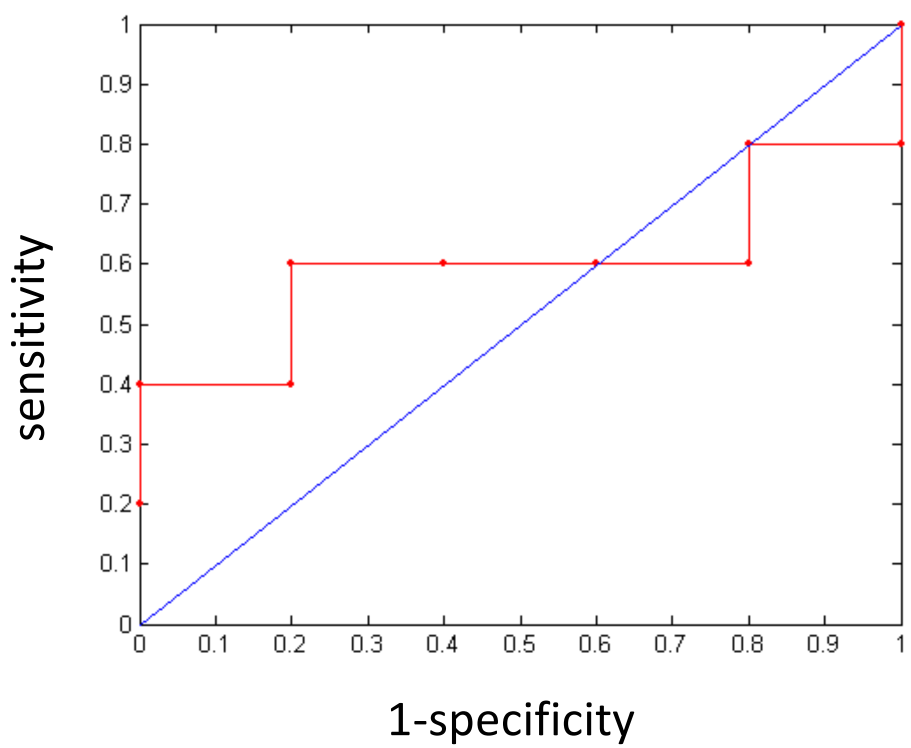
$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

Classification Evaluation: ROC

- ROC – Receiver Operating Characteristic Curve
  - visualizes the trade-off between positive hits and false alarms
  - plots sensitivity vs. 1-specificity, or tp-rate (y-axis) vs. fp-rate (x-axis)
- A curve is formed by changing the threshold of the model

Classification Evaluation: ROC

Class	+	-	+	-	-	-	+	-	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0



(TPR,FPR):

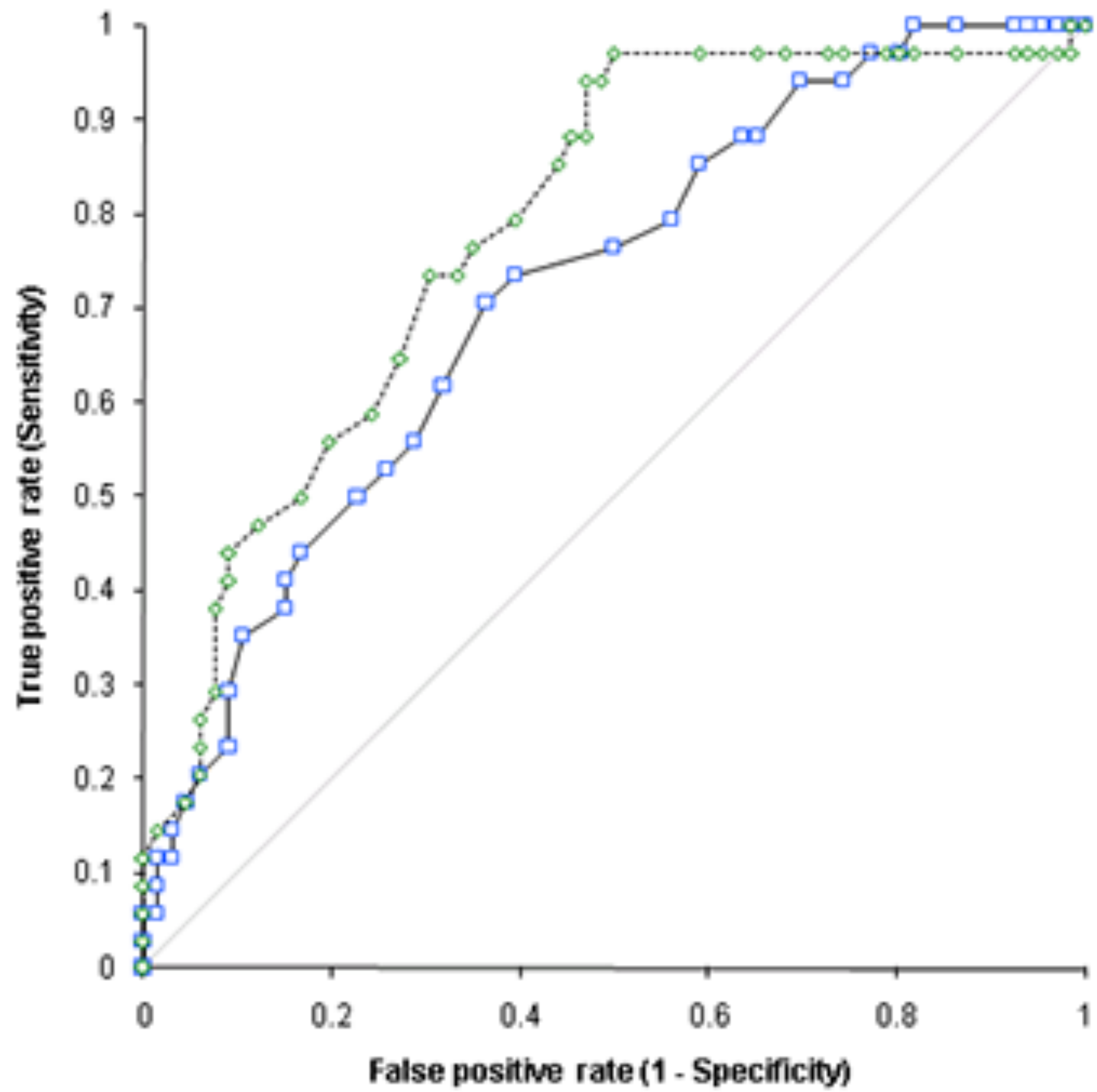
- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

Diagonal line:

- Random guessing

Classification Evaluation: AUC

- AUC may be used to compare performance of two methods, select parameters for a single model
- Must still pay attention to Occam's Razor



Regression Evaluation: Loss or Error

- Metrics for regression
- 0/1 Loss

$$\frac{1}{n}\sum_{i=1}^{n} \begin{matrix} 0 & \mathit{if}\ \text{pred. outcome=actual outcome} \\ 1 & \mathit{otherwise} \end{matrix}$$

- L1 loss

$$\frac{1}{n}\sum_{i=1}^{n}\left|(pred.\,outcome_i - actual\ outcome_i)\right|$$

- Mean squared error (MSE), quadratic loss

$$\frac{1}{n}\sum_{i=1}^{n}(pred.\,outcome_i - actual\ outcome_i)^2$$