Evaluation

DATA MINING

LI ZENG

Contact: L.Zeng@tilburguniversity.edu

Model Evaluation

- SSE as a model goodness indicator?
- Why not?
  - Size -> as we add more datapoints, SSE?
  - Interpretation -> SSE is measured in squared units
  - Comparability -> units of measurement are different, SSE?
- Solution?
  - To fix the size issue -> mean squared error (MSE)
  - To fix the interpretation issue -> root mean squared error (RMSE)
  - To fix the comparability issue -> R-squared (R^2)
  - To improve sensitivity to outliers -> mean absolute error (MAE)

$$\begin{aligned} SSE(\beta_0, \beta_1) &= \sum_{i=1}^{N} e_i^2 = \\ &= \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \\ &= \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 \cdot x_i)]^2. \end{aligned}$$

$$MSE = \frac{1}{N} \sum_{i=1}^{N} e_i^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

$$RMSE = \sqrt{MSE}.$$

$$MAE(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

R^2

- SSE does not tell much about model performance unless we know how data spread out
- Define total sum of squares (TSS):

$$TSS = \sum_{i=1}^{N} (y_i - \bar{y})^2$$

- TSS is the total variation in the data
- SSE is the variation left unexplained by the model
- Define R^2:

$$R^2 = 1 - \frac{\text{SSE}}{\text{TSS}}.$$

R^2 - when is good enough?

- ❑Range – [0,1]
- ❑Prediction – high R^2 as possible
- ❑Inference – less important? Focus on interpretation of betas
- ❑In social science, it is common to observe R^2 around 0.3 for ordinary regression
- ❑Last note: SSE, TSS and R^2 are well defined for all supervised learning models with continuous outcomes

Adjusted R^2

- ❑When we add more predictors to a simple linear regression model, consider how the value of R^2 may change?

- ❑Recall: $$R^2 = 1 - \frac{\text{SSE}}{\text{TSS}}.$$

$$\min_{\beta} \sum_{i}^{n} [y_i - \widehat{y}_i(\beta)]^2$$

- ❑When an extra predictor (variable) is included, SSE?

- ❑Adjusted R^2 adjusts for the number of predictor in a model
  - ❑Adding more useless variables -> penalty

Classification

- RMSE and R^2 as model goodness indicators for classification?
- Why not?
  - Categories are nominal or ordinal measures, hence $y_i - \hat{y_i}$ is not well-defined
  - Prediction output: 0 or 1. Unable to distinguish between types of errors
  - $y_i - \hat{y_i}$ fails to tell how “far” off predictions from the observations are
- Solutions
  - To fix the first and second issues -> confusion matrix
  - To fix the third issue -> predicted probabilities of category (next week: logistic regression)

Confusion Matrix

- A table of the actual and predicted classes
- P – actual positives; N – actual negatives
- $\hat{P}$ - predicted positives; $\widehat{N}$ - predicted negatives

Actual	Predicted		
	-	+	Total
-	TN	FP	N
+	FN	TP	P
Total	$\widehat{N}$	$\hat{P}$	T

Confusion Matrix

□ True positives (TP) are actually positive, and are correctly predicted as positive.

□ True negatives (TN) are actually negative and are correctly predicted as negative.

□ False positives (FP), also type-I errors, are cases actually negative but predicted as positive.

□ False negatives (FN), also type-II errors, are cases actually positive but predicted as negative.

Actual	Predicted		
	-	+	Total
-	TN	FP	N
+	FN	TP	P
Total	$\widehat{N}$	$\widehat{P}$	T

Model Goodness Measures

- Accuracy: percentage of correct answers

$$Accuracy = \frac{TP + \mathrm{TN}}{T}$$

- Recall: percentage of actual positives that are correctly identified as positives

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

- Precision: percentage of predicted positives that turn out to be correct

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

Model Goodness Measures

❑F-score: attempt to find a balance between recall and precision – a harmonic mean of these two measures

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

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

Exercise

Look at classifying cases into two color categories: red and yellow. Assume yellow is positive. Compute accuracy, precision, recall and F-score.

Actual	Predicted		
	Red	Yellow	Total
Red	10	20	30
Yellow	10	60	70
Total	20	80	100

Consider:

- Given the confusion matrix, how good is this model?

Actual	Predicted		
	-	+	Total
-	2900	100	3000
+	50	50	100
Total	2950	150	3100

More measures (names)

- True positive rate or sensitivity = recall
- Specificity = recall for negative outcomes

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

- False positive rate: percentage of negatives that are falsely categorized as positives

$$False\ positive\ rate = \underline{1 - Specificity} = \frac{FP}{TN + FP}$$

- False negative rate: percentage of positives that are falsely categorized as negatives:

$$False\ negative\ rate = \frac{FN}{P} = \frac{FN}{TP + FN}$$

Precision/Recall tradeoff

- In some cases, we can decide whether a high precision or recall is more important
  - Scenario 1: A model to classify which videos are suitable for kids to watch.
  - Scenario 2: A model to detect a patient is having a disease or not
  - Scenario 3: A model to help judge a person as guilty

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

- When precision and recall are equally important:

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

Overfitting

- When working with machine learning models, we typically start with the “learning” part (i.e., model training)
- The data we use for training is called **training data**
- Later, we use the model for testing and predicting
- The data we use for evaluating the model performance is called **test data**
- If a model fits training data too well, but its flexibility may lead to really poor performance on unseen data -> overfitting

Overfitting

Let’s look at this toy dataset. We want to model the relationship between income and age.

Consider a number of different polynomial regression models:

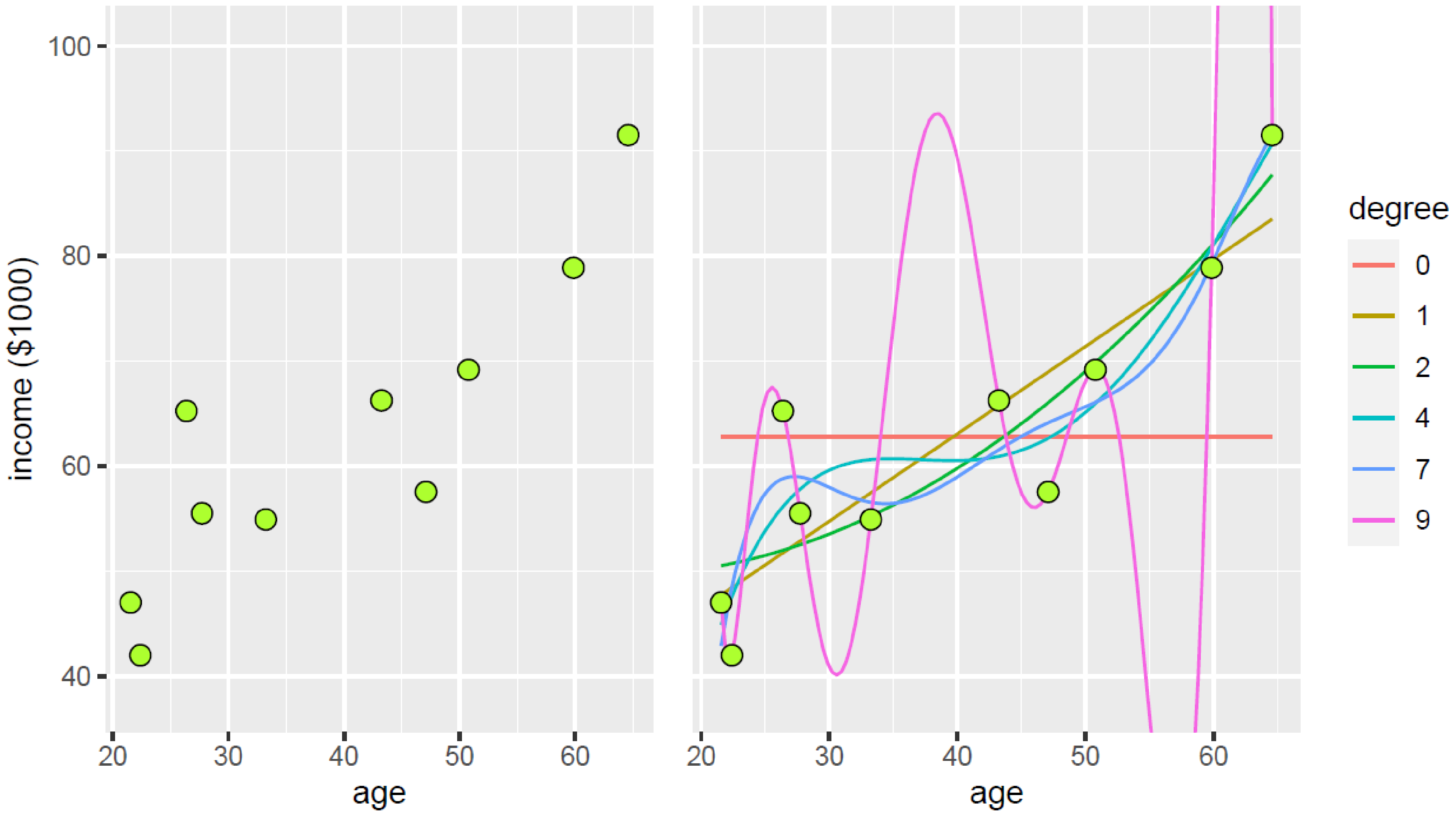
$$y_i = \beta_0 + \beta_1 \cdot age_i + \epsilon_i$$

$$y_i = \beta_0 + \beta_1 \cdot age_i + \beta_2 \cdot age_i^2 + \epsilon_i$$

$$\vdots$$

$$y_i = \beta_0 + \beta_1 \cdot age_i + \beta_2 \cdot age_i^2 + \cdots + \epsilon_i$$

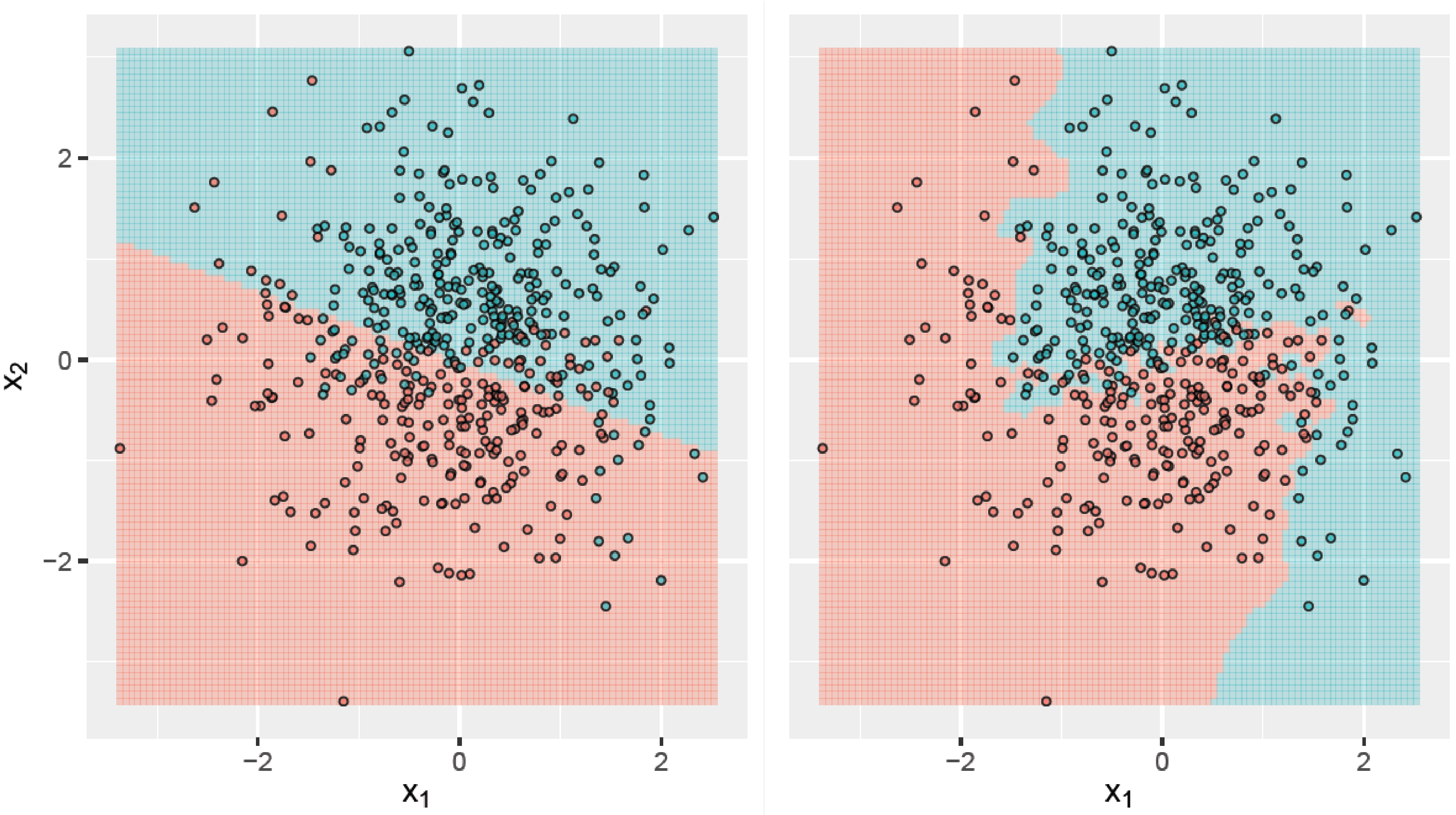
Degree	0	1	4	6	8
R^2	0	0.59	0.61	0.88	0.99



Degree	0	1	4	7	9
RMSE	14.02	6.64	4.75	4.36	0.00

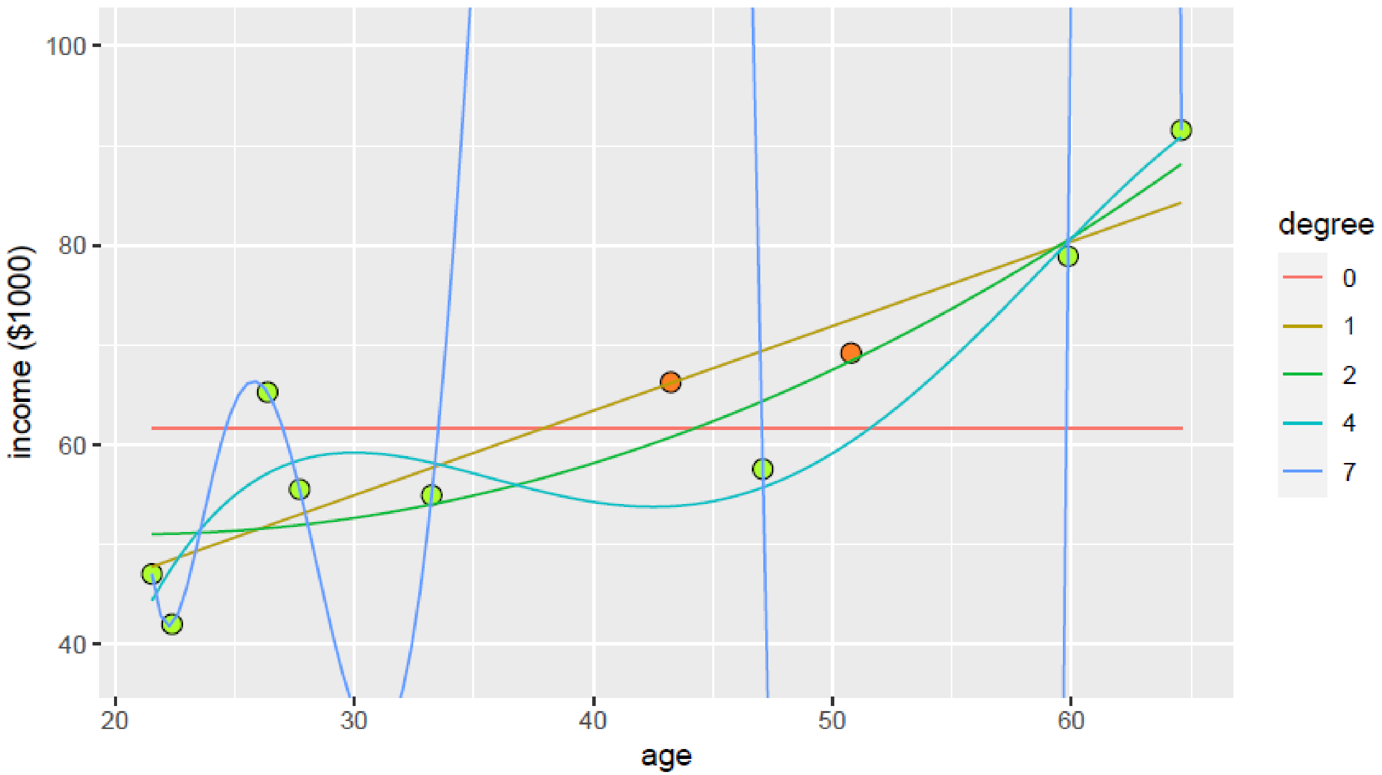
Overfitting

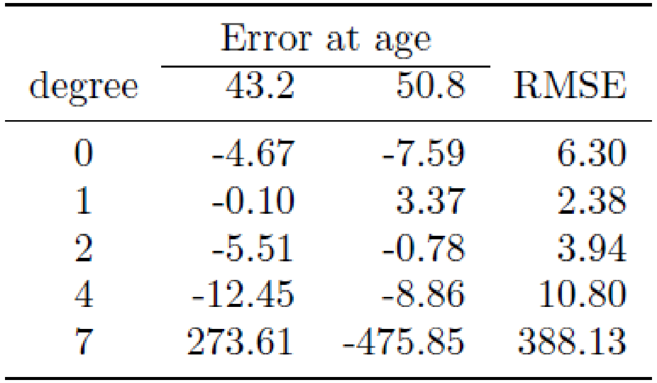
- Underfitting:
  - additional complexity improves test and training performance
  - test, training performance equal
- Overfitting:
  - additional complexity improves training performance
  - testing performance deteriorates
  - test performance much worse than training performance
- Optimal complexity
  - optimum of test performance


2
0
−2
x_2
−2
0
2
x_1
−2
0
2
x_1

Validation

- □If we have additional data, we could use model to predict and compare the predictions with the “known answers” in this additional dataset
- □Instead of using all data to train a model, we can keep a small portion for test
- □For example, we hold out two data points age 43.2 and 50.8 and use everything else to train





	Error at age		
degree	43.2	50.8	RMSE
0	-4.67	-7.59	6.30
1	-0.10	3.37	2.38
2	-5.51	-0.78	3.94
4	-12.45	-8.86	10.80
7	273.61	-475.85	388.13

K-fold Cross Validation

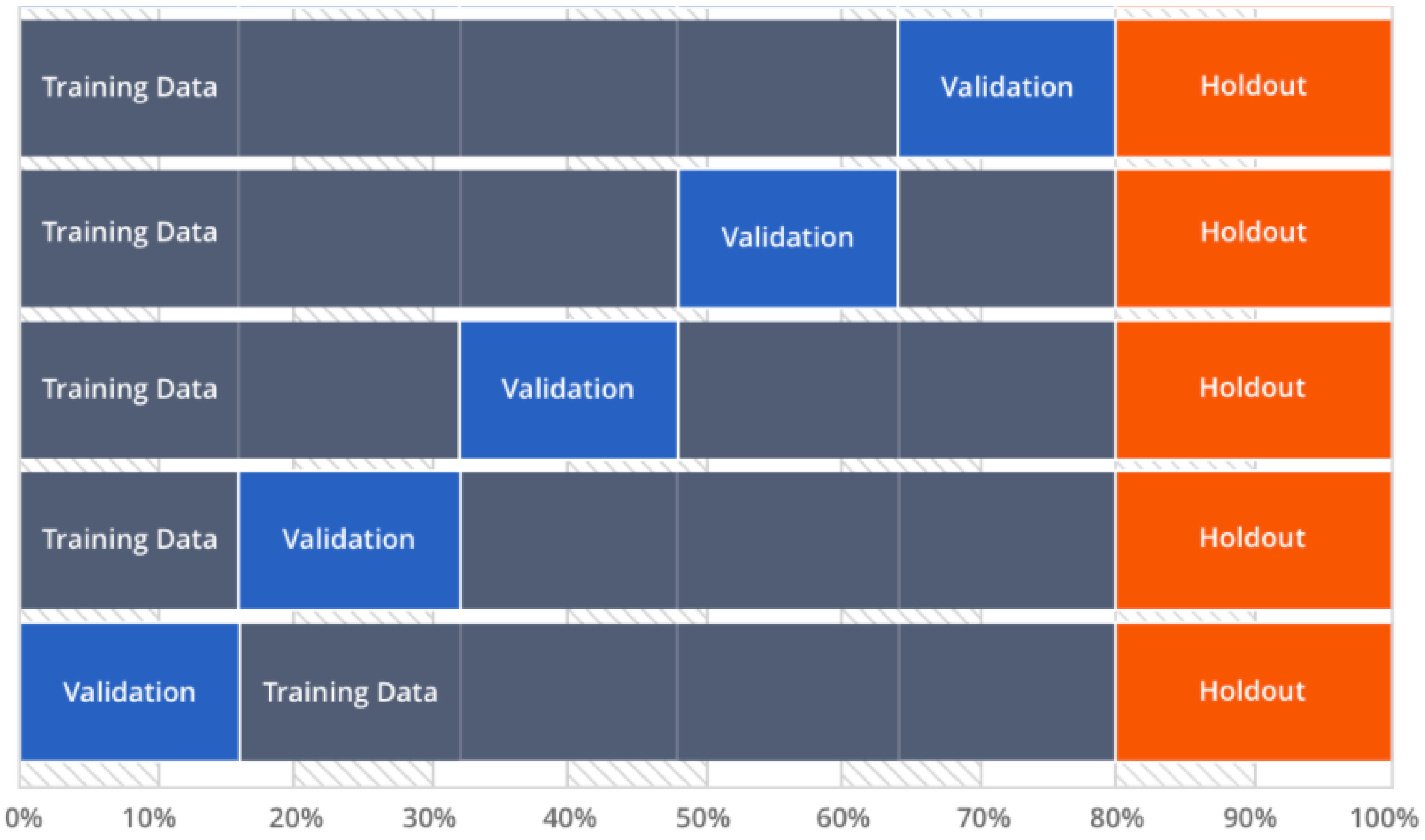
❑Partition the data into K disjoint subsets $C_k = C_1, C_2, \dots, C_K$.

❑Conduct K training replications.

❑For each training replication, collapse K-1 partitions into a set of training data.

❑Compute the test measurement (eg. RMSE) for the kth partition, $RMSE_k$, by using subset C_K as the test data for the kth fitted model.

❑Compute the overall K-fold cross validation error as $\sum_{k=1}^{K} \frac{N_k}{N} RMSE_k$


Training Data
Validation
Holdout
Training Data
Validation
Holdout
Training Data
Validation
Holdout
Training Data
Validation
Holdout
Validation
Training Data
Holdout
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

Training-Validation-Testing

- Split your data into working and testing (holdout) data
  - Testing data is only for the final test
- Split your working data into training and validation data
  - use the validation data for hyperparameter tuning
  - model selection
  - compare different models on validation data
- Report the final performance using testing data.
  - do not look at your testing data before the final test!