Cross Validation

Cross Validation

- Cross validation is a more advanced set of methods for splitting data into training and testing sets.
- Cross Validation Relevant Reading:
    - Section 5.1 of ISLR

Cross Validation

- We understand the intuition behind performing a train test split, we want to fairly evaluate our model’s performance on unseen data.
- Unfortunately this means we are not able to tune hyperparameters to the **entire** dataset.

Cross Validation

- Is there a way we can achieve the following:
    - Train on all the data
    - Evaluate on all the data
- While it sounds impossible, we can achieve this with cross validation!
- Let’s have an overview of the concept...

Cross Validation

- Imagine our data set:

X			y
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Cross Validation

- Let's convert this data into colored blocks for cross-validation

X			y
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Cross Validation

- Convert to generalized form

X			y
x_1	x_2	x_3	y
x^1_1	x^1_1	x^1_1	y_1
x^2_1	x^2_1	x^2_1	y_2
x^3_1	x^3_1	x^3_1	y_3
x^4_1	x^4_1	x^4_1	y_4
x^5_1	x^5_1	x^5_1	y_5

Cross Validation

- Color based off train vs. test set.

X			y
x_1	x_2	x_3	y
x^1_1	x^1_1	x^1_1	y_1
x^2_1	x^2_1	x^2_1	y_2
x^3_1	x^3_1	x^3_1	y_3
x^4_1	x^4_1	x^4_1	y_4
x^5_1	x^5_1	x^5_1	y_5

Cross Validation

- Color based off train vs. test set.

	X			**y**
	x_1	x_2	x_3	y
TRAIN	x^1_1	x^1_1	x^1_1	y_1
	x^2_1	x^2_1	x^2_1	y_2
	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
	x^5_1	x^5_1	x^5_1	y_5

Cross Validation

- Color based off train vs. test set.

	X			**y**
	x_1	x_2	x_3	y
TRAIN	x^1_1	x^1_1	x^1_1	y_1
	x^2_1	x^2_1	x^2_1	y_2
	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
	x^5_1	x^5_1	x^5_1	y_5

Cross Validation

- For now just consider training vs testing:

	X			y
	x_1	x_2	x_3	y
TRAIN	x^1_1	x^1_1	x^1_1	y_1
TRAIN	x^2_1	x^2_1	x^2_1	y_2
TRAIN	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
TEST	x^5_1	x^5_1	x^5_1	y_5

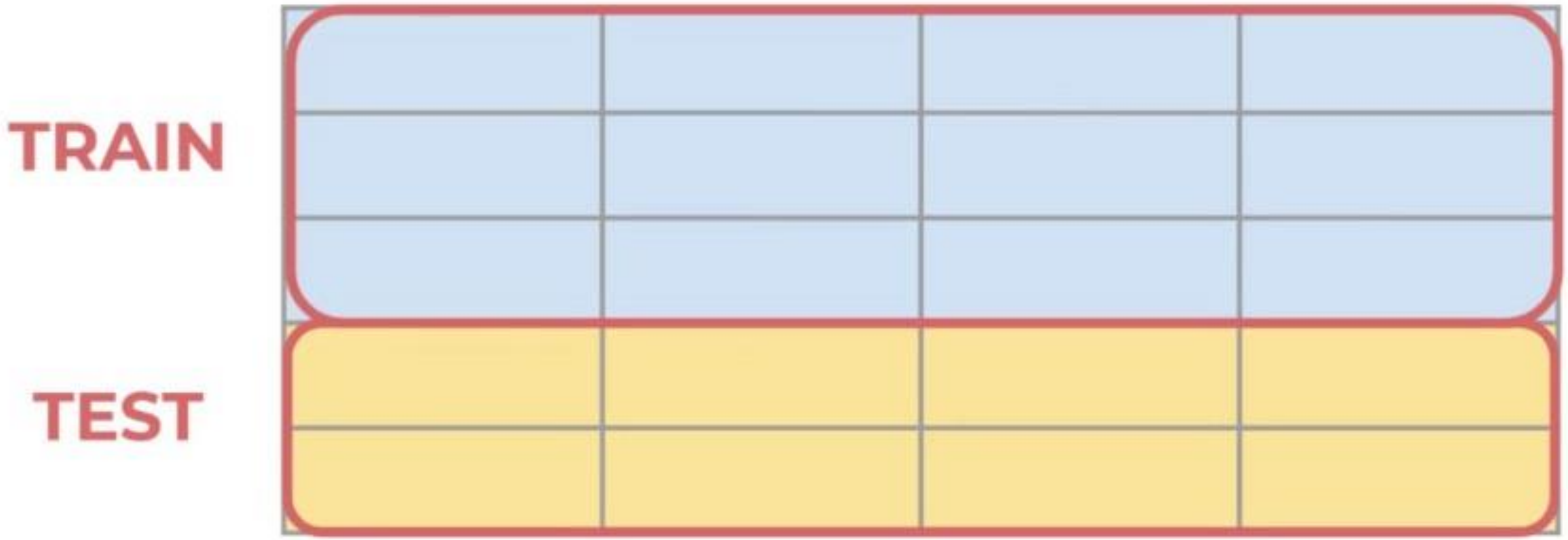
Cross Validation

- For now just consider training vs testing:

TRAIN	x^1_1	x^1_1	x^1_1	y_1
	x^2_1	x^2_1	x^2_1	y_2
	x^3_1	x^3_1	x^3_1	y_3
TEST	x^4_1	x^4_1	x^4_1	y_4
	x^5_1	x^5_1	x^5_1	y_5

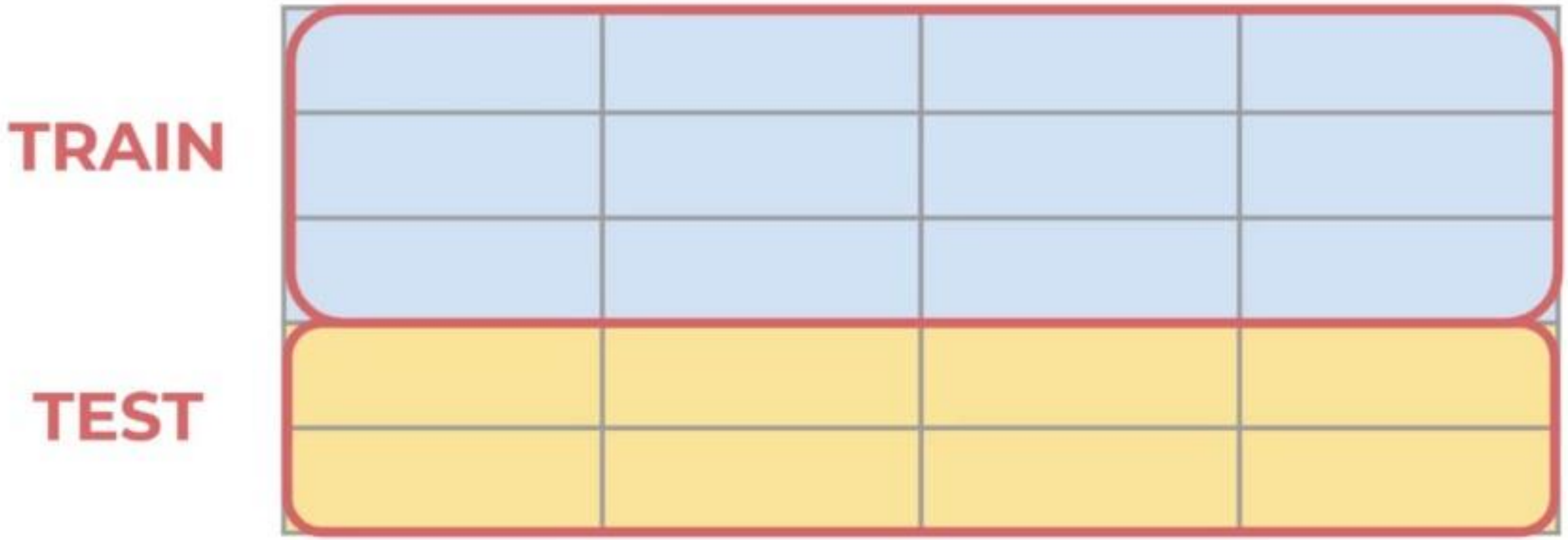
Cross Validation

- Now we have all data, colored by training set versus test set.



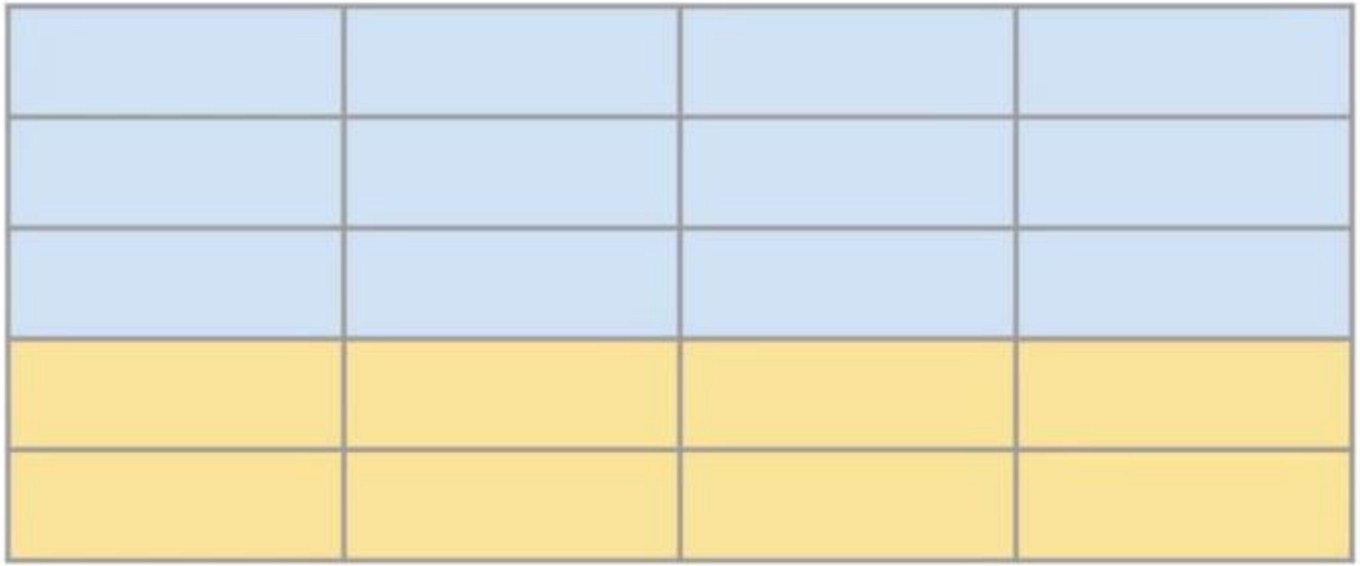
Cross Validation

- Rotate and resize:



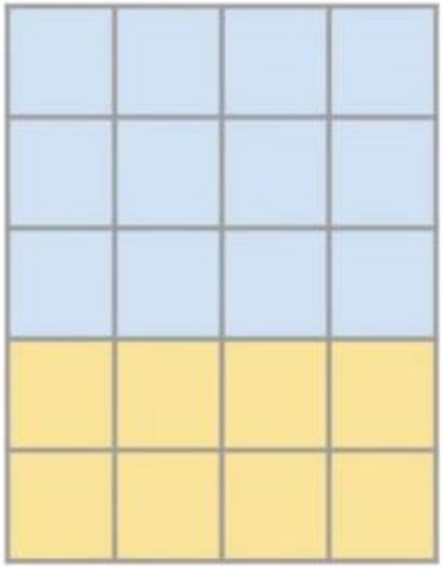
Cross Validation

- Rotate and resize:



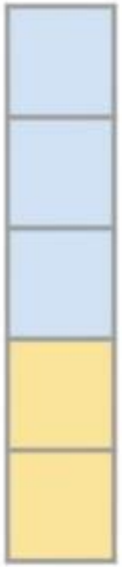
Cross Validation

- Rotate and resize:



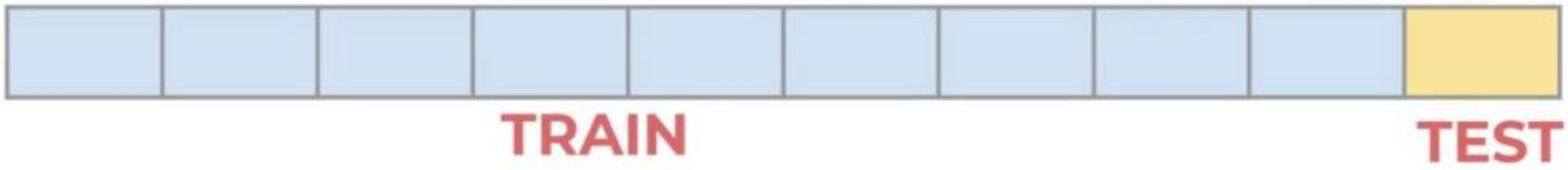
Cross Validation

- Rotate and resize:



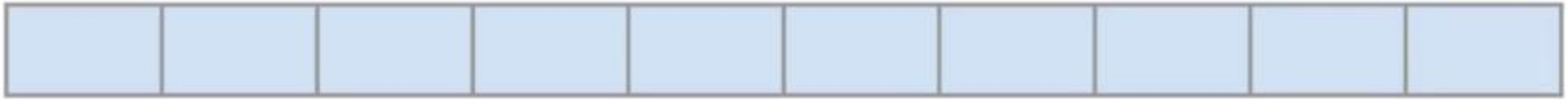
Cross Validation

- Now we can represent full data and splits:



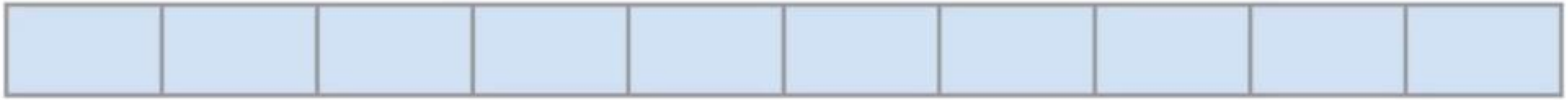
Cross Validation

- Let’s start with the entire original data:



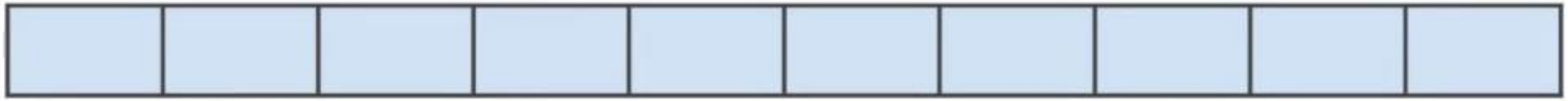
Cross Validation

- How does cross validation work?



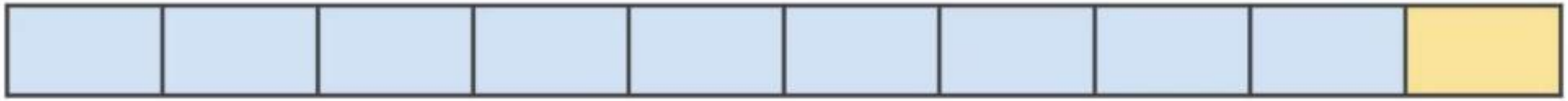
Cross Validation

- Split data into K equal parts:



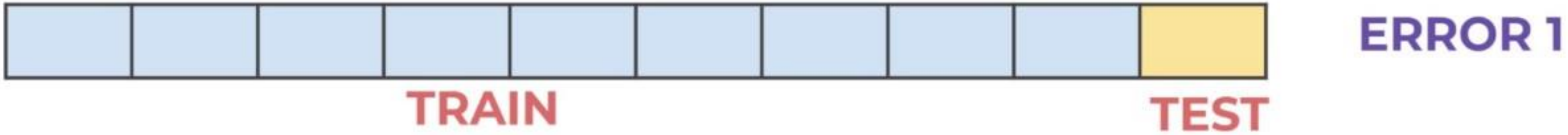
Cross Validation

- 1/K left as test set



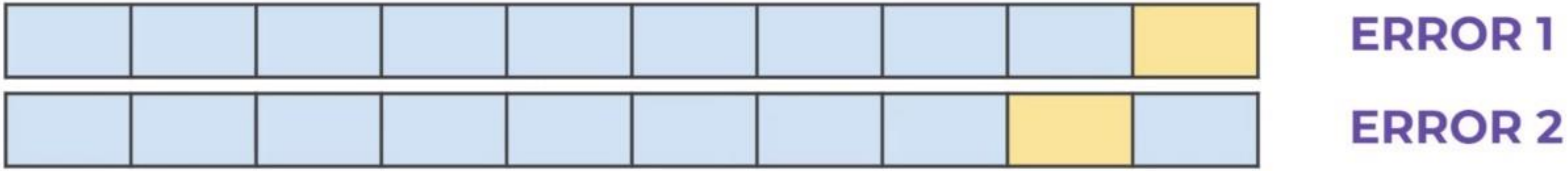
Cross Validation

- Train model and get error metric for split:



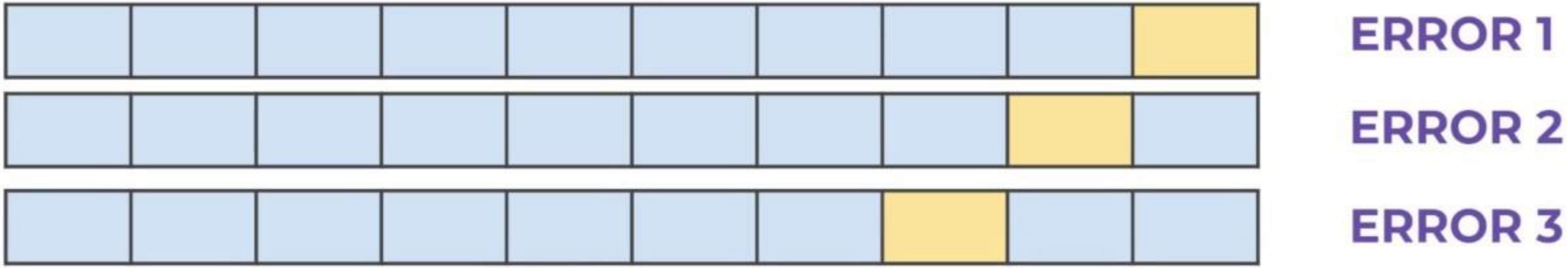
Cross Validation

- Repeat for another 1/K split



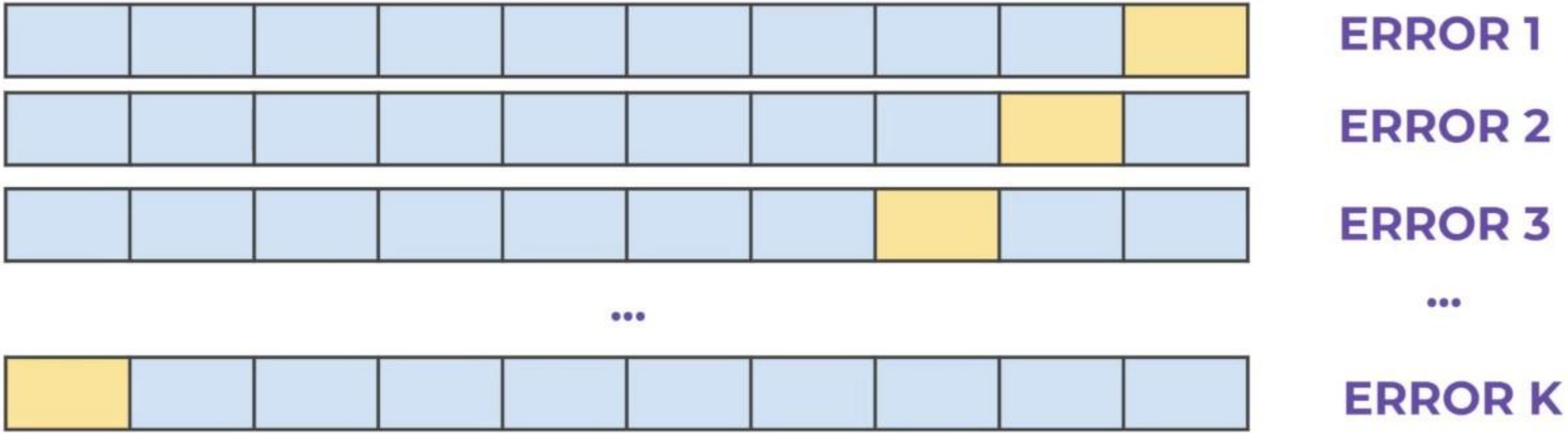
Cross Validation

- Keep repeating for all possible splits



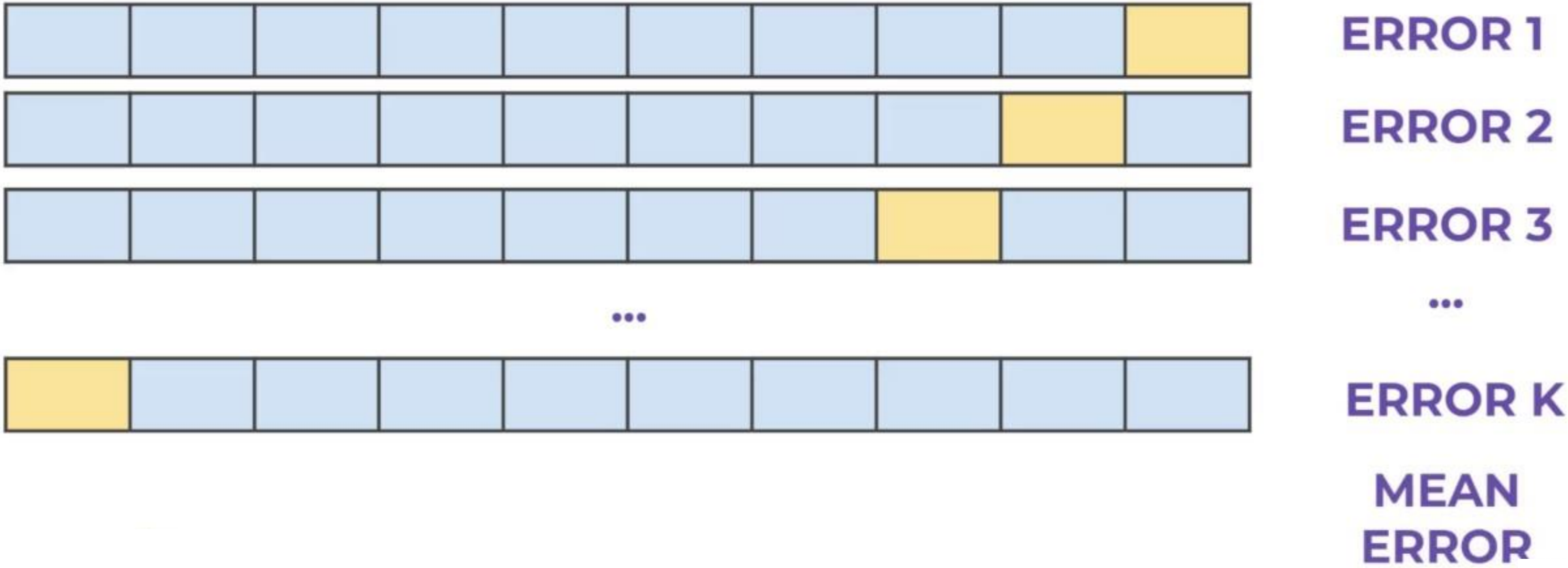
Cross Validation

- Keep repeating for all possible splits



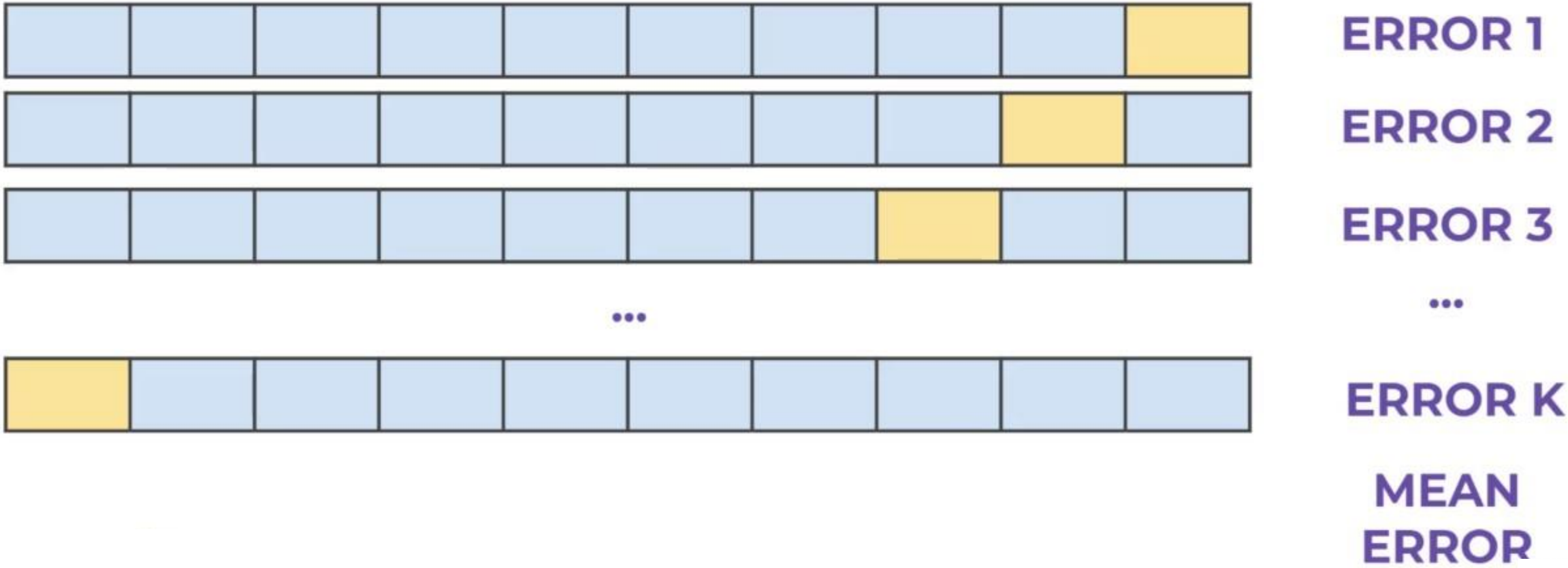
Cross Validation

- Get average error



Cross Validation

- Average error is the expected performance



Cross Validation

- We were able to train on all data **and** evaluate on all data!
- We get a better sense of true performance across multiple potential splits.
- What is the cost of this?
  - We have to repeat computations K number of times!

Cross Validation

- This is known as K-fold cross-validation.
- Common choice for K is 10 so each test set is 10% of your total data.
- Largest K possible would be K equal to the number of number of rows.
  - This is known as **leave one out** cross validation.
  - Computationally expensive!

Cross Validation

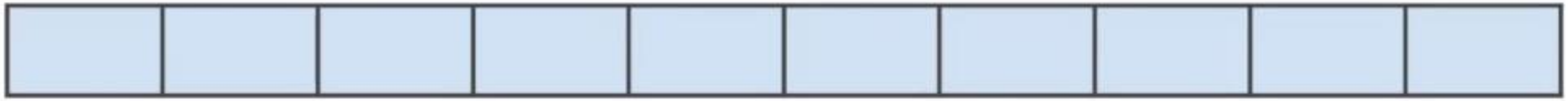
- One consideration to note with K-fold cross validation and a standard train test split is fairly tuning hyperparameters.
- If we tune hyperparameters to test data performance, are we ever fairly getting performance metrics?

Cross Validation

- How can we understand how the model behaves for data that is has not seen **and** not been influenced by for hyperparameter tuning?
- For this we can use a **hold out** test set.
- Let’s explore what this looks like...

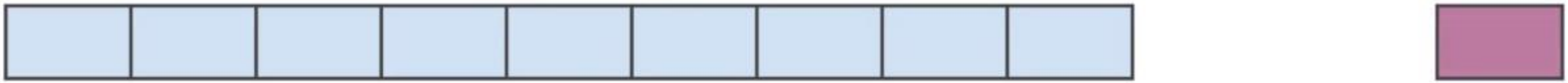
Cross Validation

- Start with entire data set:



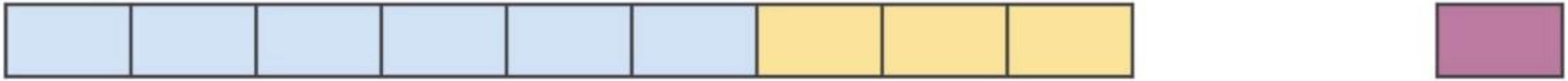
Cross Validation

- Remove a hold out test set



Cross Validation

- Perform “classic” train test split:

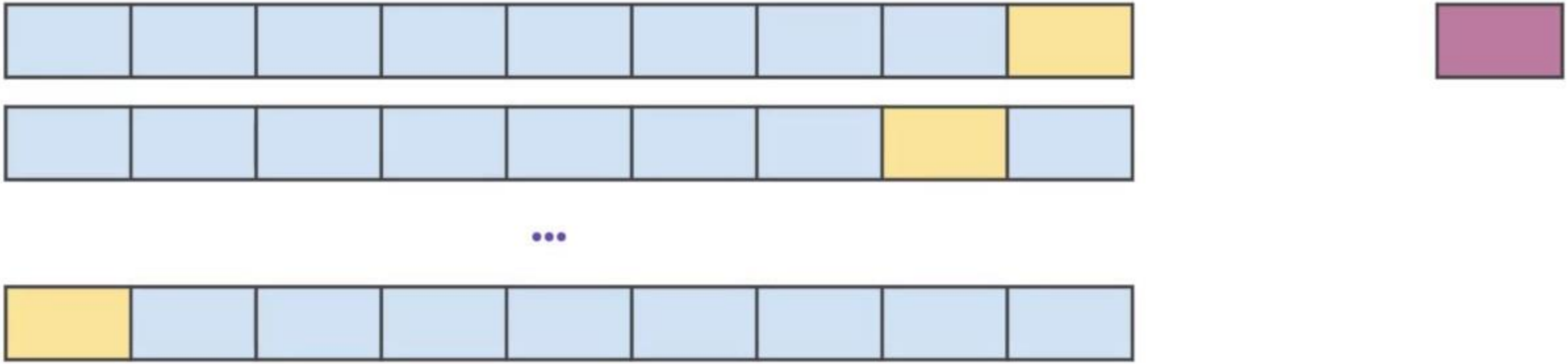


Cross Validation

- Train and tune on this data:

Cross Validation

- Or K-Fold cross validation



Cross Validation

- Train **and** tune on this data:

Cross Validation

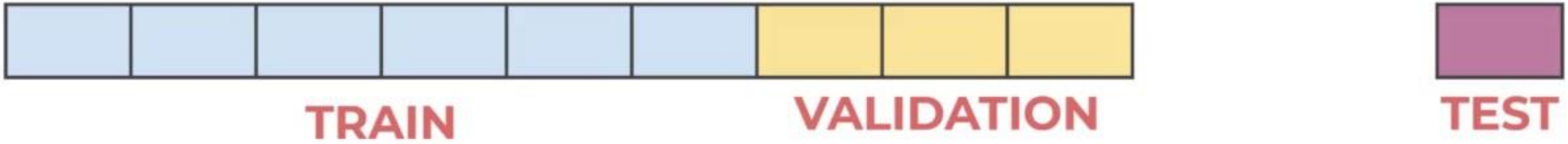
- **After** training and tuning perform **final evaluation** hold out test set.

Cross Validation

- Can **not** tune after this **final** test evaluation!

Cross Validation

- Train | Validation | Test Split



- Allows us to get a true final performance metric to report.
- No editing model after this!

Cross Validation

- All these approaches are valid, each situation is unique!
- Keep in mind:
  - Previous modeling work
  - Reporting requirements
  - Fairness of evaluation
  - Context of data and model

Cross Validation

- Many regularization methods have tunable parameters we can adjust based on cross-validation techniques.
- For simplicity, there are times in the course we will opt for a simple two part train test split.