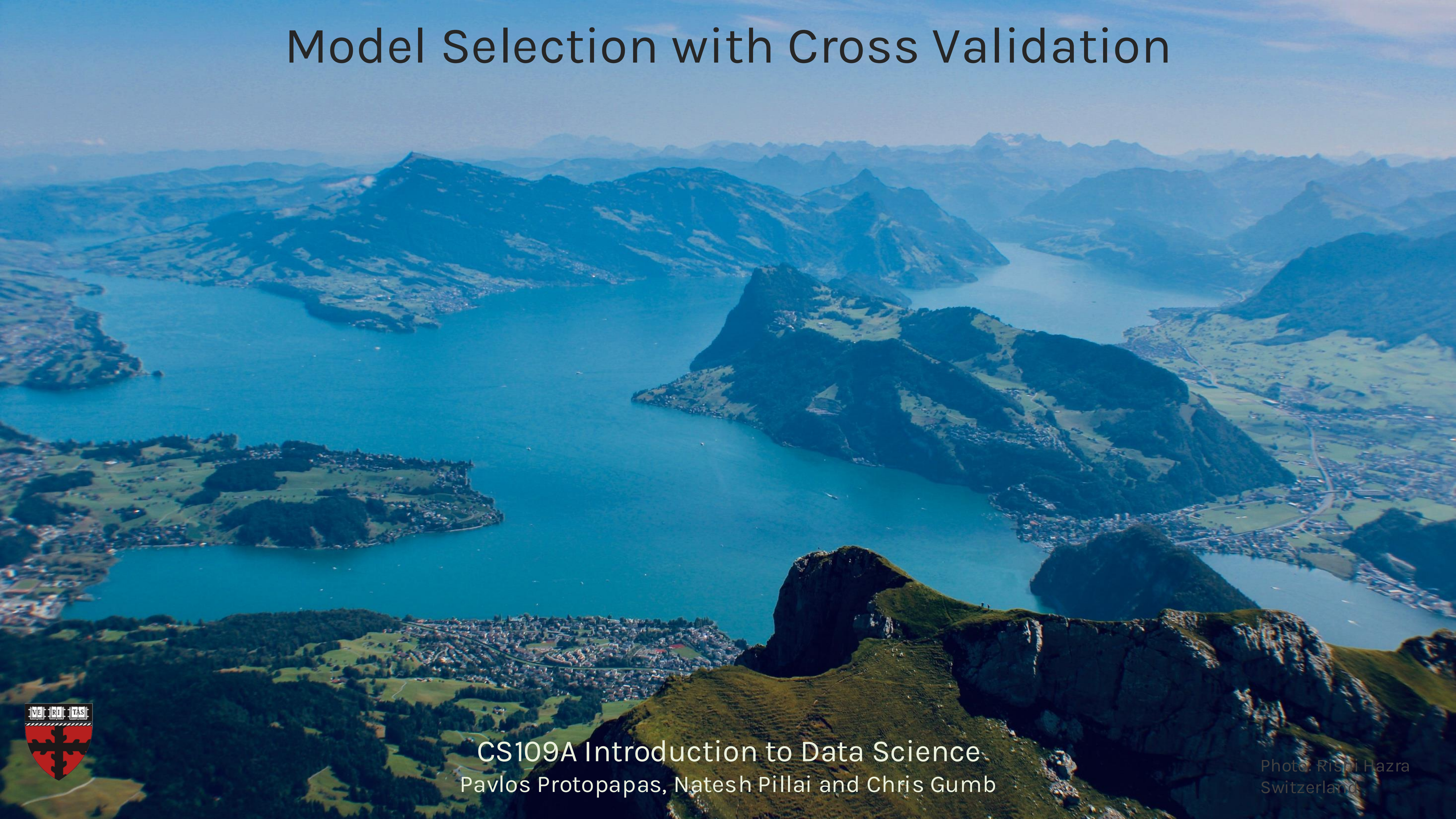
Model Selection with Cross Validation

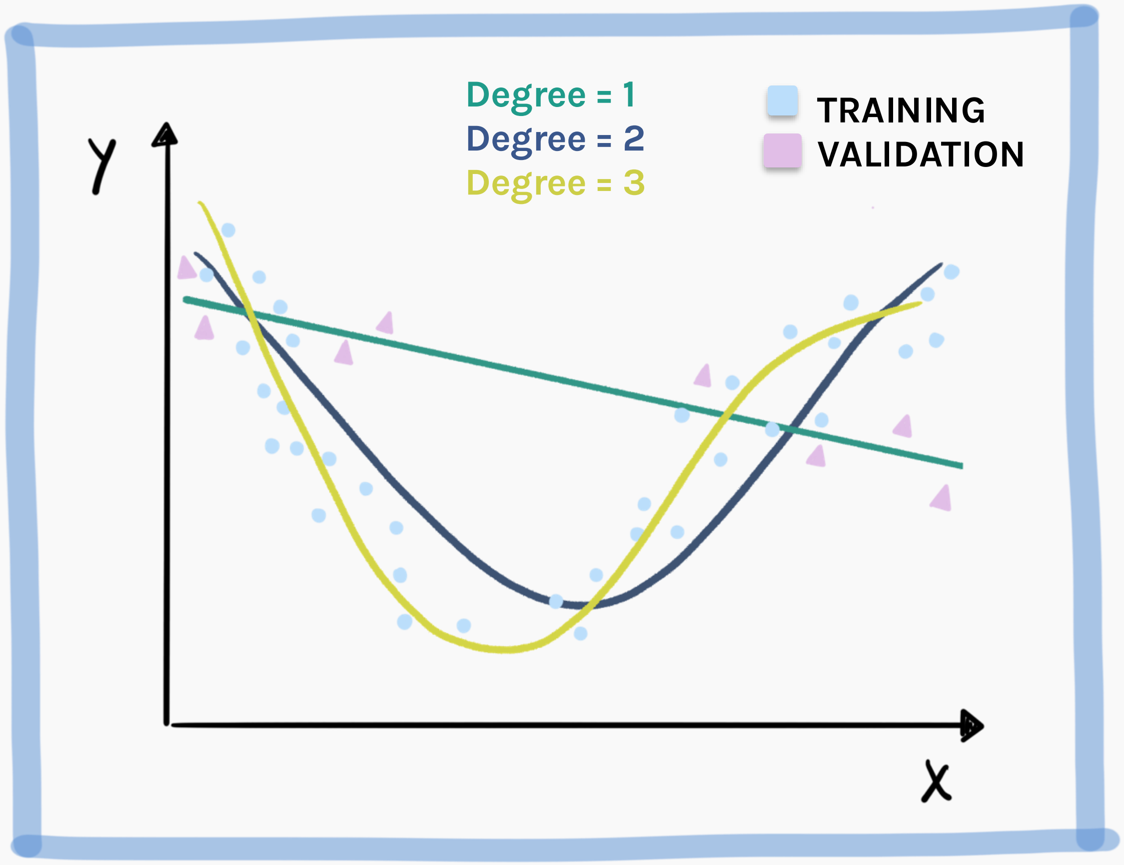


CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb

Photo: Rishi Hazra
Switzerland

Cross Validation: Motivation



It is obvious that degree=3 is the correct model, but the validation set by chance favors the linear model.

Using a single validation set to select amongst multiple models can be problematic - **there is the possibility of overfitting to the validation set**.

Cross Validation: Motivation

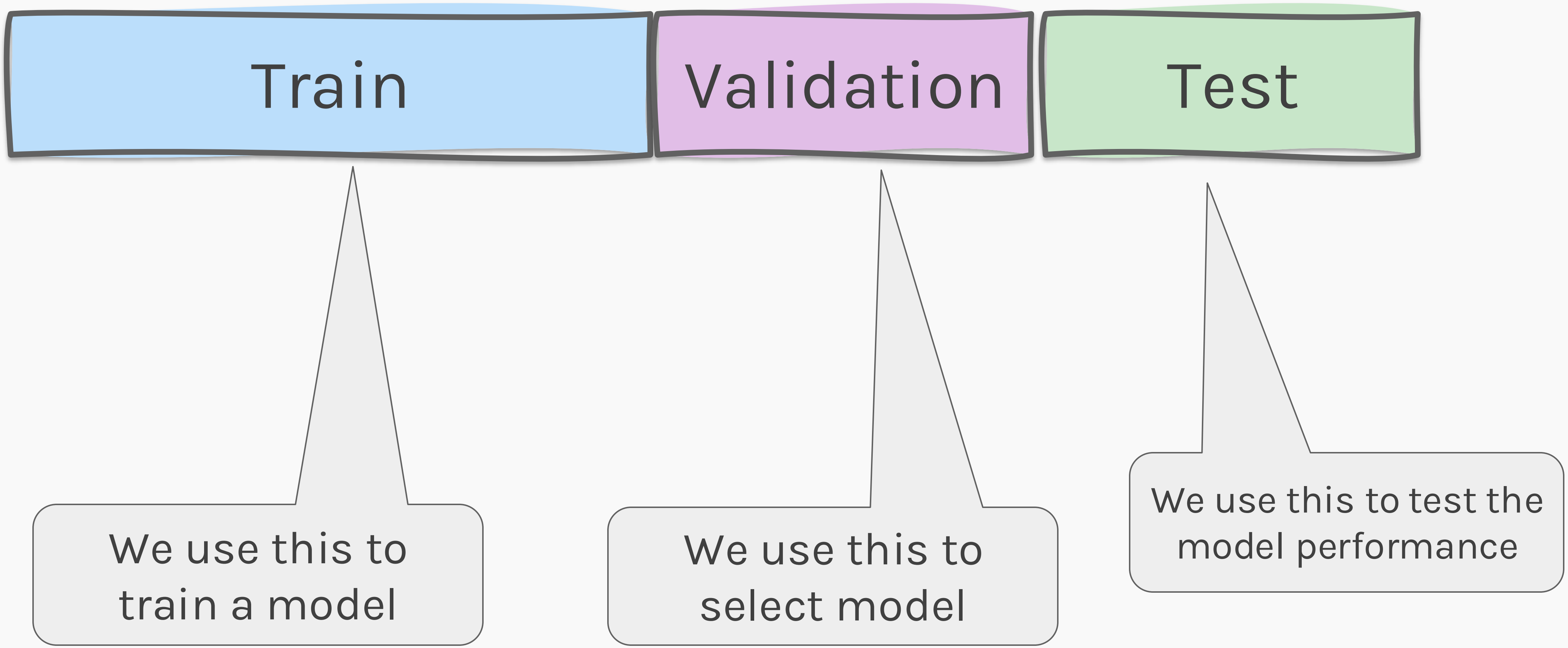
Using a single validation set to select amongst multiple models can be problematic - **there is the possibility of overfitting to the validation set**.

One solution to the problems raised by using a single validation set is to evaluate each model on **multiple** validation sets and average the validation performance.

One can randomly split the training set into training and validation multiple times **but** randomly creating these sets can create the scenario where important features of the data never appear in our random draws.

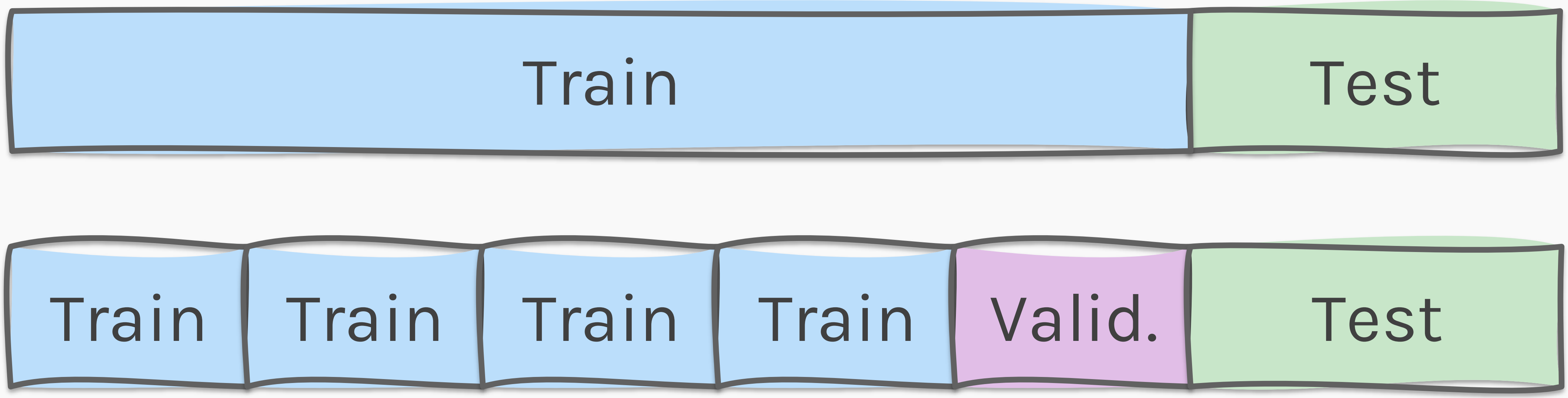
Train-Validation-Test

We introduced a different sub-set, which we called validation and we use it to select the model.

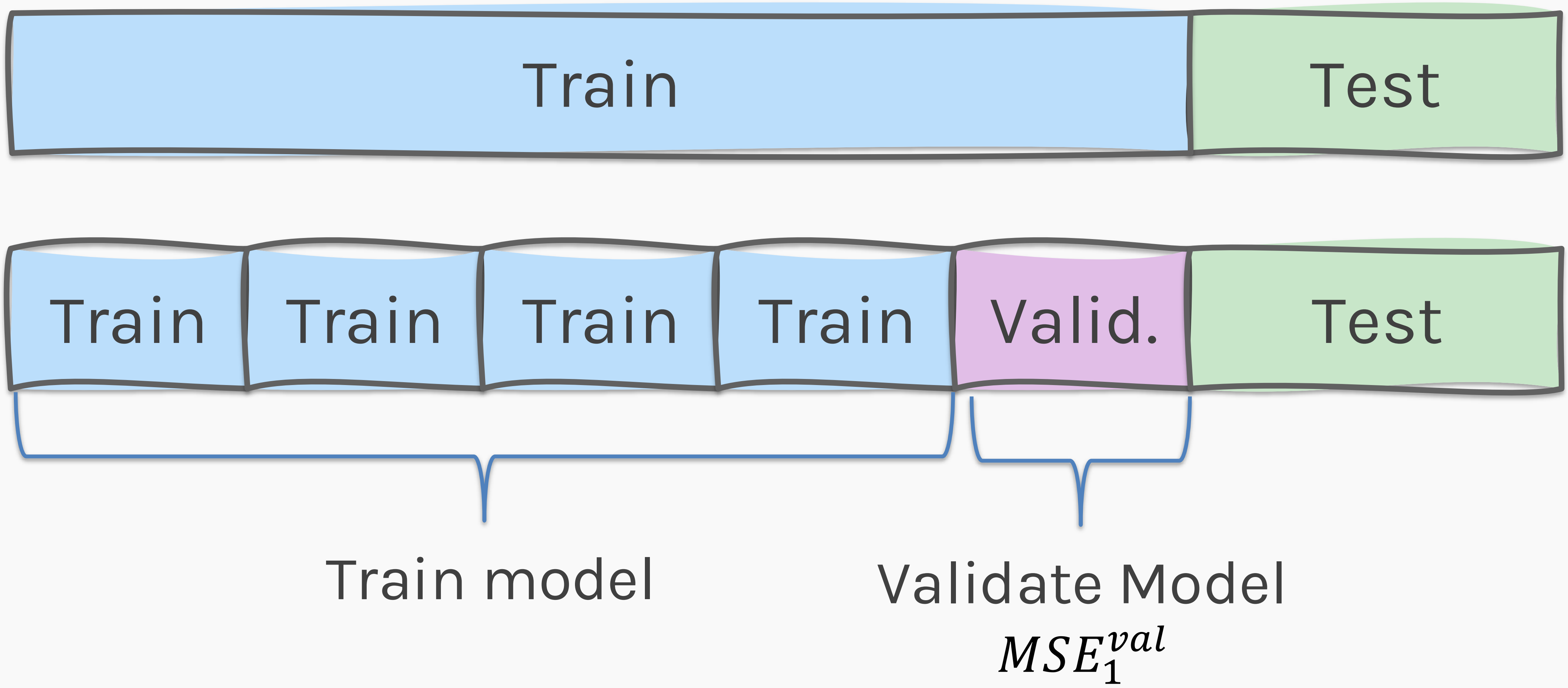


Cross Validation

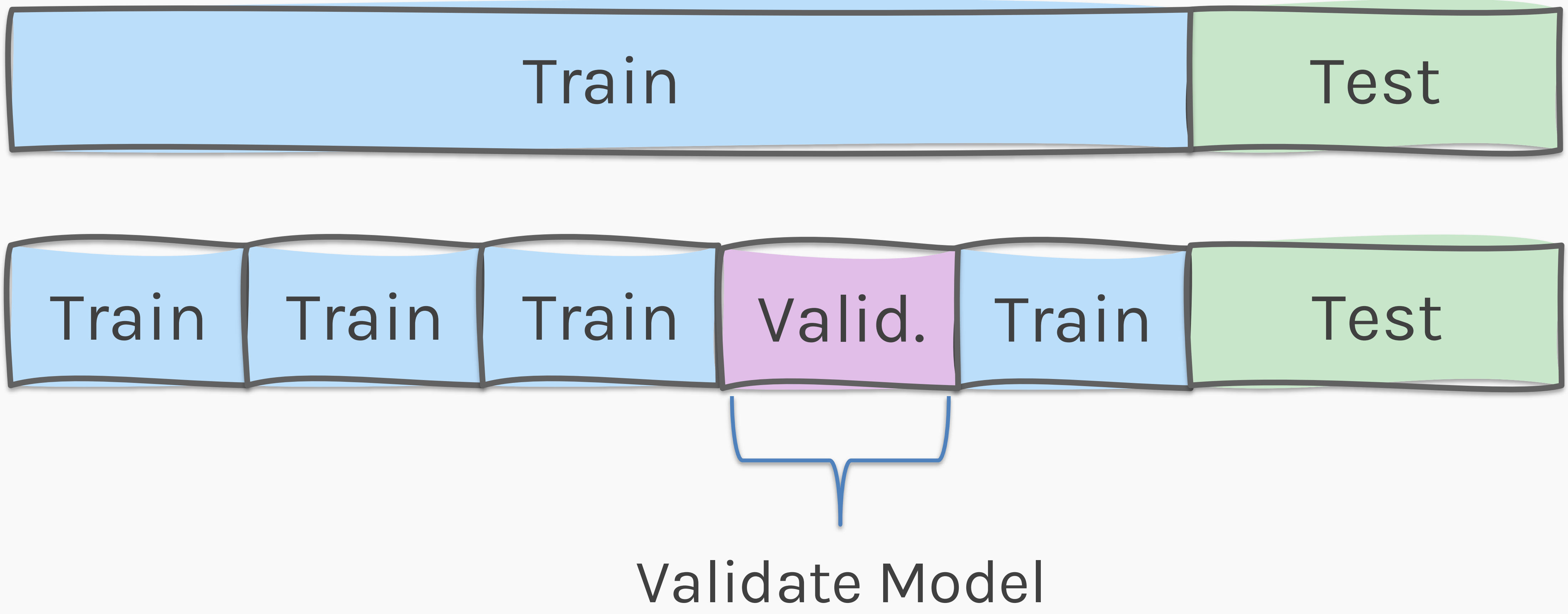
Cross Validation



Cross Validation

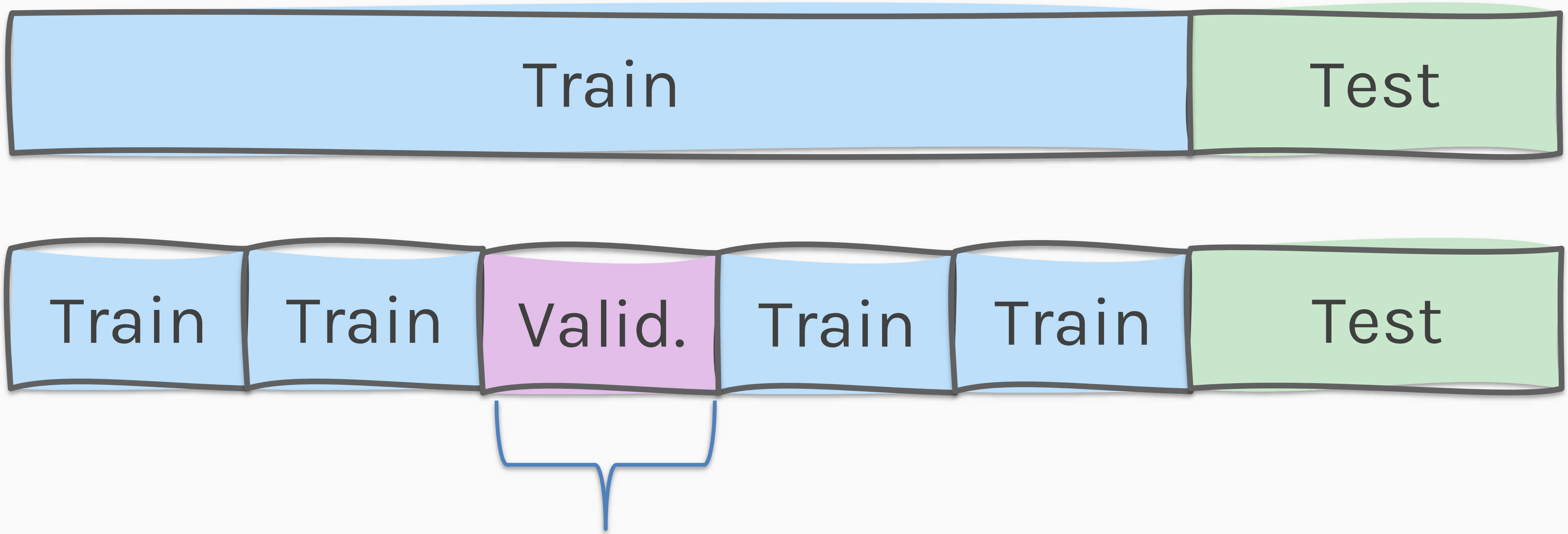


Cross Validation



MSE_2^{val}

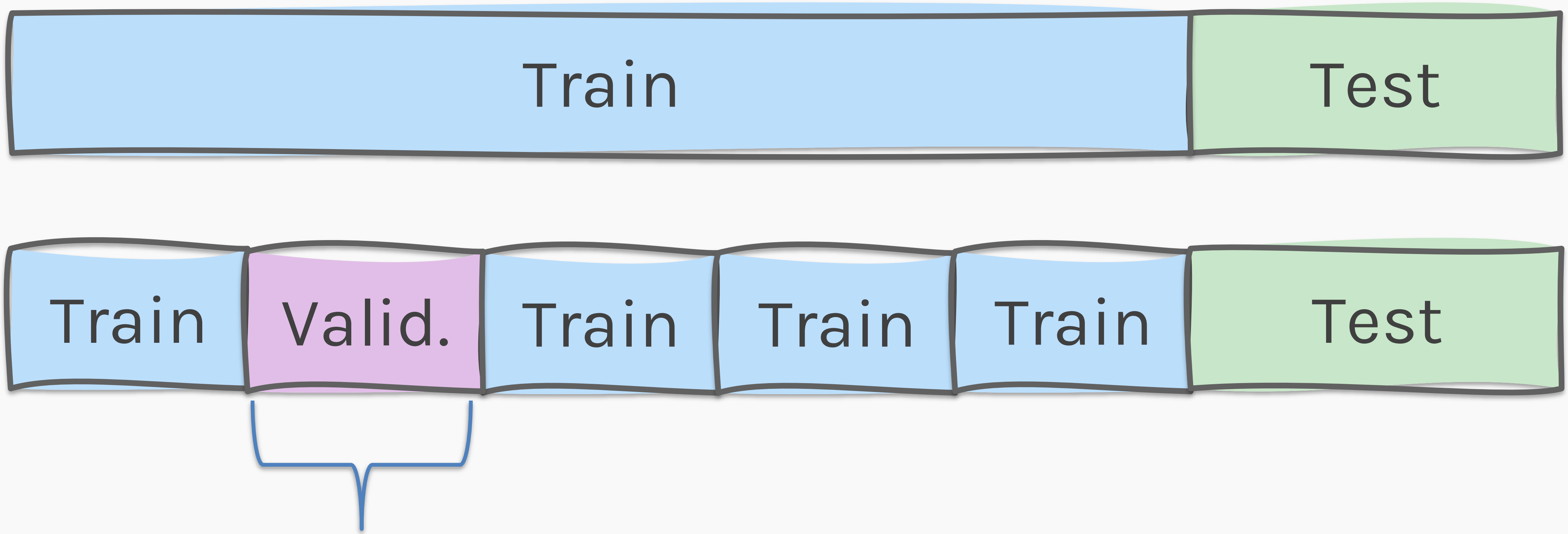
Cross Validation



Validate Model

MSE_3^{val}

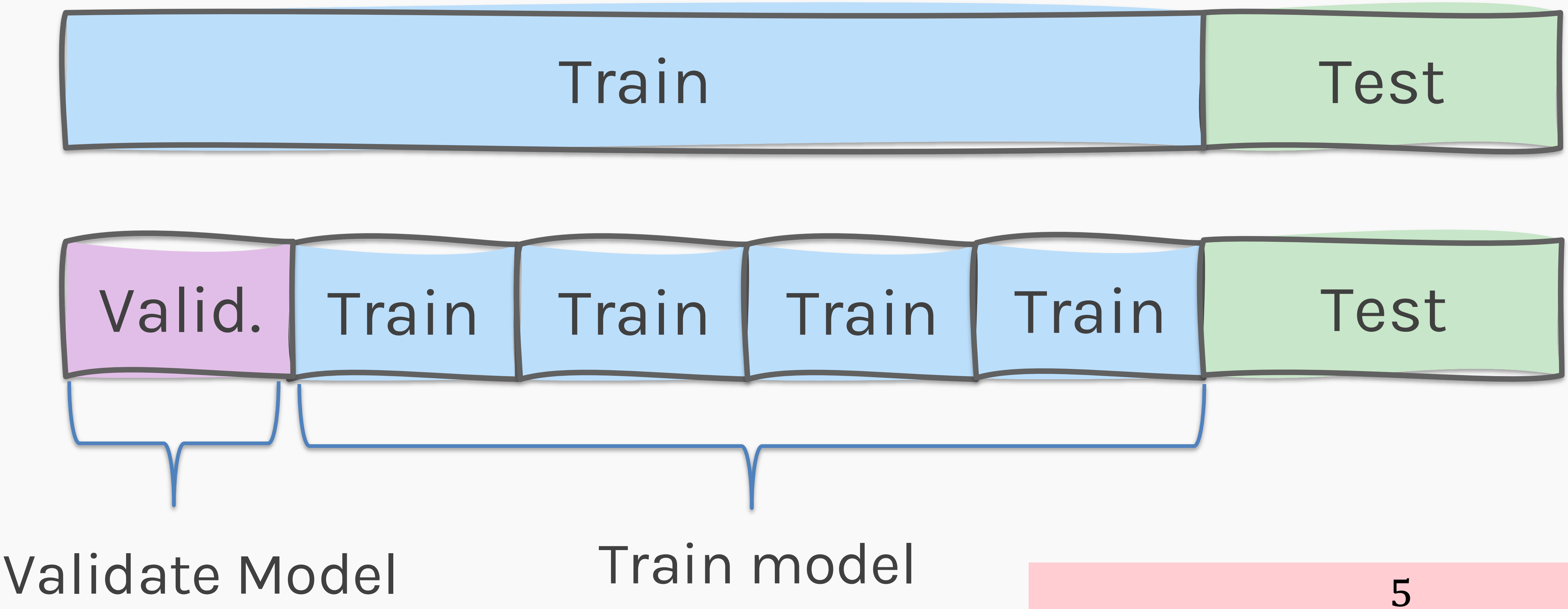
Cross Validation



Validate Model

$$MSE_4^{val}$$

Cross Validation



MSE_5^{val}

$$MSE^{val} = \frac{1}{5}\sum_{i=1}^{5} MSE_i^{val}$$

K-Fold Cross Validation

Given a data set $\{X_1, \ldots, X_n\}$, containing J features.

To ensure that every observation in the dataset is included in at least one training set and at least one validation set we use the ***K-fold validation*** :

- split the data into K uniformly sized chunks, $\{C_1, \ldots, C_K\}$
- we create K number of training/validation splits, using one of the K chunks for validation and the rest for training.

We fit the model on each training set, denoted $\hat{f}_{C_{-i}}$, and evaluate it on the corresponding validation set, $\hat{f}_{C_{-i}}(C_i)$. The ***cross validation is the performance*** of the model averaged across all validation sets:

$$CV(\text{Model}) = \frac{1}{K}\sum_{i=1}^{K} L(\hat{f}_{C_{-i}}(C_i))$$

where L is a loss function.

C_{-i} Represents the dataset after removing the *i-th* observation.

Leave-One-Out

Or using the ***leave one out*** method:

- validation set: $\{X_i\}$
- training set: $X_{-i} = \{X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n\}$

for $i = 1, \dots, n$:

We fit the model on each training set, denoted $\hat{f}_{X_{-i}}$, and evaluate it on the corresponding validation set, $\hat{f}_{X_{-i}}(X_i)$.

The ***cross validation score*** is the performance of the model averaged across all validation sets:

$$CV(\text{Model}) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{f}_{X_{-i}}(X_i))$$

where *L* is a loss function.

The model used to fit the data

Dictionary containing several metrics about the evaluation

The data to fit

The target variable to predict on

```
cv_results =
sklearn.model_selection.cross_validate(estimator, X, y,
             scoring, cv, return_train_score)
```

Number of folds

Strategy to evaluate the performance of the cross-validated model on the test set.
Use “neg_mean_squared_error” for regression

Set to True to include train scores

Dictionary containing several
metrics about the evaluation

The model us

iable to

```
cv_results =
sklearn.model_selection.c                       or, X, y,
              scoring,      eturn_train_score)
```

scikit-learn's cross-validation
framework is designed to maximize a
scoring function.
In the case of regression problems,
however, we usually want to minimize
the error (such as mean squared error
or MSE), not maximize it.

Number of folds

Strategy to evaluate the performance of the
cross-validated model on the test set.

Set to True to include
train scores

Use “neg_mean_squared_error” for
regression


El Magnifico