Leveraging Different Validation Strategies in Data Modeling

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

IID (Independent and Identically Distributed) data

Cross-validation to build robust models

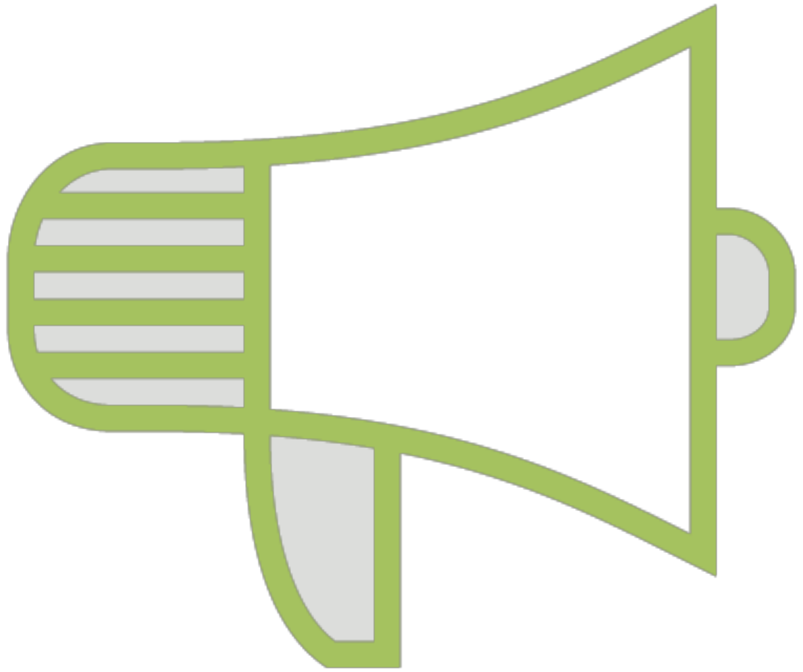
Iterative K-fold cross-validation

Repeated K-fold cross-validation

Stratified cross-validation

Grouped cross-validation

IID Assumption



Usually, points in a data set are assumed to be

- *I*ndependent of each other
- *I*dentically *D*istributed, i.e. similar to each other in statistical properties

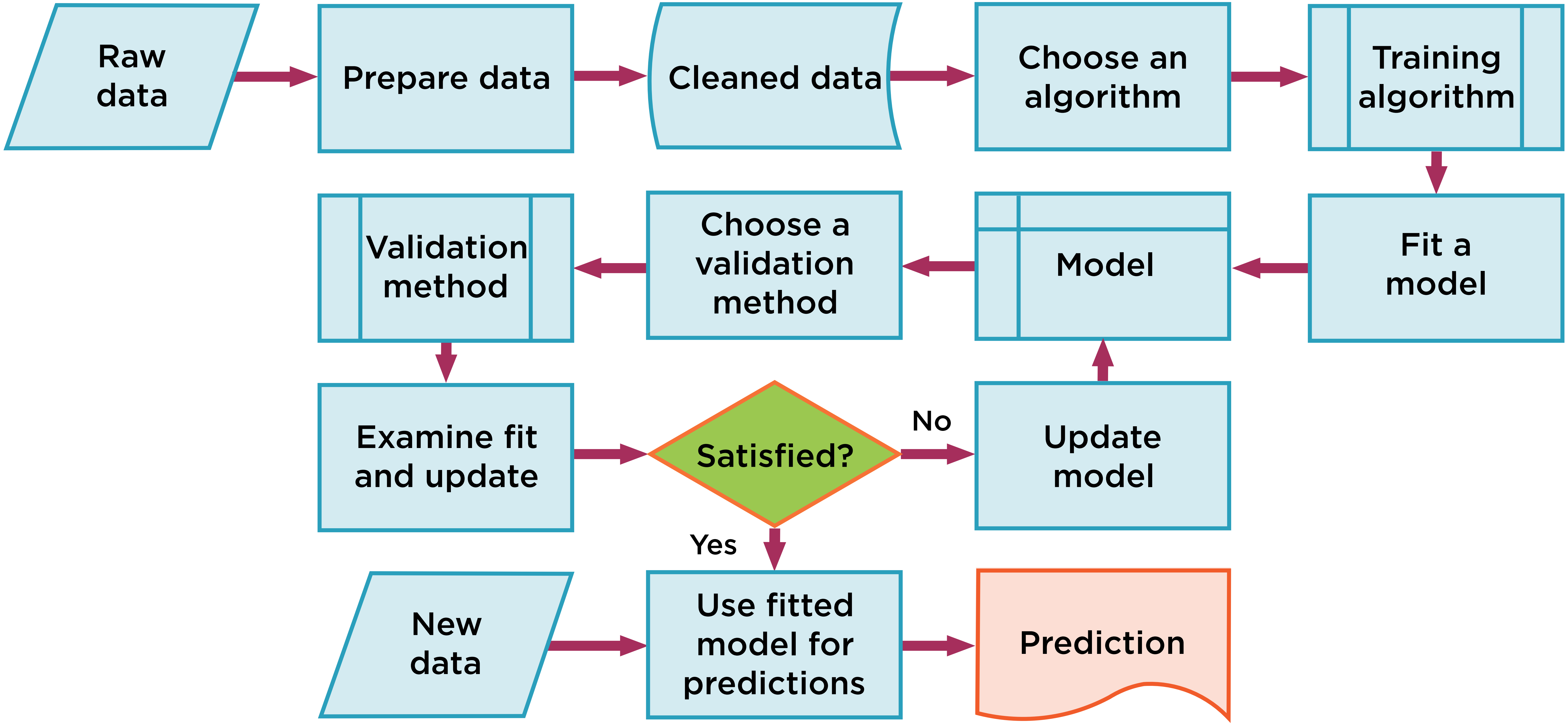
The IID assumption is an important one, implicit in the training of virtually all ML models

Cross-validation

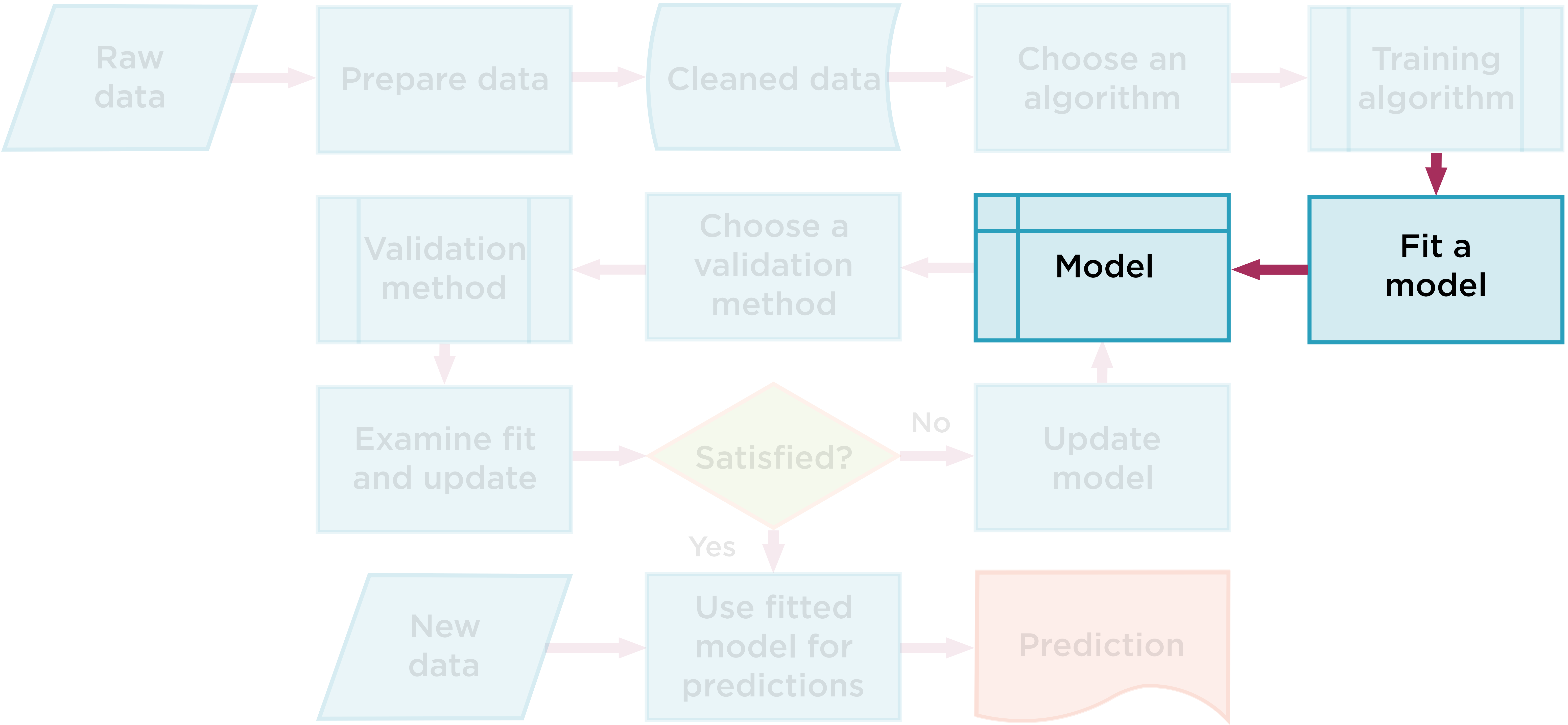
Cross-validation

Model validation technique to assess how the results of a statistical analysis will generalize to an independent dataset - helps determine how well a model will perform in practice

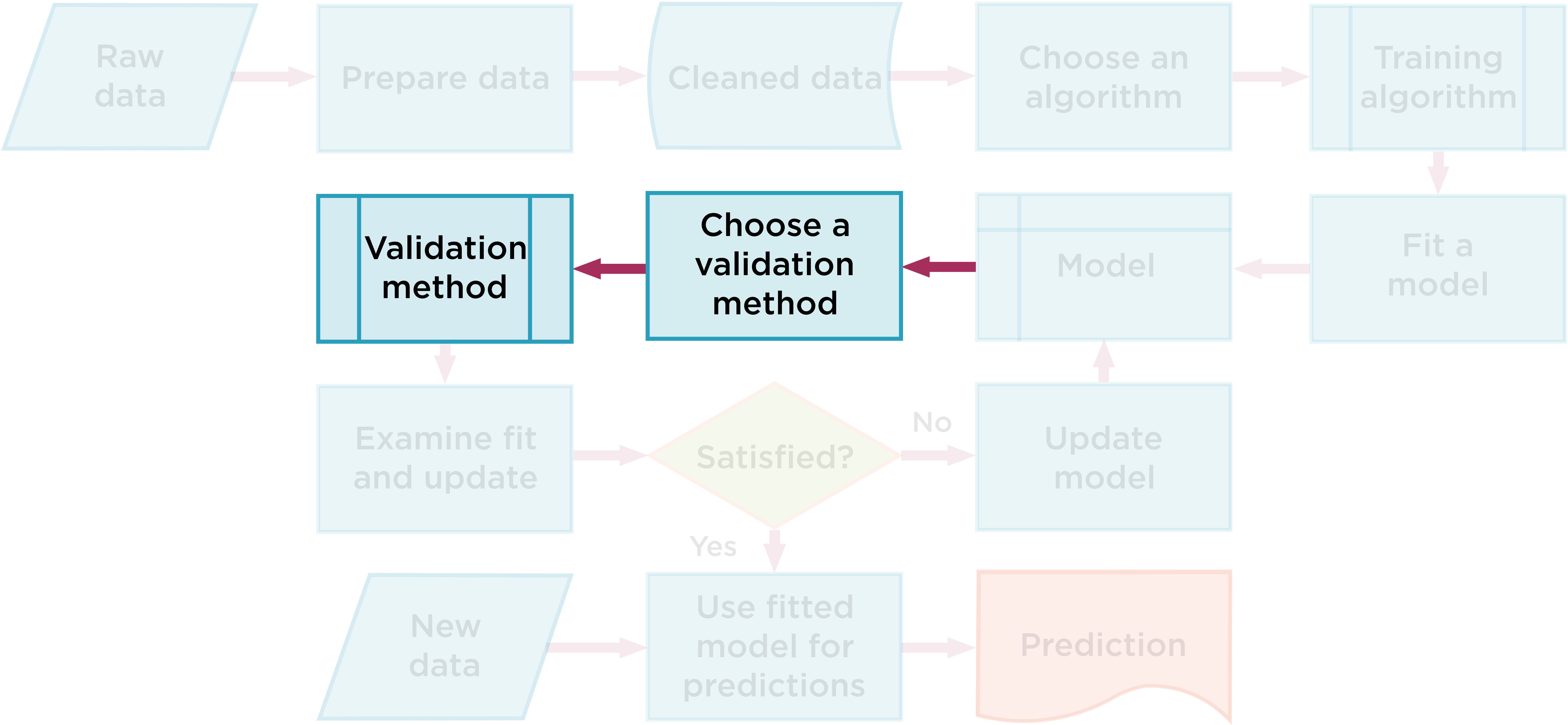
Basic Machine Learning Workflow



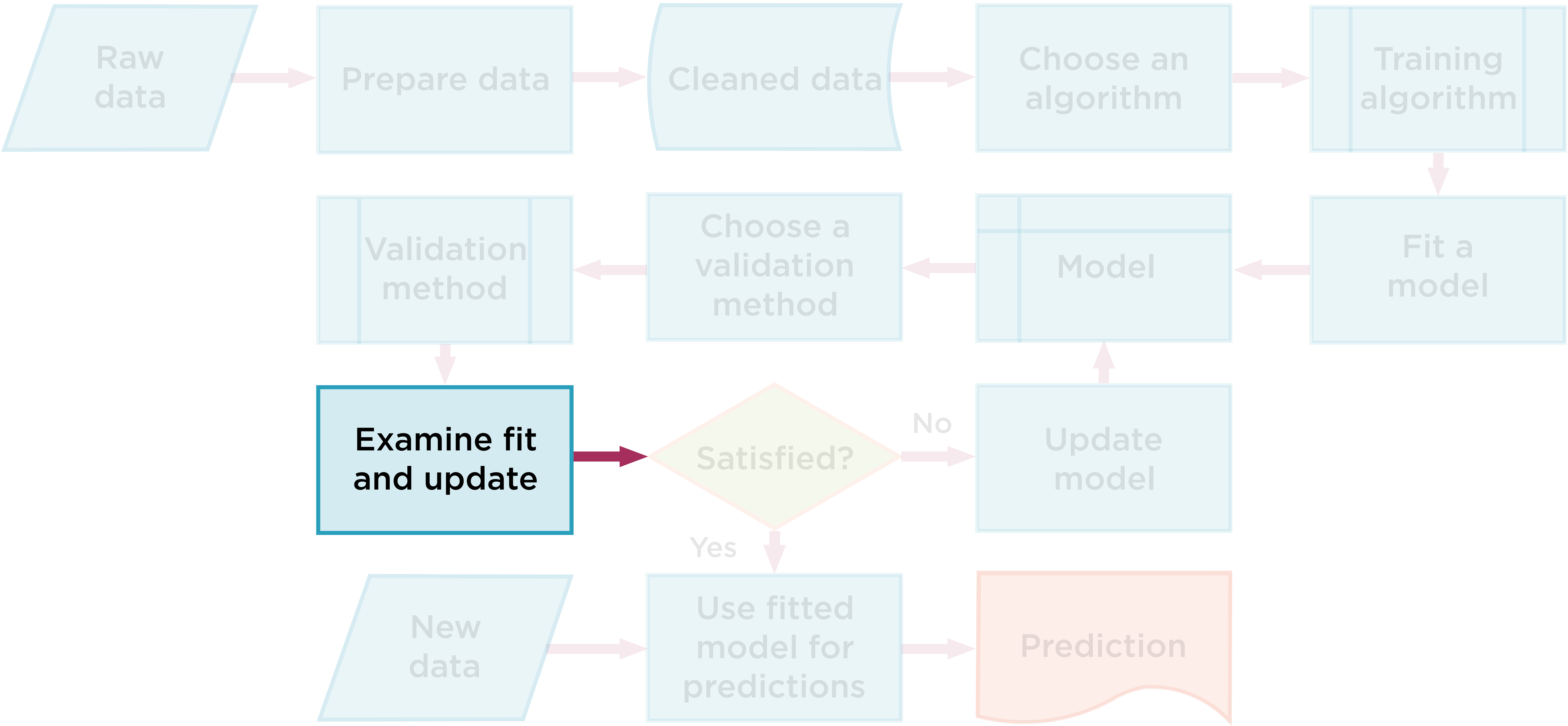
Training to Find Model Parameters



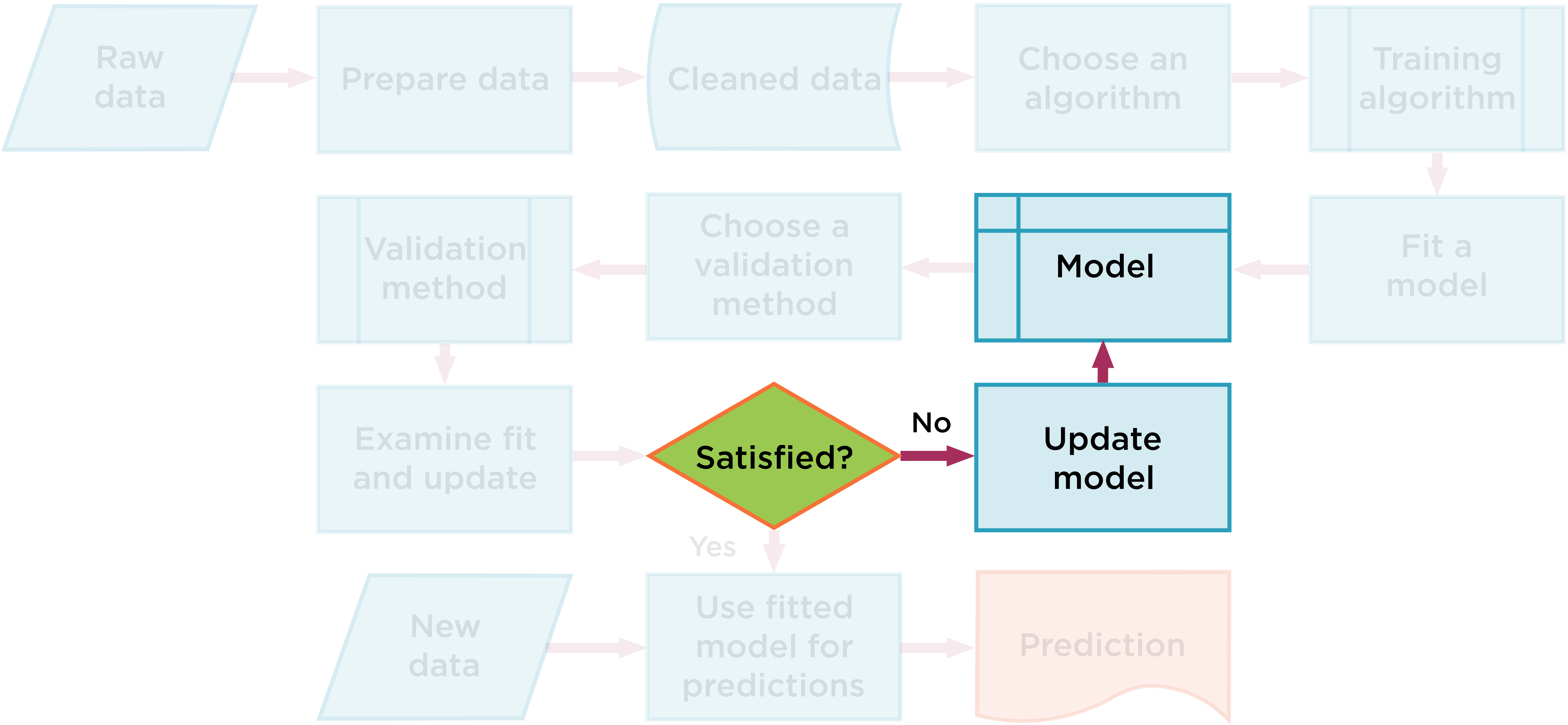
Evaluate the Model



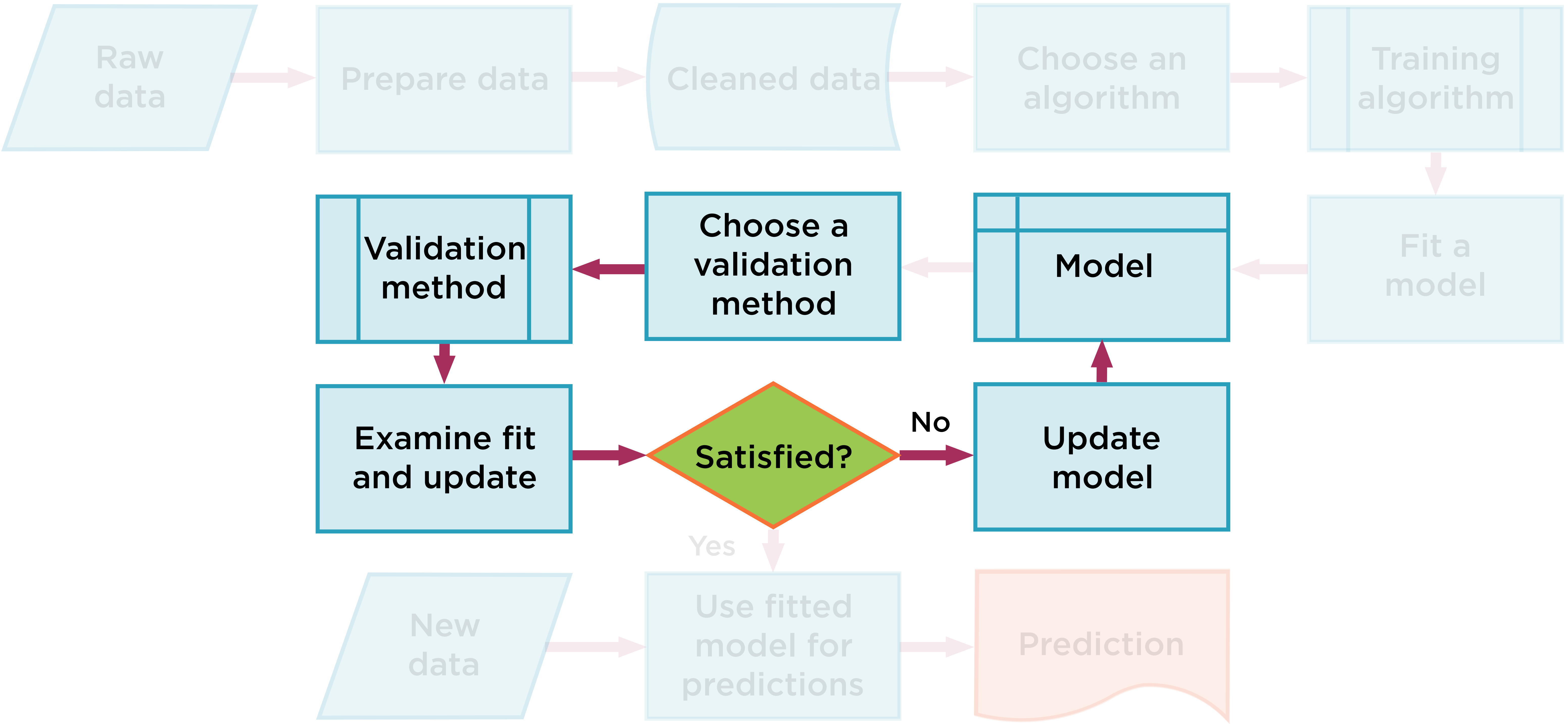
Score the Model



Different Algorithm, More Data, More Training?

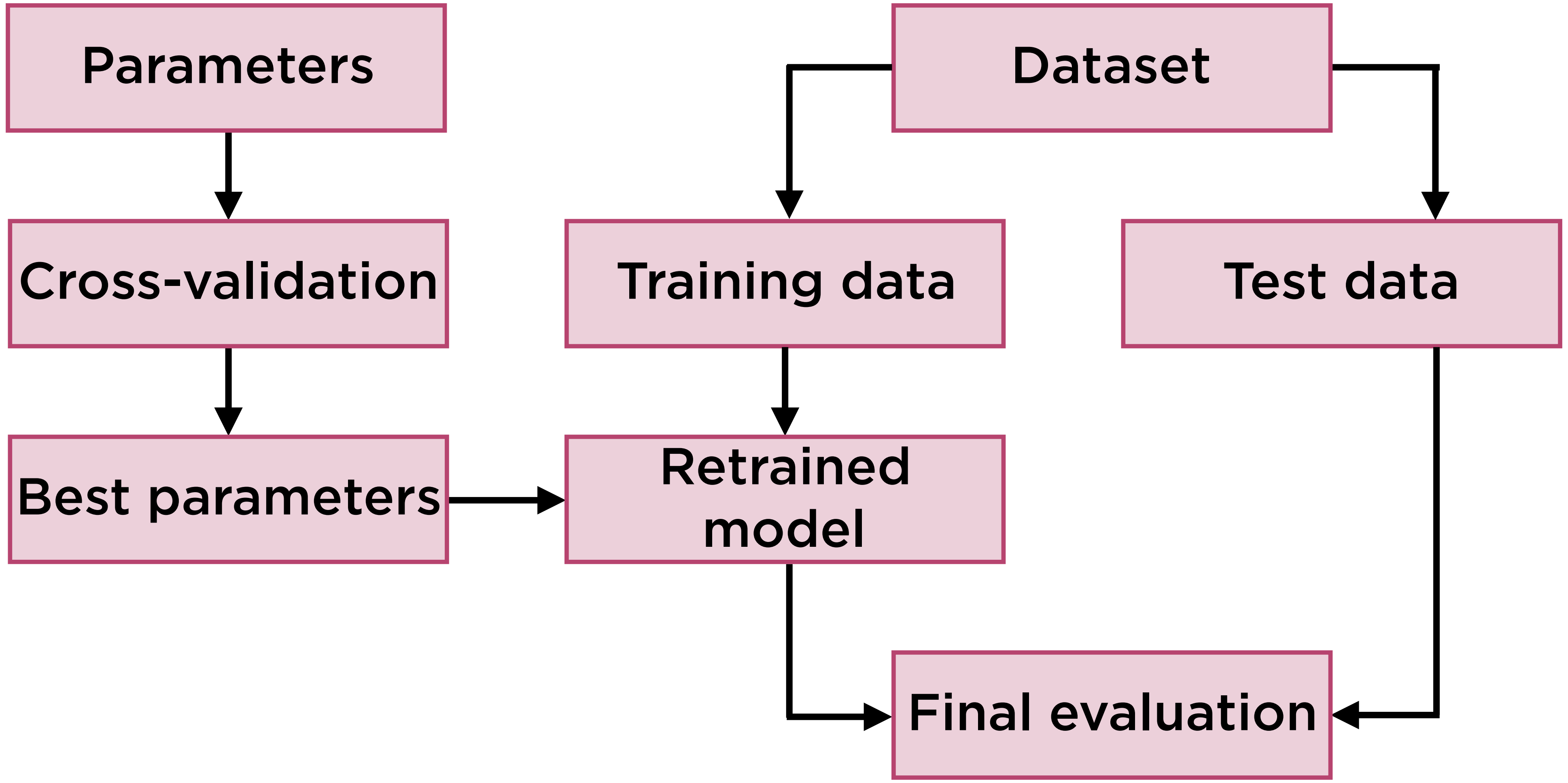


Iterate Till Model Finalized

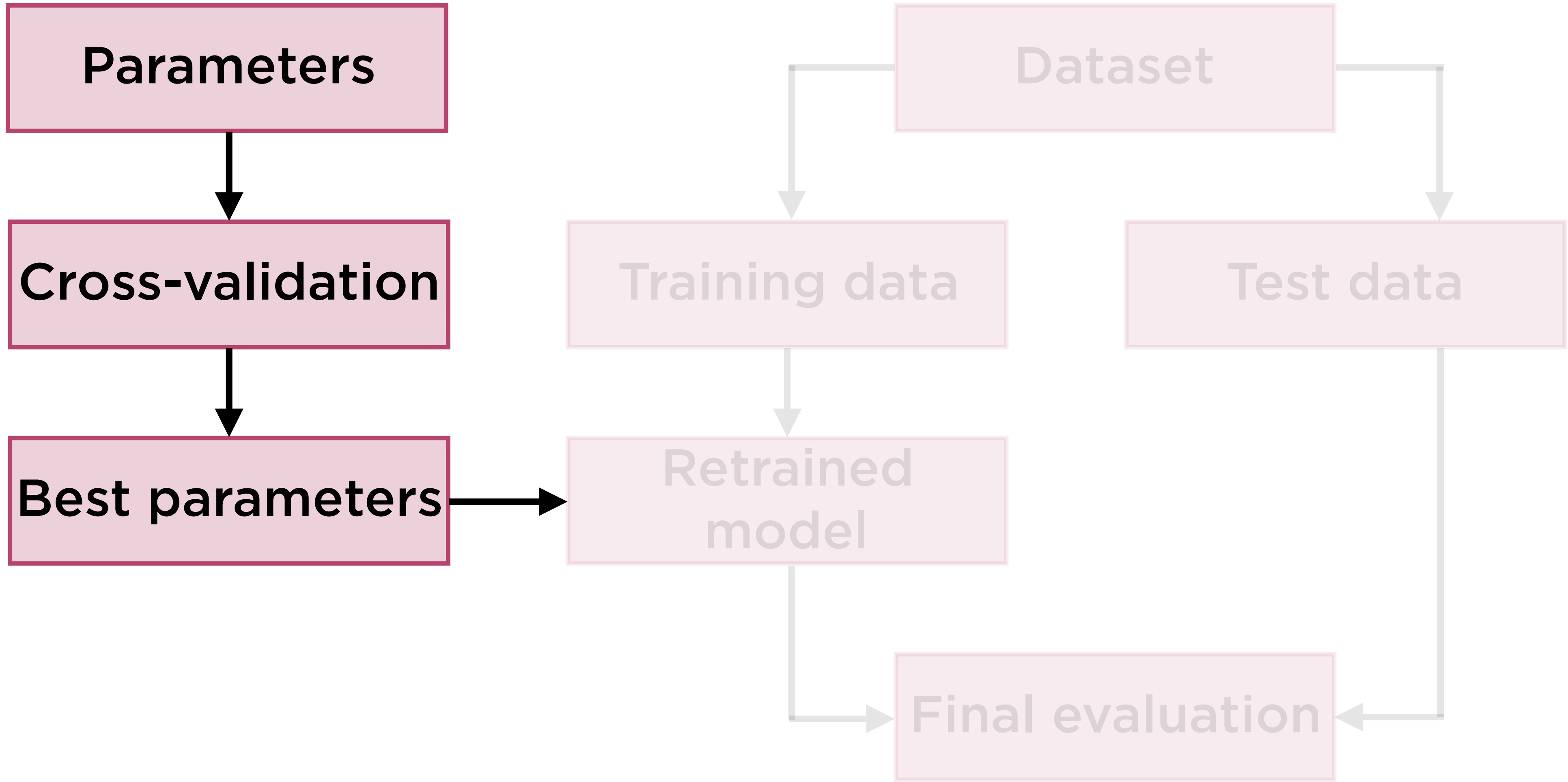


Singular Cross-validation

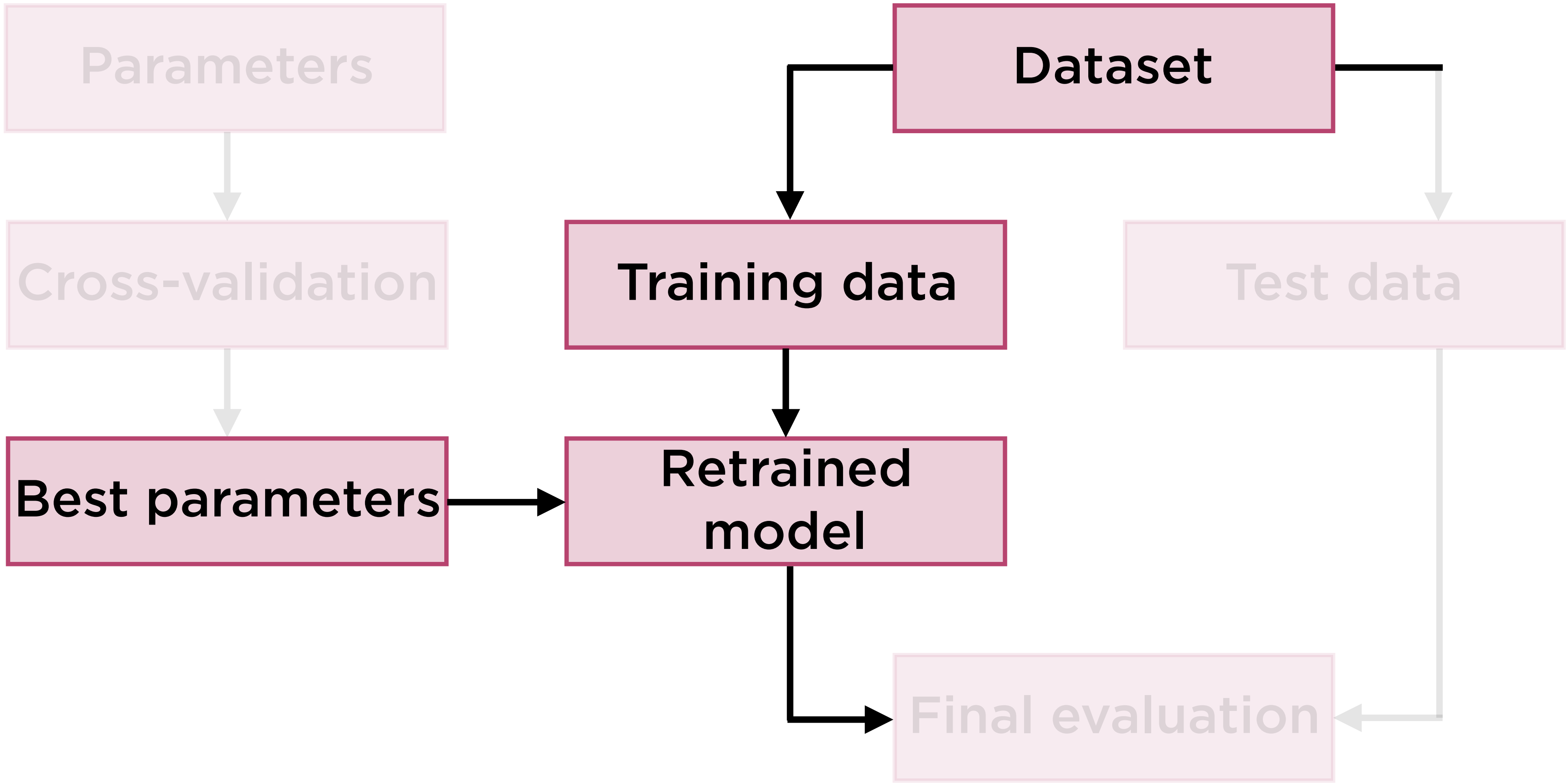
Evaluating Model Performance



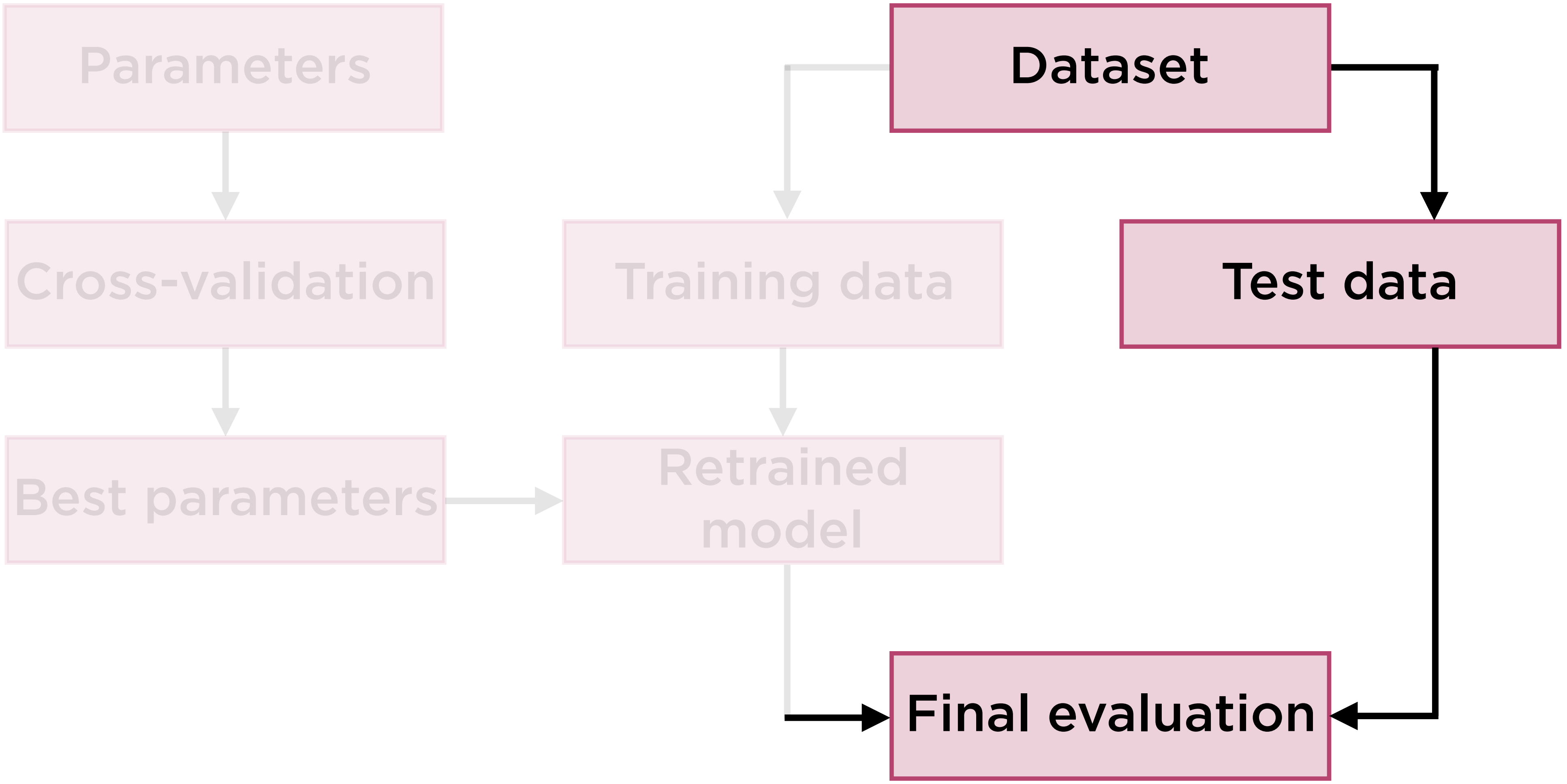
Refine Original Model Parameters



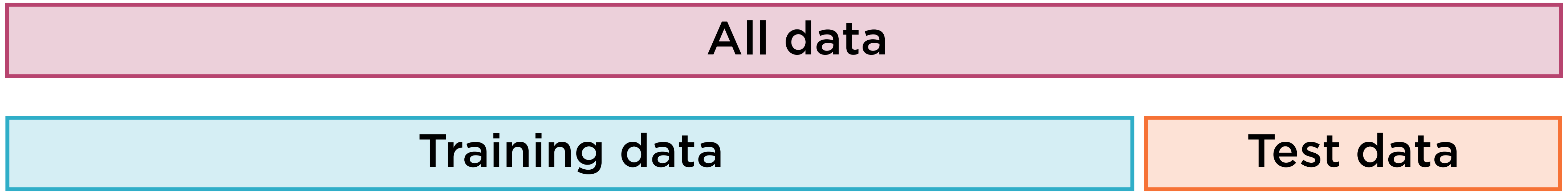
Build Model with Best Parameters



Final Evaluation on Test Data

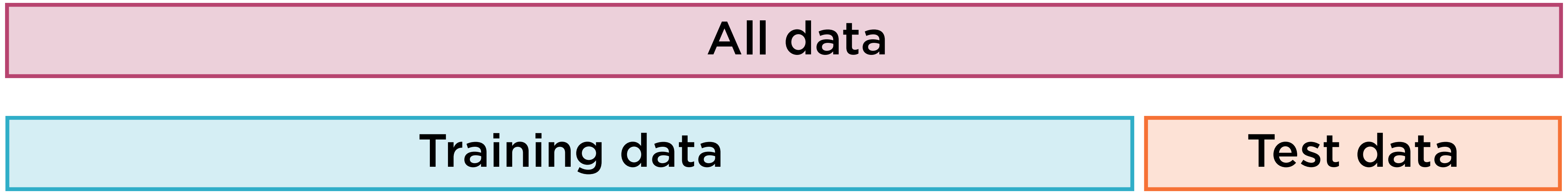


No Validation



Just one candidate model - train it using Training Data, sanity-check it using Test Data

No Validation

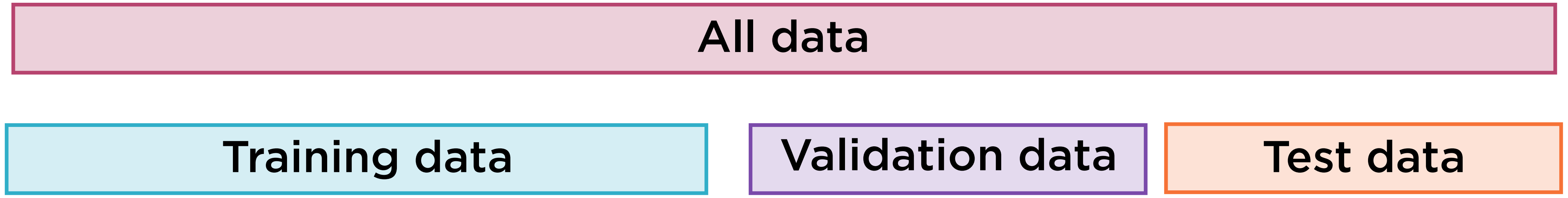


For N candidate models, run N training and N test processes

Overfitting on Test Set

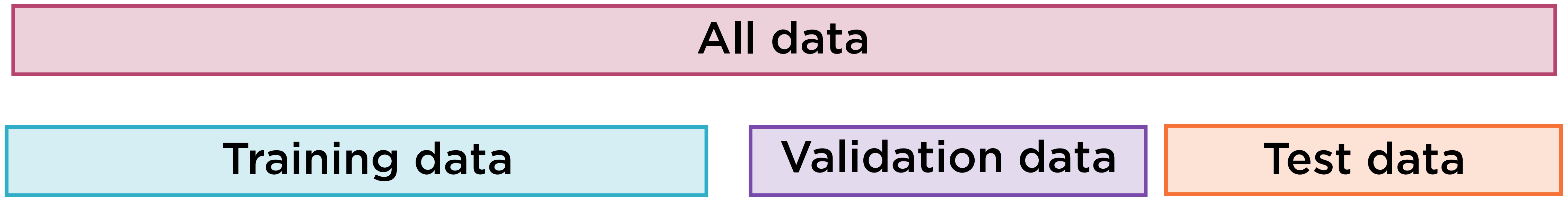
Choosing best candidate model on the Test Set leads to this form of overfitting. Occurs when data is split into just two sets: Training and test.

Singular Cross-validation



Now can have multiple candidate models, and select the best one - Hyperparameter Tuning

Singular Cross-validation

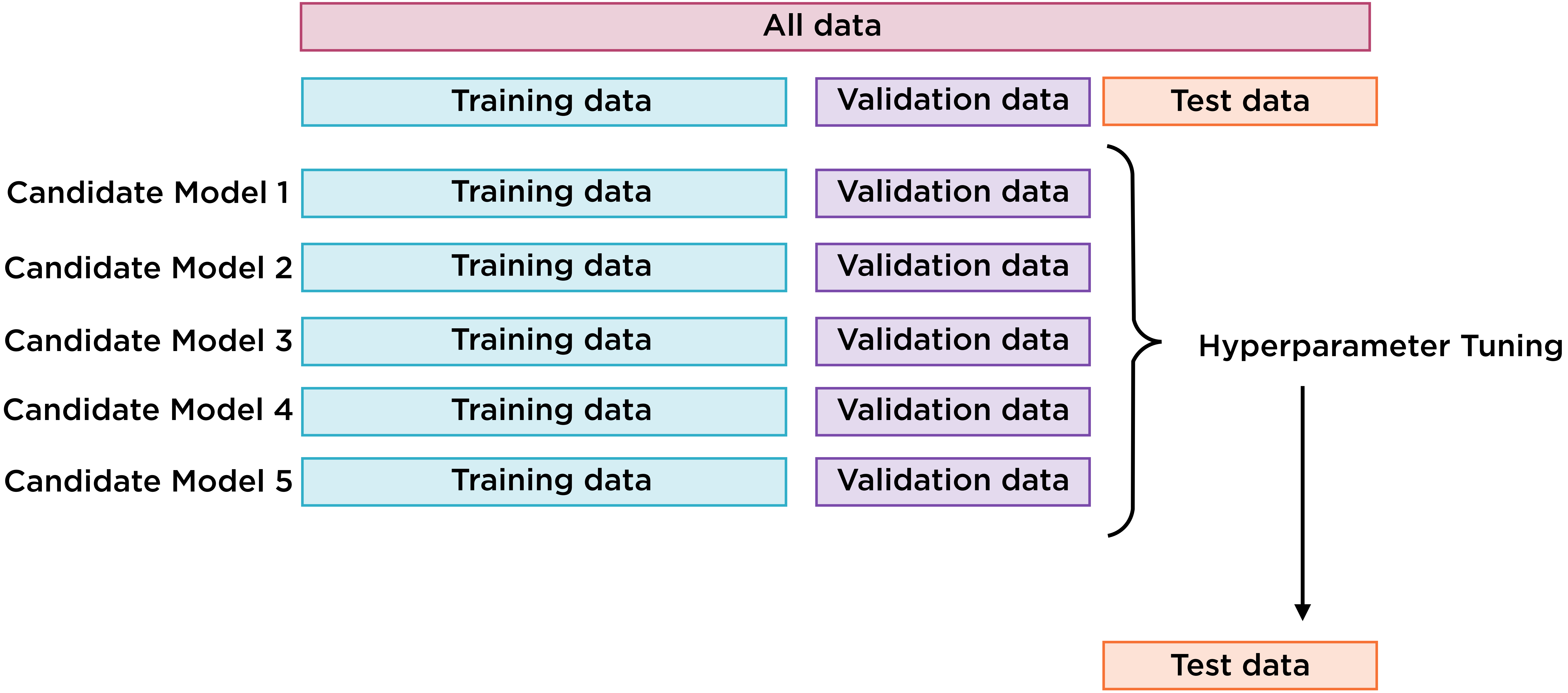


For N candidate models, run N training and N validation processes but just 1 test process

Cross-validation

Carve out a separate validation set of data points; use this to evaluate different candidate models. Data now split into three sets: Training, validation and test.

Singular Cross-validation



Demo

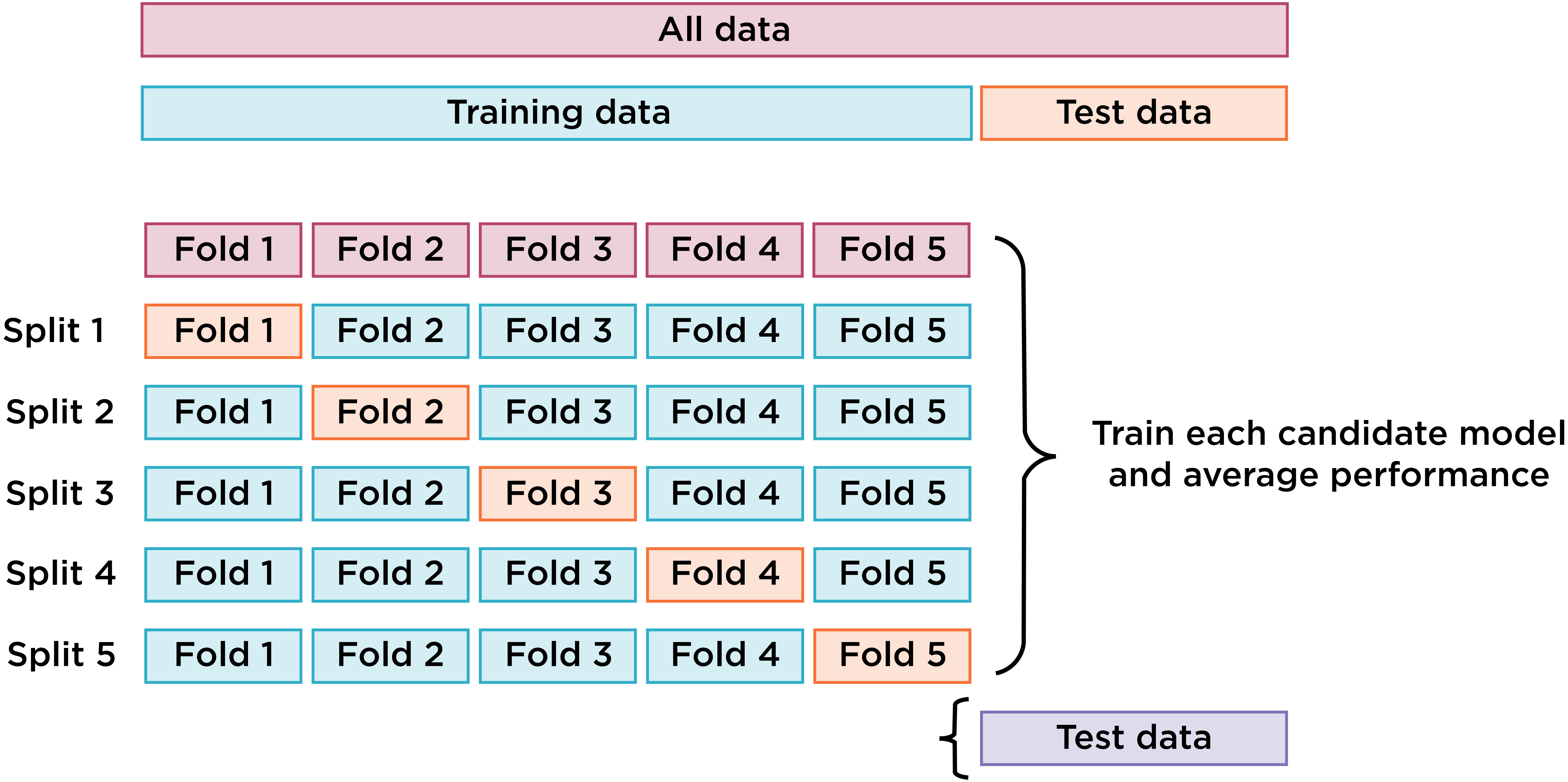
Cross-validating models using Azure ML Studio

K-fold Cross-validation and Variants

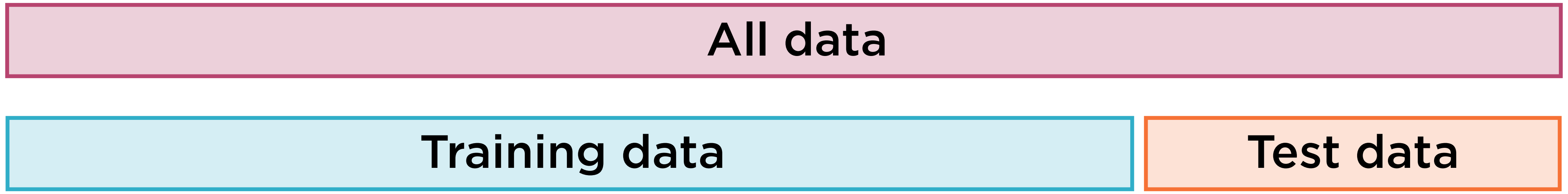
K-fold Cross-validation

For each candidate model, repeatedly train, and validate using different subsets of training data. Much more computationally intensive, but very robust - does not “waste” data.

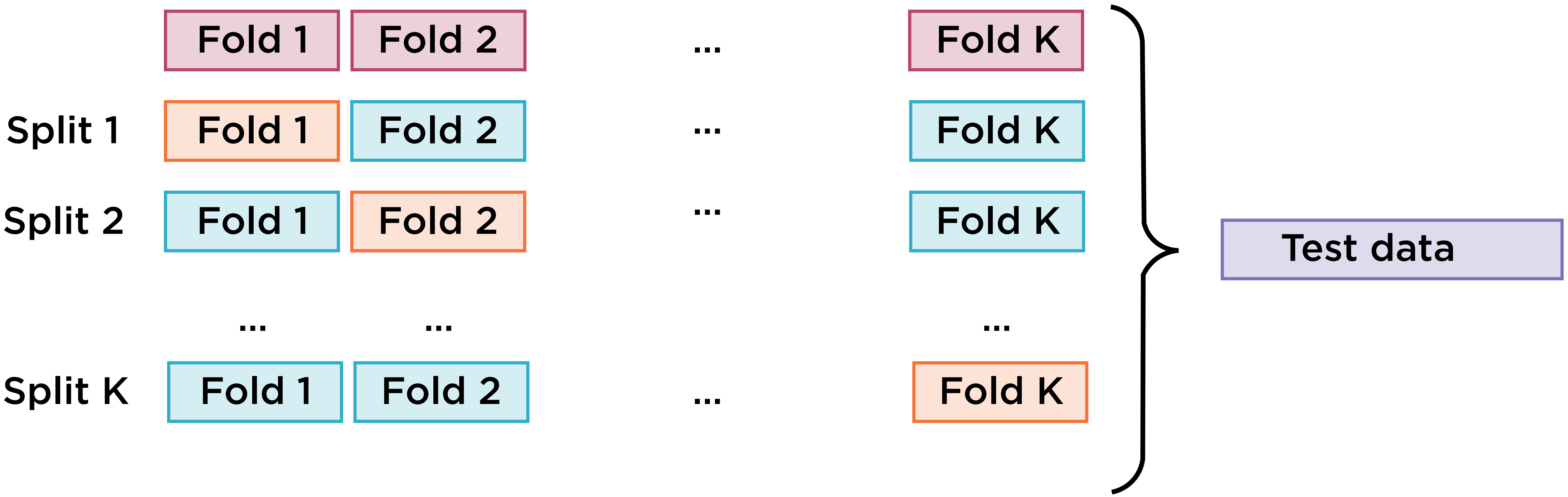
K-fold Cross-validation



K-fold Cross-validation



For N candidate models and K folds, run N*K training and N validation processes but just 1 test process



Shuffle and Split

Shuffle the points in the data set before splitting into training and test splits; helps ensure test points are picked from throughout the data set.

Group and Class

Class

Group

Group and Class

Class refers to output label (e.g. “Male” and “Female”)

Group refers to measurement group (e.g. “Before” and “After”)

Variants of K-fold

Repeated K-fold

Leave-one-out

Leave-P-out

3.1.2.1.2. Repeated K-Fold

`RepeatedKFold` repeats K-Fold n times. It can be used when one requires to run `KFold` n times, producing different splits in each repetition.

Example of 2-fold K-Fold repeated 2 times:

```
>>> import numpy as np
>>> from sklearn.model_selection import RepeatedKFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> random_state = 12883823
>>> rkf = RepeatedKFold(n_splits=2, n_repeats=2, random_state=random_state)
>>> for train, test in rkf.split(X):
...     print("%s %s" % (train, test))
...
[2 3] [0 1]
[0 1] [2 3]
[0 2] [1 3]
[1 3] [0 2]
```

Similarly, `RepeatedStratifiedKFold` repeats Stratified K-Fold n times with different randomization in each repetition.

3.1.2.1.3. Leave One Out (LOO)

`LeaveOneOut` (or LOO) is a simple cross-validation. Each learning set is created by taking all the samples except one, the test set being the sample left out. Thus, for n samples, we have n different training sets and n different tests set. This cross-validation procedure does not waste much data as only one sample is removed from the training set:

```
>>> from sklearn.model_selection import LeaveOneOut

>>> X = [1, 2, 3, 4]
>>> loo = LeaveOneOut()
>>> for train, test in loo.split(X):
...     print("%s %s" % (train, test))
[1 2 3] [0]
[0 2 3] [1]
[0 1 3] [2]
[0 1 2] [3]
```

Potential users of LOO for model selection should weigh a few known caveats. When compared with k-fold cross validation, one builds n models from n samples instead of k models, where $n > k$. Moreover, each is trained on $n - 1$ samples rather than $(k-1)n/k$. In both ways, assuming k is not too large and $k < n$, LOO is more computationally expensive than k-fold cross validation.

In terms of accuracy, LOO often results in high variance as an estimator for the test error. Intuitively, since $n - 1$ of the n samples are used to build each model, models constructed from folds are virtually identical to each other and to the model built from the entire training set.

However, if the learning curve is steep for the training size in question, then 5- or 10- fold cross validation can overestimate the generalization error.

As a general rule, most authors, and empirical evidence, suggest that 5- or 10- fold cross validation should be preferred to LOO.

3.1.2.1.4. Leave P Out (LPO)

`LeavePOut` is very similar to `LeaveOneOut` as it creates all the possible training/test sets by removing p samples from the complete set. For n samples, this produces $\binom{n}{p}$ train-test pairs. Unlike `LeaveOneOut` and `KFold`, the test sets will overlap for $p > 1$.

Example of Leave-2-Out on a dataset with 4 samples:

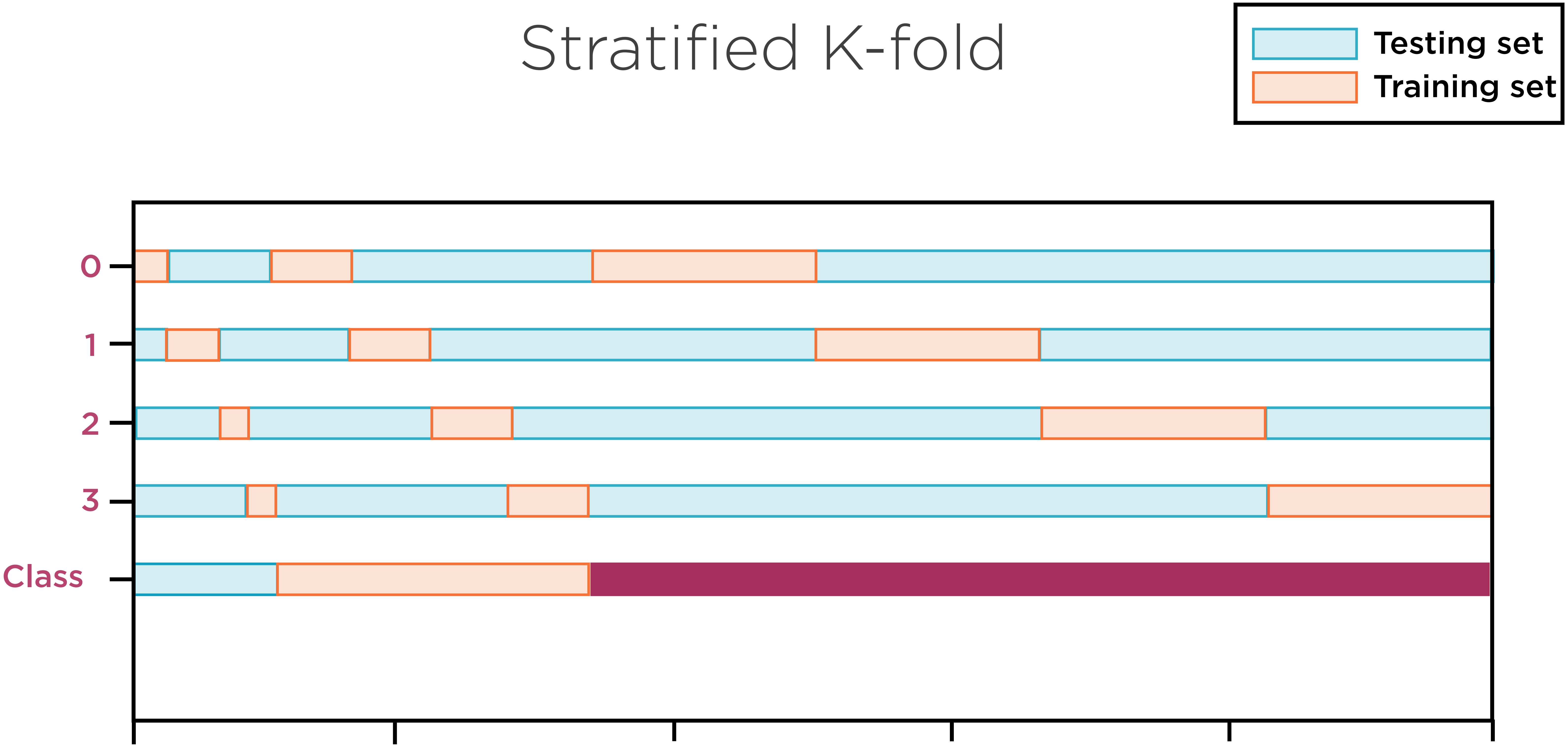
```
>>> from sklearn.model_selection import LeavePOut

>>> X = np.ones(4)
>>> lpo = LeavePOut(p=2)
>>> for train, test in lpo.split(X):
...     print("%s %s" % (train, test))
[2 3] [0 1]
[1 3] [0 2]
[1 2] [0 3]
[0 3] [1 2]
[0 2] [1 3]
[0 1] [2 3]
```

K-fold cross-validation does not take into account either group or class while splitting data

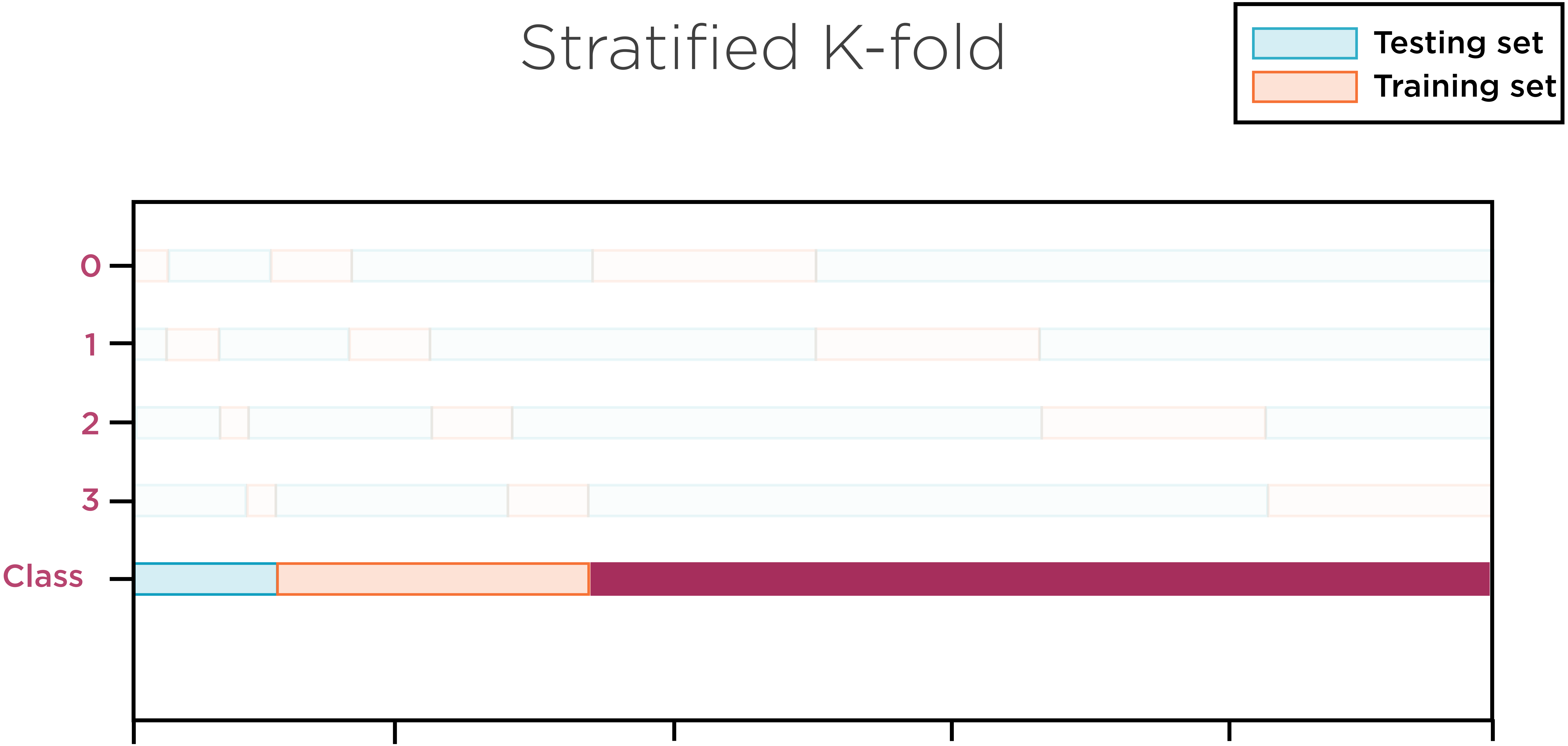
Stratified K-fold

Ensure that each fold has a representation of different classes (e.g. each fold has approximately representative mix of “Male” and “Female”)

Stratified K-fold

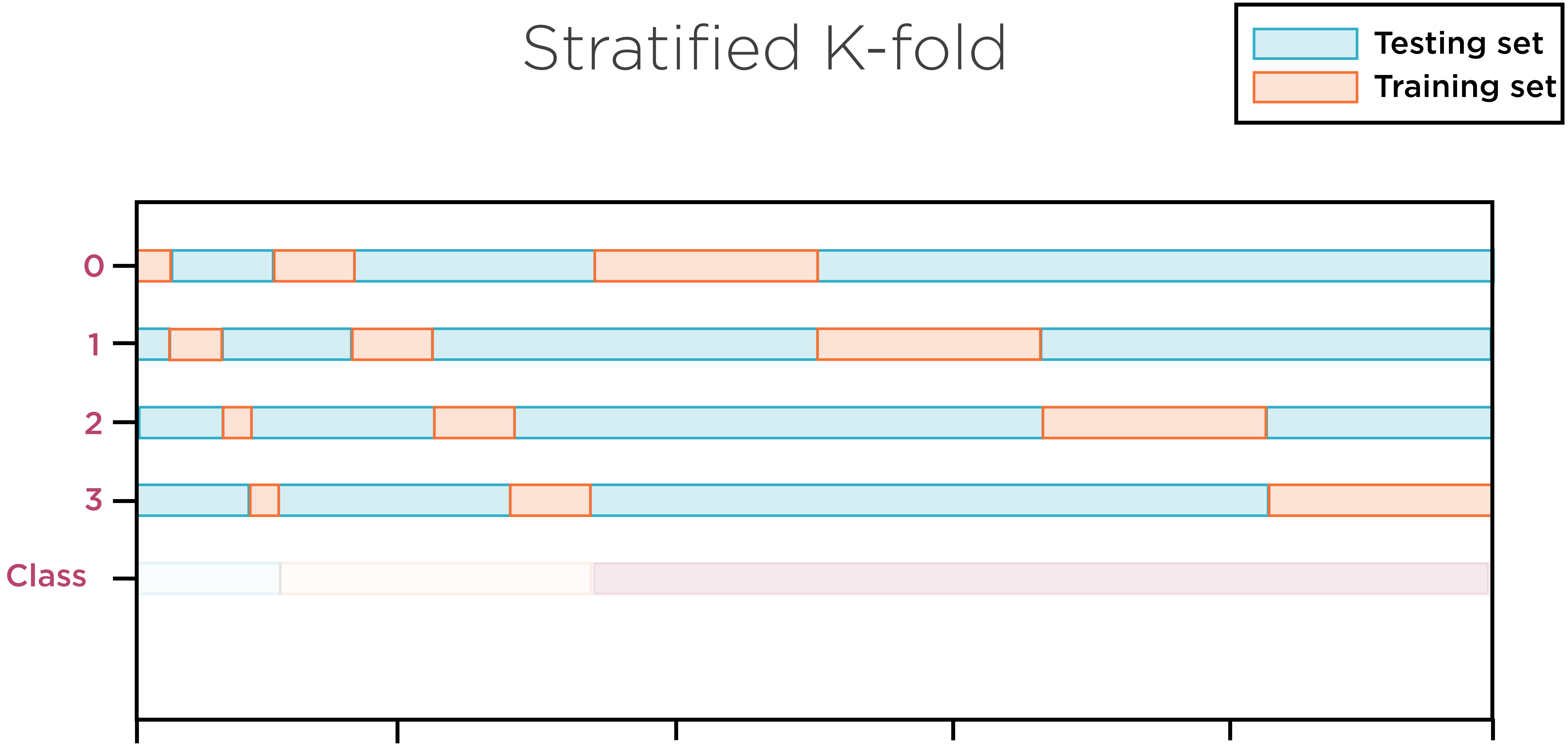


Stratified K-fold



Every record belongs one of three classes

Stratified K-fold



Each fold of data will contain records in the same proportion as the overall dataset

Stratified cross-validation techniques account for class but not for group

Grouped Data

Samples are not independent of each other, and must be identified as related to each other using group identifiers.

Grouped Data

Common in real-world scenarios

- Before/after tests
- Observations from sensors, some of which have known biases
- Multiple patients and multiple samples per patient

Grouped Data

IID Assumption: Data points are *I*ndependent, *I*dentically *D*istributed IID

Implicit assumption usually made in most modeling

No longer true once data are grouped

Grouped Data

Need special cross-validation procedures

Need to ensure that each group is either entirely in the training data

Or entirely in validation data

Group IDs should not cross fold boundaries

Demo

K-fold cross-validation

Demo

Repeated K-fold cross-validation

Demo

Stratified K-fold cross-validation

Demo

Group K-fold cross-validation

Summary

IID (Independent and Identically Distributed) data

Iterative K-fold cross-validation

Repeated K-fold cross-validation

Stratified cross-validation

Grouped cross-validation