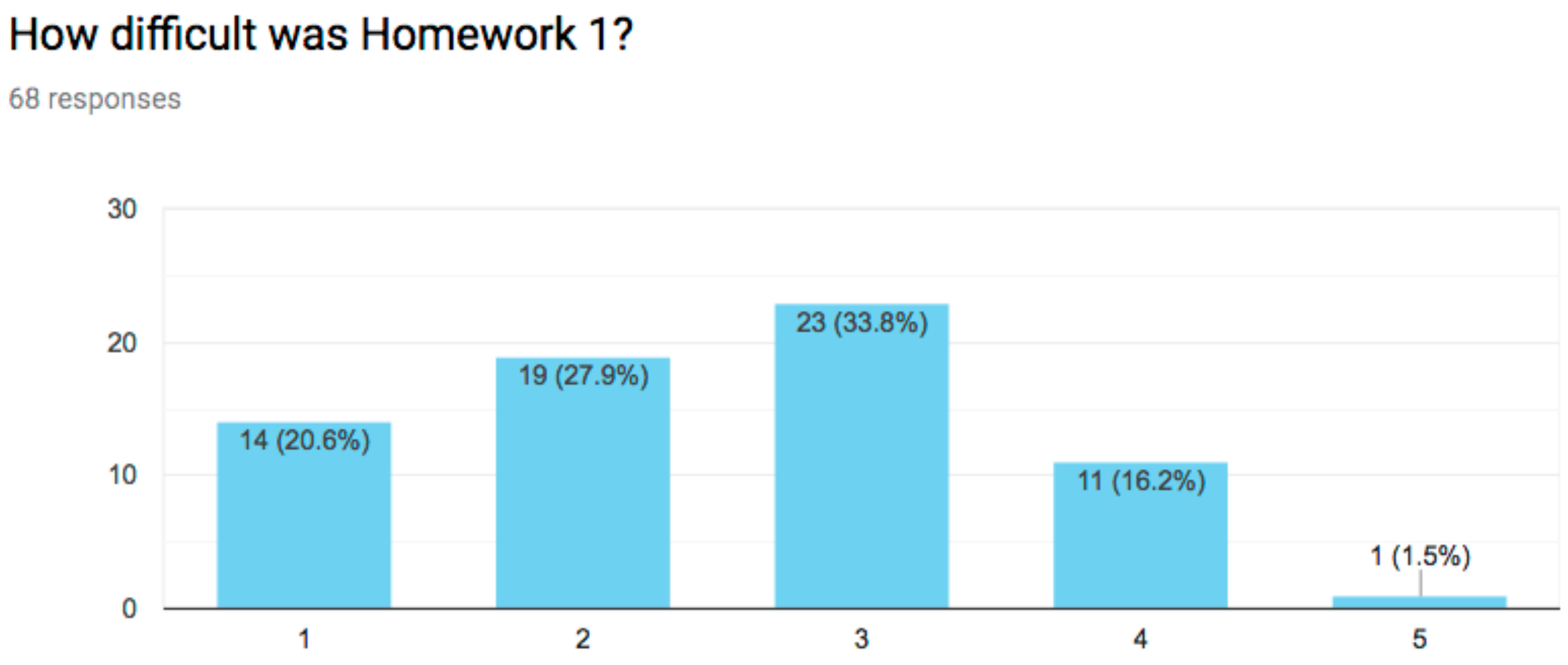
CME 250: Introduction to Machine Learning

Lecture 4: Cross-validation and Imputation

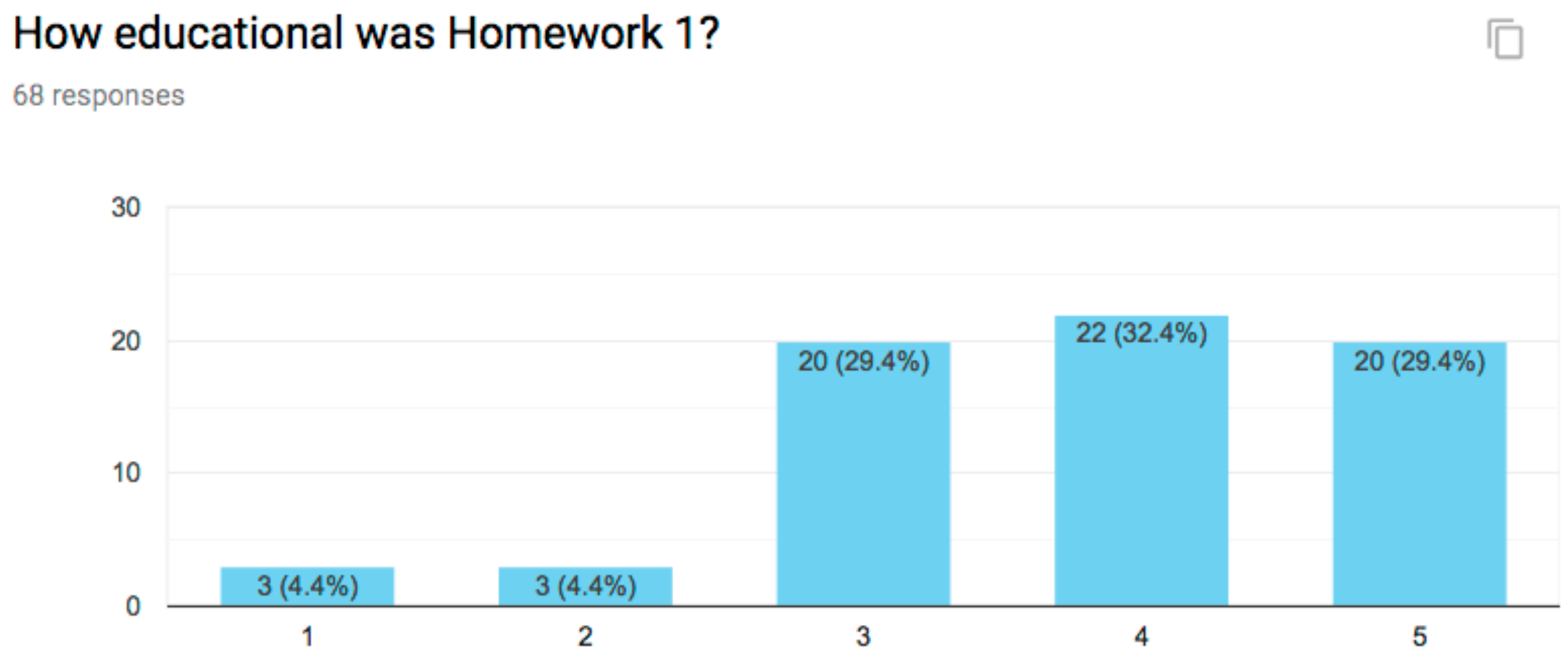
Sherrie Wang
sherwang@stanford.edu

Homework 1 Feedback



How difficult was Homework 1?

68 responses



How educational was Homework 1?

68 responses

Agenda

Slides are online at
cme250.stanford.edu

- Cross-validation
- Missing data
  - Missing completely at random (MCAR)
  - Imputation methods

Cross-Validation

Recall the Validation & Test Sets

Goals:

- Pick the best model
- Estimate the average error on new, unseen data

To do this, we *hold out* a part of the dataset and apply the trained model to these held out samples.

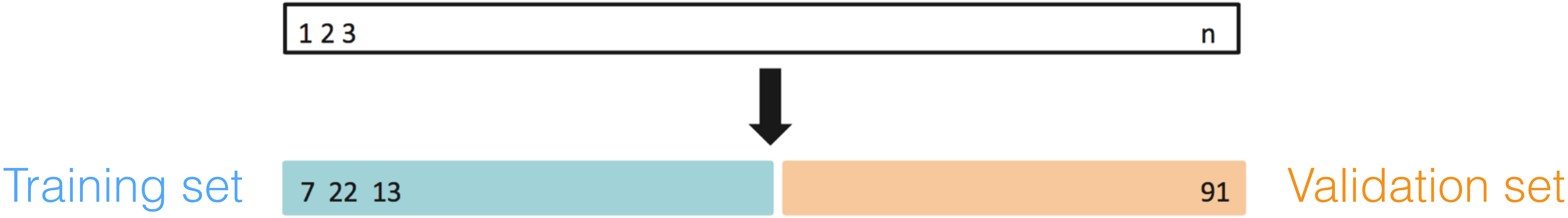


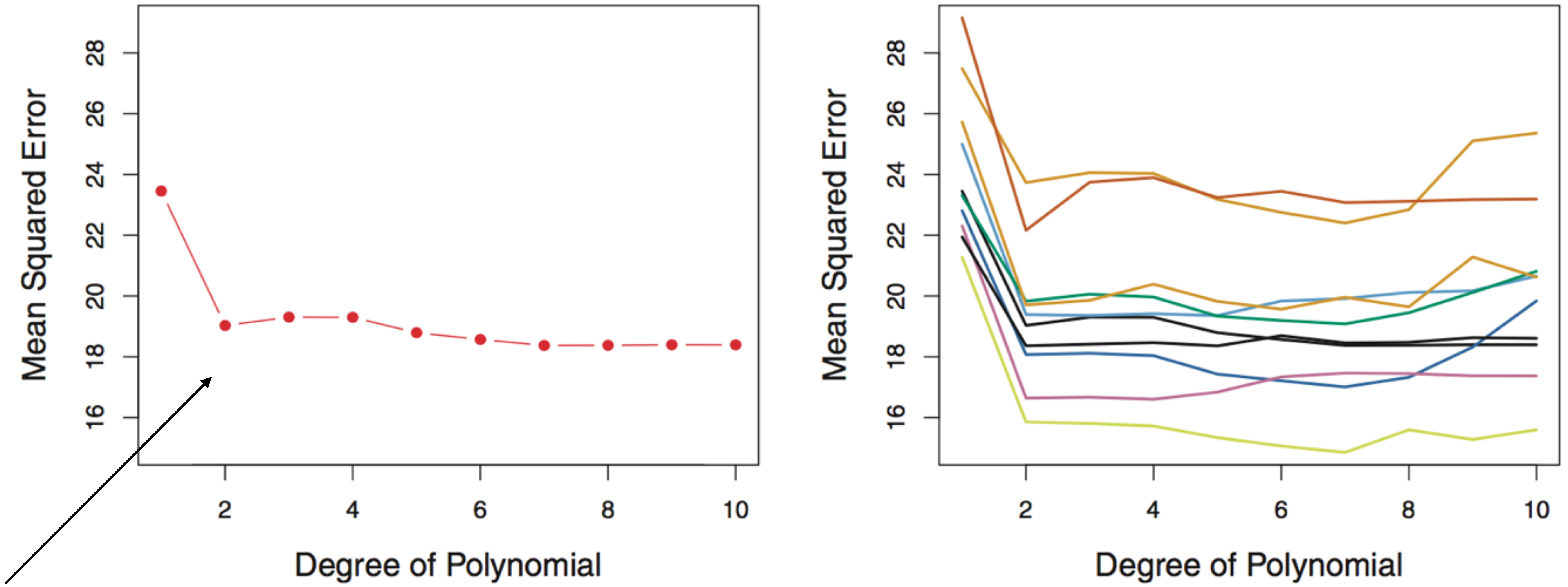
FIGURE 5.1, ISL (8th printing 2017)

Problems with 1 Dataset Split

Problems:

- Estimate of average error on unseen data can vary a lot, depending on which observations are in training, validation, and test sets.
- Only a subset of dataset is used to train the model. Since statistical methods tend to perform worse when trained on fewer observations, validation and test set errors may *overestimate* expected error on new data for a model fit on the entire dataset.

Problems with 1 Dataset Split



One train/
validation split

Ten train/
validation splits

FIGURE 5.2, ISL (8th printing 2017)

Leave-One-Out Cross-Validation

Step 1.

A single observation is used for the validation set; the remaining *n-1* observations make up the training set.

Step 2.

Model is fit on *n-1* training observations.

Step 3.

The error on the held-out observation (x_i, y_i) is an unbiased estimate for the error on new data.

E.g. $\mathrm{MSE}_i = (y_i - \hat{y}_i)^2$

Leave-One-Out Cross-Validation

The error estimated from a single observation will be highly variable, making it a poor estimate of test error.

So… we can repeat the leave-one-out procedure by selecting every observation as the validation set, and training on the remaining *n-1* observations.

This produces *n* error estimates, one from each held-out observation.

LOOCV estimate for test MSE: $$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i$$

Leave-One-Out Cross-Validation

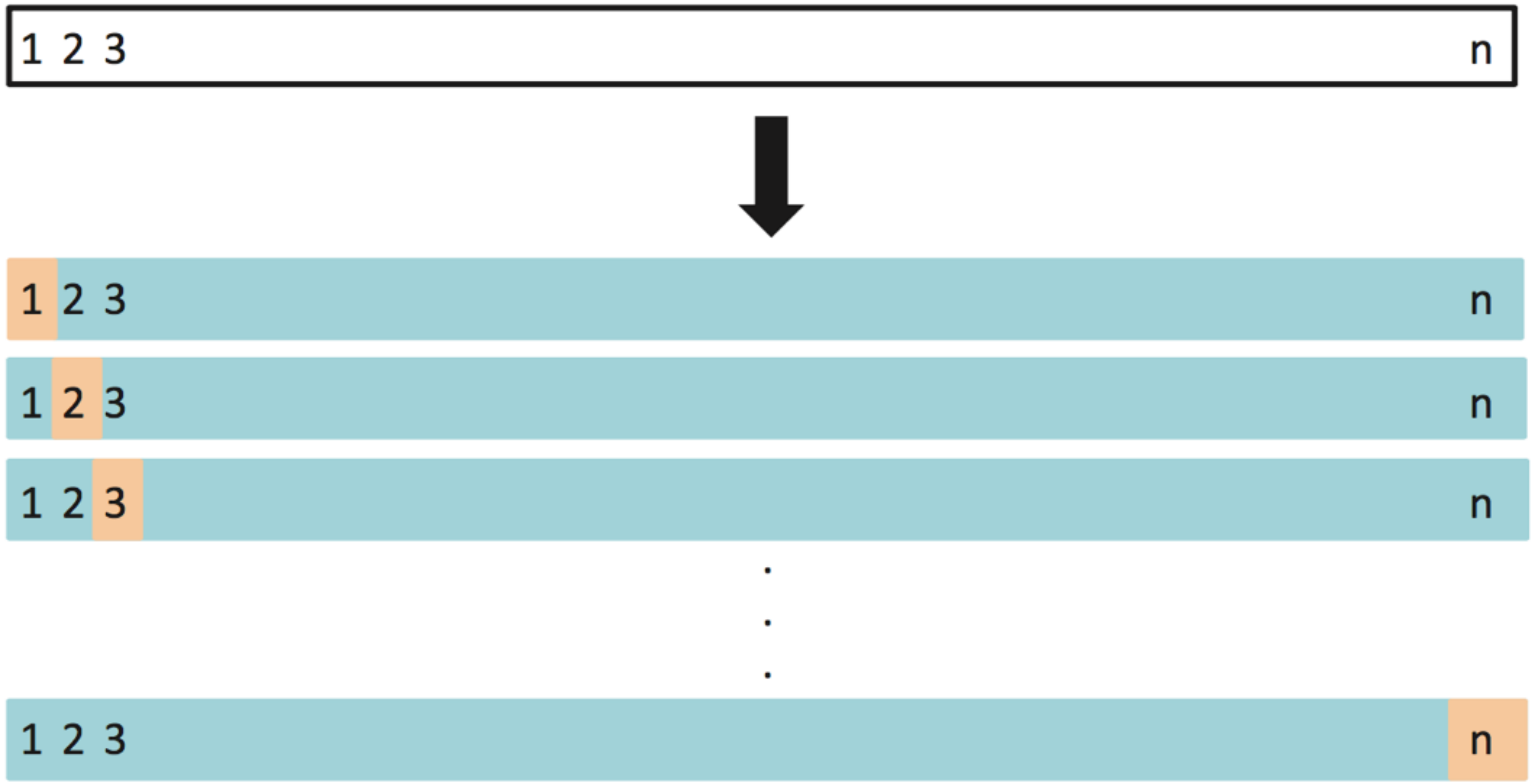


FIGURE 5.3, ISL (8th printing 2017)

Leave-One-Out Cross-Validation

Pros:

- Less bias; less overestimation of test error since n-1 is close to n
- Every LOOCV on a dataset will yield same results; no variation from exact training / validation split

Cons:

- Can be computationally expensive to implement

k-Fold Cross-Validation

Step 1.

Randomly divide the dataset into *k* groups, aka "*folds*". First fold is validation set; remaining *k-1* folds are training.

Step 2.

Model is fit on *k-1* folds of training observations.

Step 3.

The error on the held-out fold is an unbiased estimate for the error on new data.

E.g.

$$\mathrm{MSE}_1 = \frac{1}{|\mathrm{Fold}_1|} \sum_{(x_i, y_i) \in \mathrm{Fold}_1} (y_i - \hat{y}_i)^2$$

k-Fold Cross-Validation

Like in LOOCV, repeat this for each of the k folds.

This produces k error estimates, one from each held-out fold.

k-fold estimate for test MSE: $$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_i$$

LOOCV is a special case of k-fold CV with $k = n$.

In practice, usually set $k = 5$ or $k = 10$.

k-Fold Cross-Validation

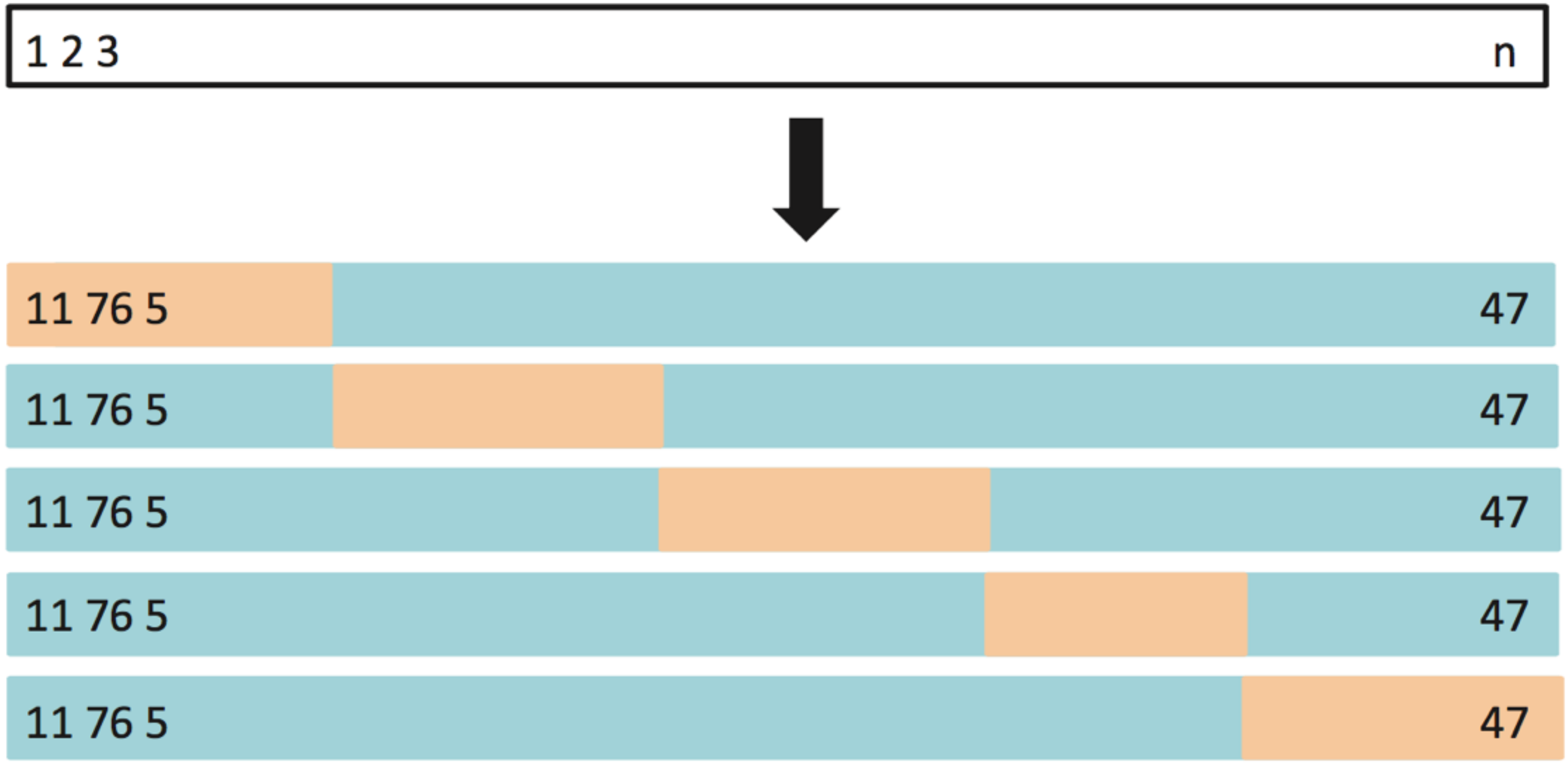


FIGURE 5.5, ISL (8th printing 2017)

k-Fold Cross-Validation

Pros:

- Since it's an average over k splits, less variation than one training / validation split
- Computationally more feasible than LOOCV, also less variance

Cons:

- Still more biased / more prone to overestimating error than LOOCV since 10-20% data not used for training

LOOCV vs. k-fold CV

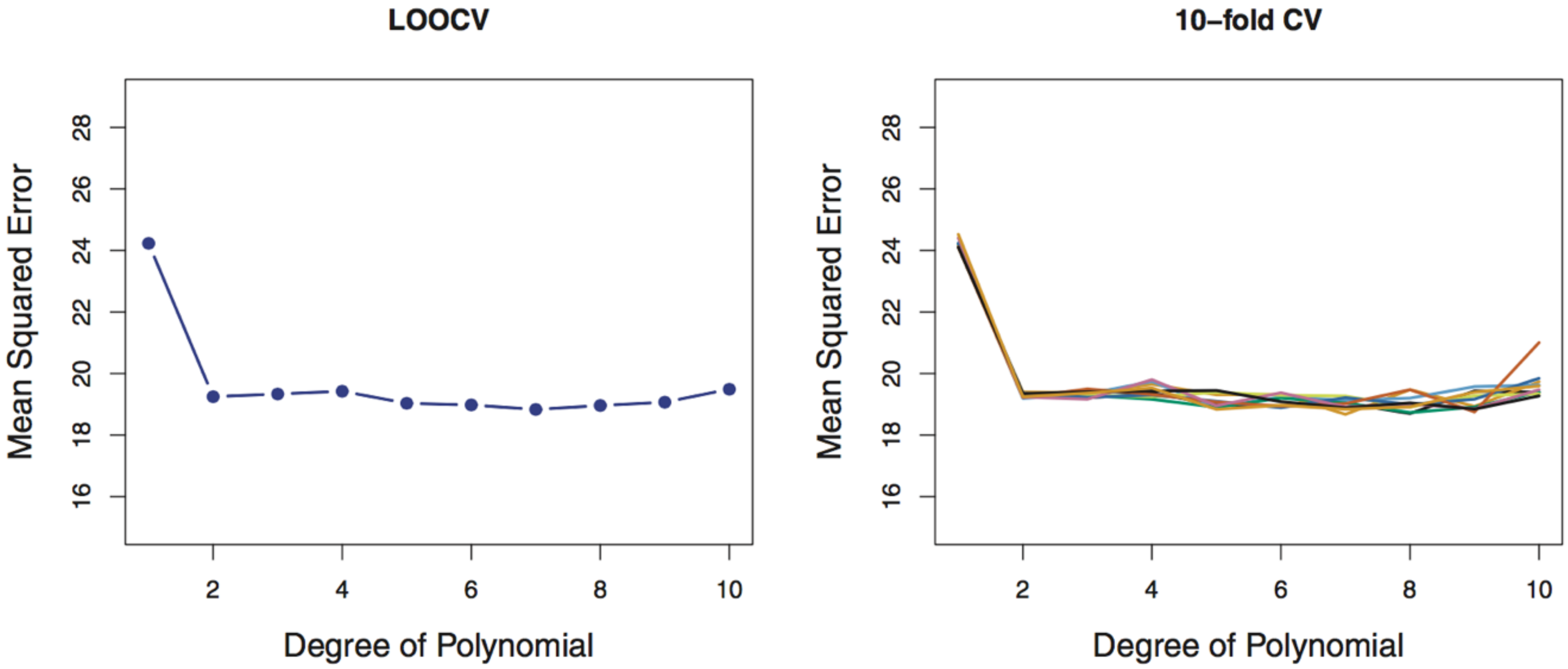


FIGURE 5.4, ISL (8th printing 2017)

Bias-Variance Tradeoff in CV

Bias: LOOCV gives less biased estimate of generalization error than k-fold CV.

Variance: LOOCV has higher variance than k-fold CV.

Why? LOOCV averages error from n models, each of which is trained on nearly identical datasets. These errors are highly positively correlated.

The variance of the mean of many highly correlated quantities is higher than the variance of the mean of less correlated quantities.

In practice, people use k-fold CV with $k = 5$ or $k = 10$.

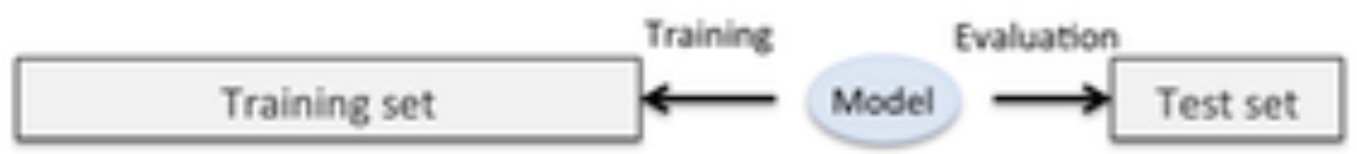
What about the test set?

If we fit many models to a validation set and select the best one, the error estimate risks becoming an underestimate of true generalization error due to being tuned to the validation set.

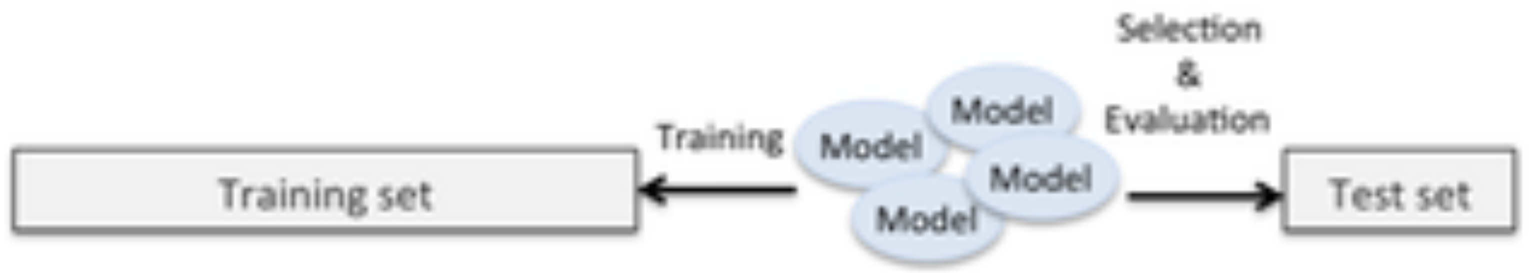
Option 1: Find best hyperparameters on 1 validation set and apply to 1 test set.

Option 2: Find best hyperparameters on k validation folds and apply to 1 test set.

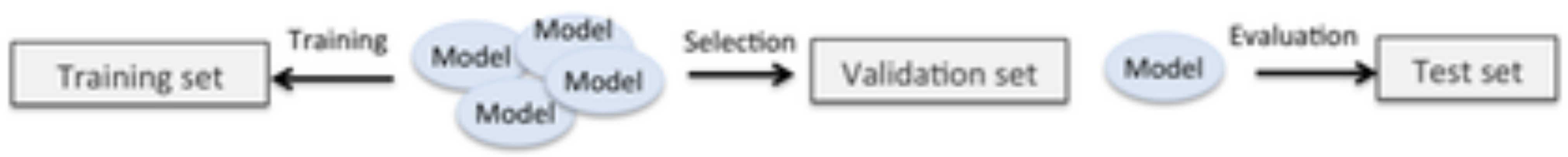
✓



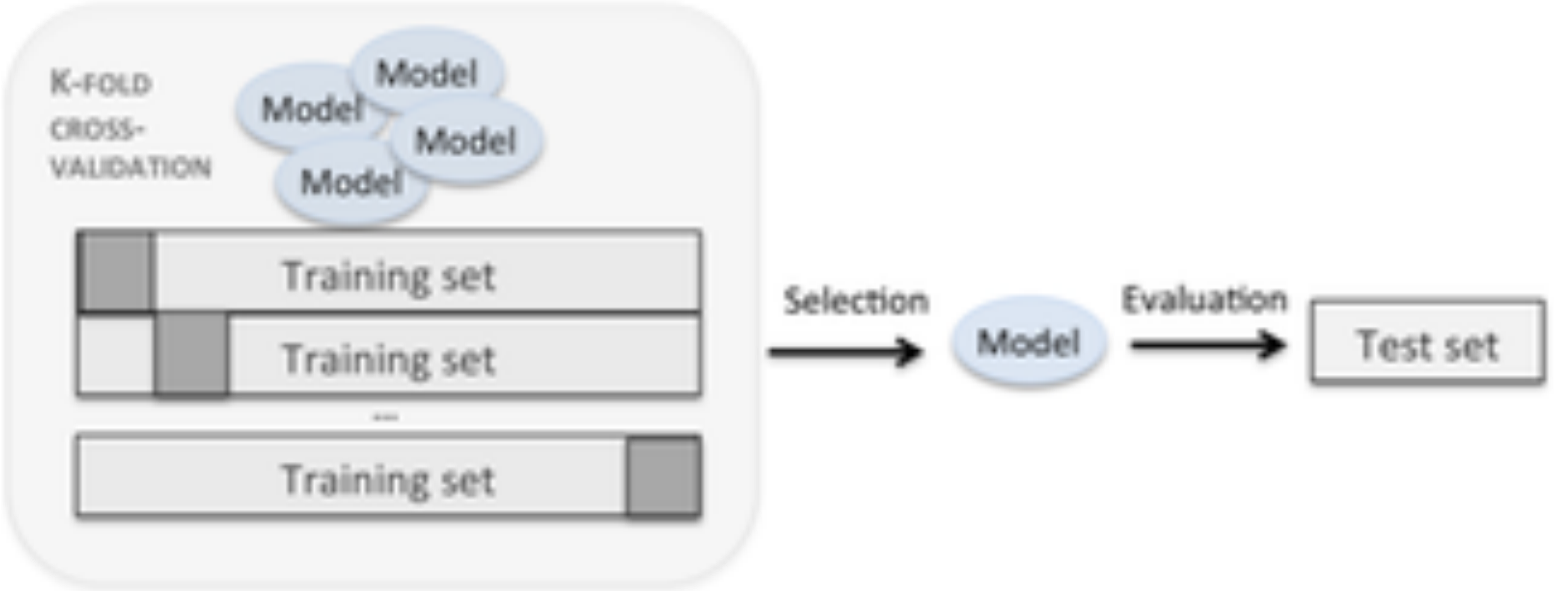
✗



✓



✓✓

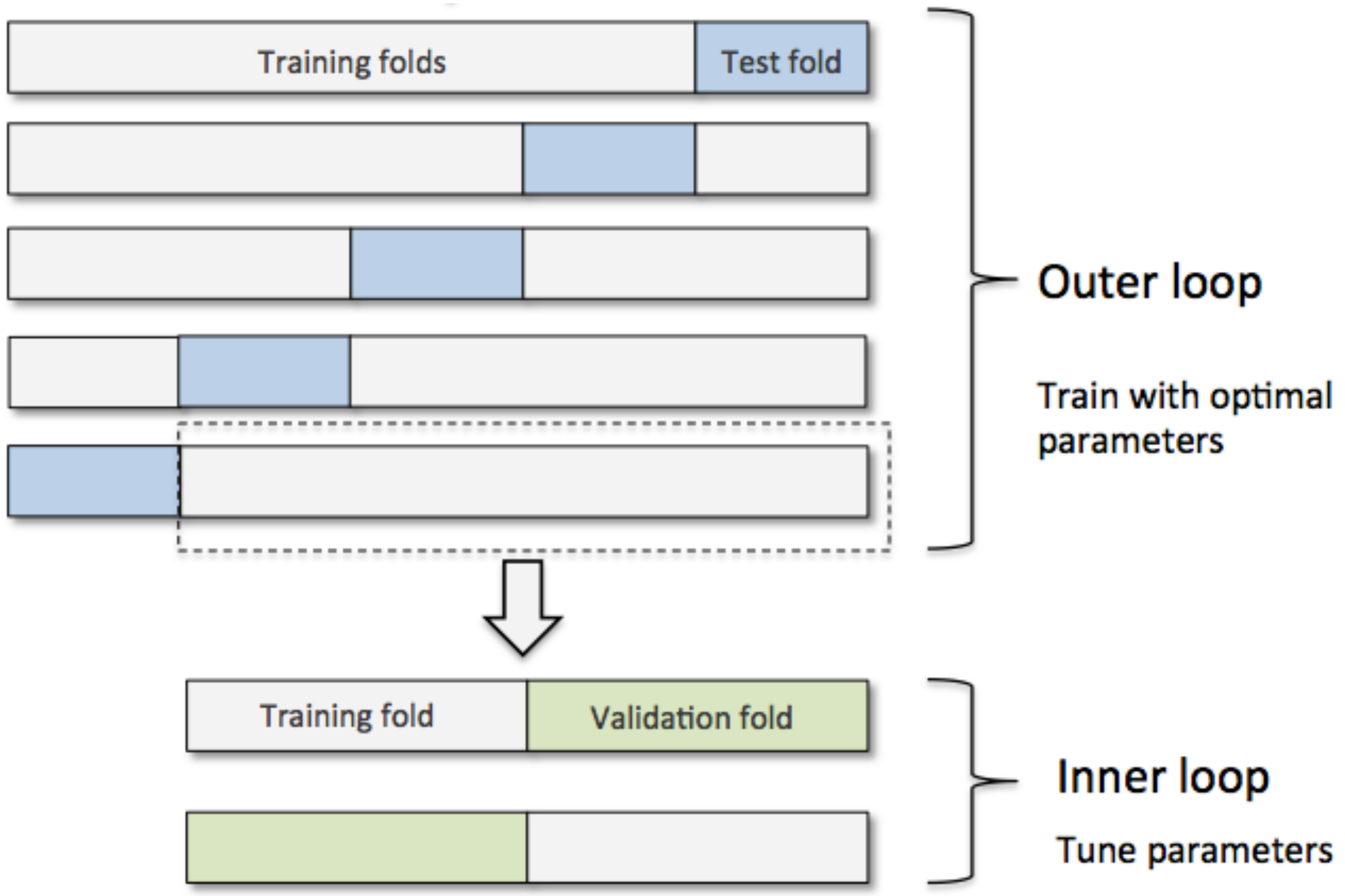


Nested Cross-Validation

Option 3: Tune hyperparameters on the validation set in an “inner loop”, estimate generalization error in an “outer loop”.

If your models are stable*, each completion of the inner loop should yield similar hyperparameters.

*stable = does not change a lot with small perturbations in training data



https://sebastianraschka.com/faq/docs/evaluate-a-model.html

Cross-validation in Python: `sklearn`

Non-nested 5-fold CV:

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV

lasso = Lasso(random_state=0)
alphas = np.logspace(-4, -0.5, 30)
params = {‘alpha’: alphas}

gridcv = GridSearchCV(estimator=lasso, param_grid=params, cv=5)
gridcv.fit(X, y)
scores = gridcv.cv_results_[‘mean_test_score’]
```

Cross-validation in Python: `sklearn`

Nested 5-fold CV:

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV, cross_val_score, KFold

lasso = Lasso(random_state=0)
alphas = np.logspace(-4, -0.5, 30)
params = {‘alpha’: alphas}

inner_cv = KFold(n_splits=5, shuffle=True, random_state=0)
outer_cv = KFold(n_splits=5, shuffle=True, random_state=0)

gridcv = GridSearchCV(estimator=lasso, param_grid=params, cv=inner_cv)
nested_score = cross_val_score(estimator=gridcv, X=X, y=y, cv=outer_cv)
scores = clf.cv_results_[‘mean_test_score’]
```

Good explanation: https://stackoverflow.com/questions/42228735/scikit-learn-gridsearchcv-with-multiple-repetitions/42230764#42230764

Cross-validation in R: caret

```
require(caret)

train_set <- createDataPartition(y, p=0.8)

cv_splits <- createFolds(y, k = 5, returnTrain=TRUE)

params <- expand.grid(alpha = c(0, 0.1, 0.2, 0.5))
ctrl <- trainControl(method = “cv”, number = 5)
fit <- train(response ~ ., data = df,
             method = “glmnet”, tuneGrid = params,
             trControl = ctrl)
```

http://www.milanor.net/blog/cross-validation-for-predictive-analytics-using-r/

Cross-validation in Matlab

Useful functions:

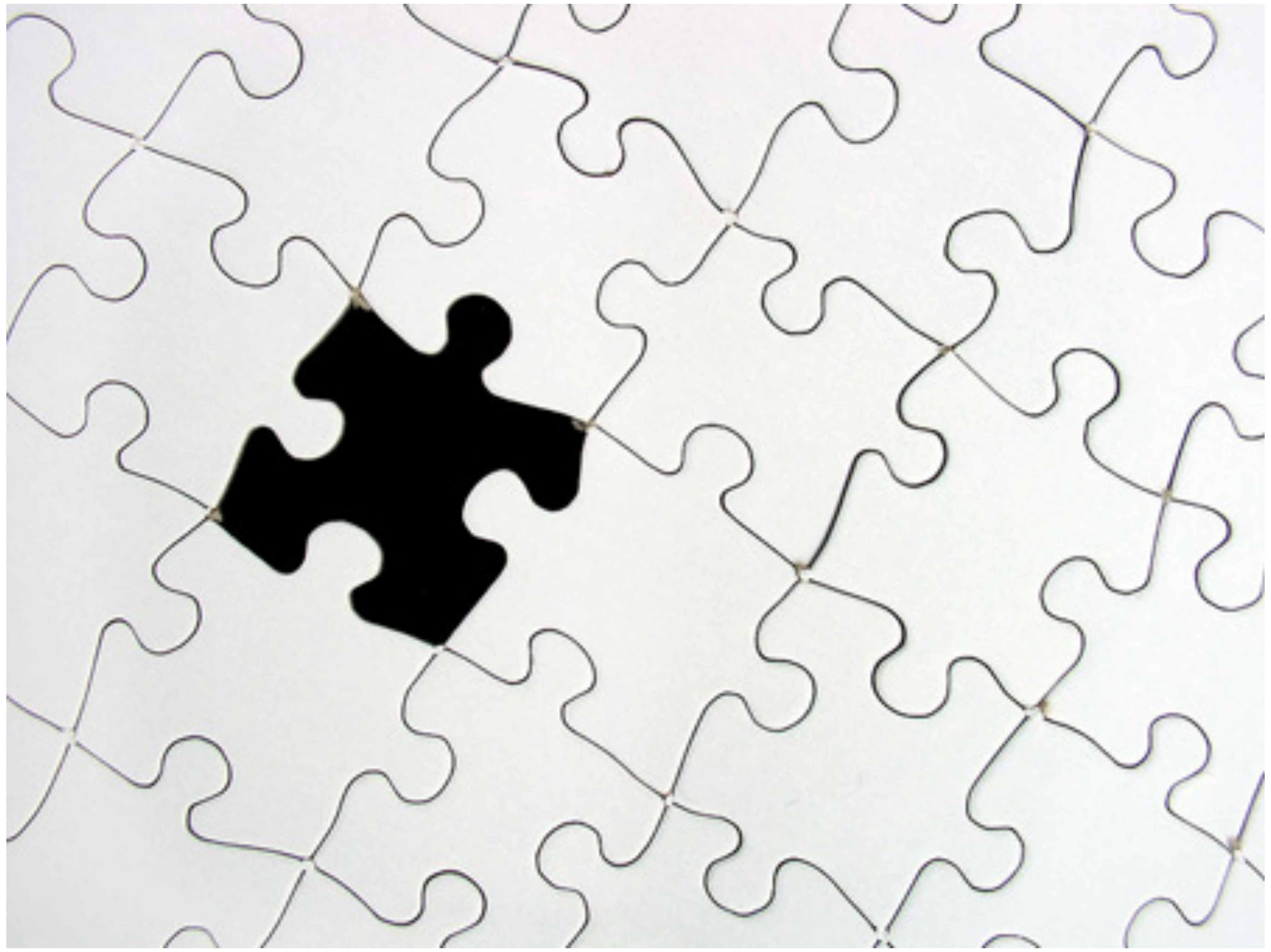
- `vals =` **`crossval`**`(fun, X)`
- `c =` **`cvpartition`**`(n, ‘KFold’, k)`
- `[X,Y] =` **`meshgrid`**`(x,y)`

Missing Data

Missing Data

Occurs when no value is stored for a feature or response variable at a particular sample.

Very common in reality. Can arise from non-response, study participant attrition, data logging mistakes.



How to handle missing data?

A few options:

- Imputation
- Deletion
- Use methods unaffected by missing values

Missing Completely at Random (MCAR)?

A fundamental assumption for imputation or deletion.

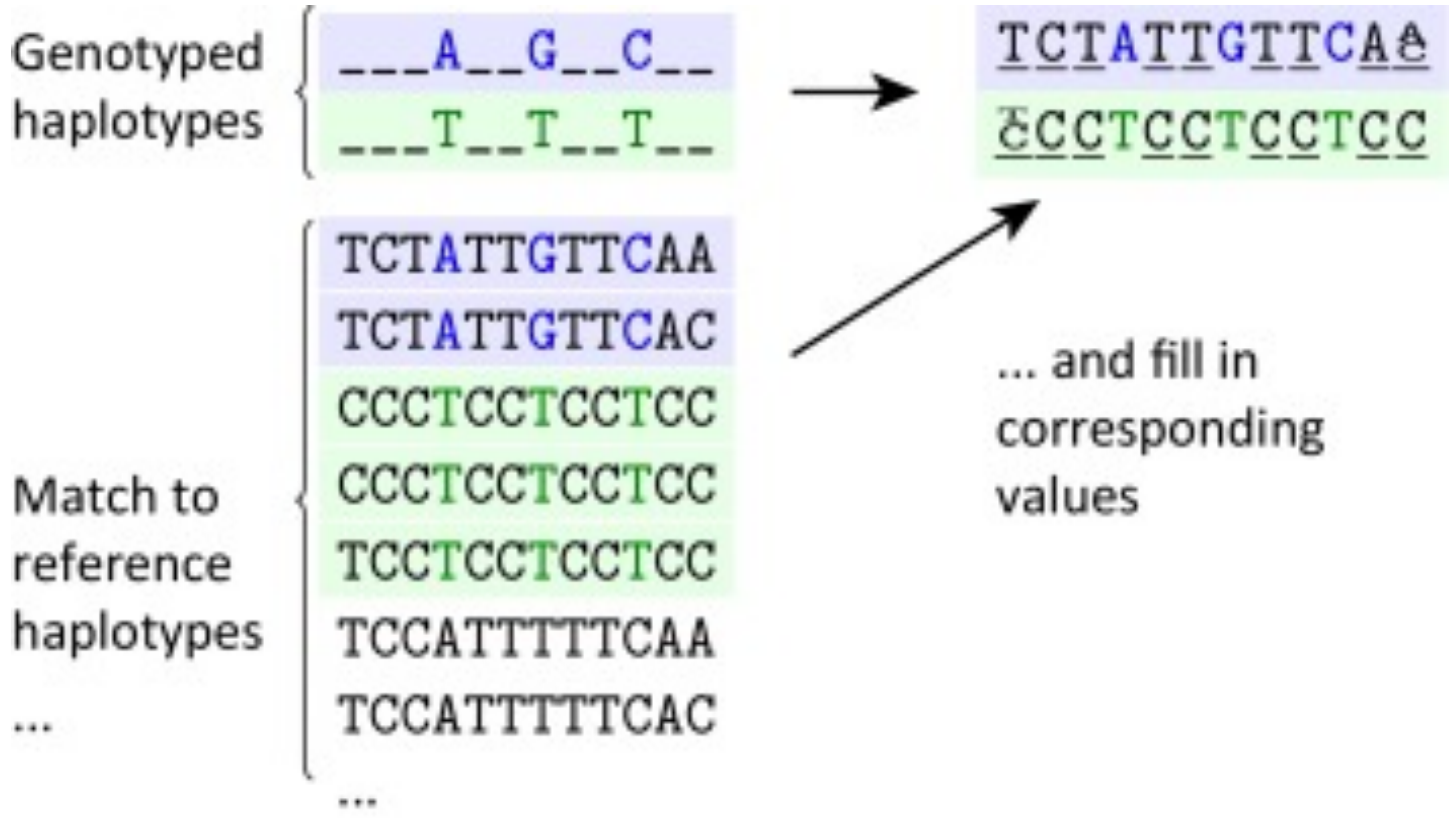
If not MCAR, imputation or deletion will bias the data.

One way to test this assumption: code missing data as “missing” and non-missing data as “not”, and then run classification with missingness as the response. If not MCAR, a supervised learning method may find a pattern to the missingness.

Imputation

The process of replacing missing data with substituted values.

Why? Most machine learning methods in their vanilla form cannot handle samples with one or more features missing.



Genotyped haplotypes

___A__G__C__

___T__T__T__

Match to reference haplotypes

...

TCTATTGTTCAA

TCTATTGTTCAC

CCCTCCTCCTCC

CCCTCCTCCTCC

TCCTCCTCCTCC

TCCATTTTTCAA

TCCATTTTTCAC

...

TCTATTGTTCAA

CCCTCCTCCTCC

... and fill in corresponding values

Trends in Genetics

Mean Imputation

Compute the overall mean for that feature and fill in missing value with that value. Can also use median or mode.

Pros: Fast to compute, does not change feature mean

Cons: Reduces variance in dataset

KNN Imputation

1. Fill in missing values using the mean or median for that variable.

2. Compute the distance between observation missing a value and all other observations to find the **k** nearest neighbors. Ignore the variable that is missing the value when computing the distance.

3. Fill in missing values with the mean or median of that variable of the **k** nearest neighbors.

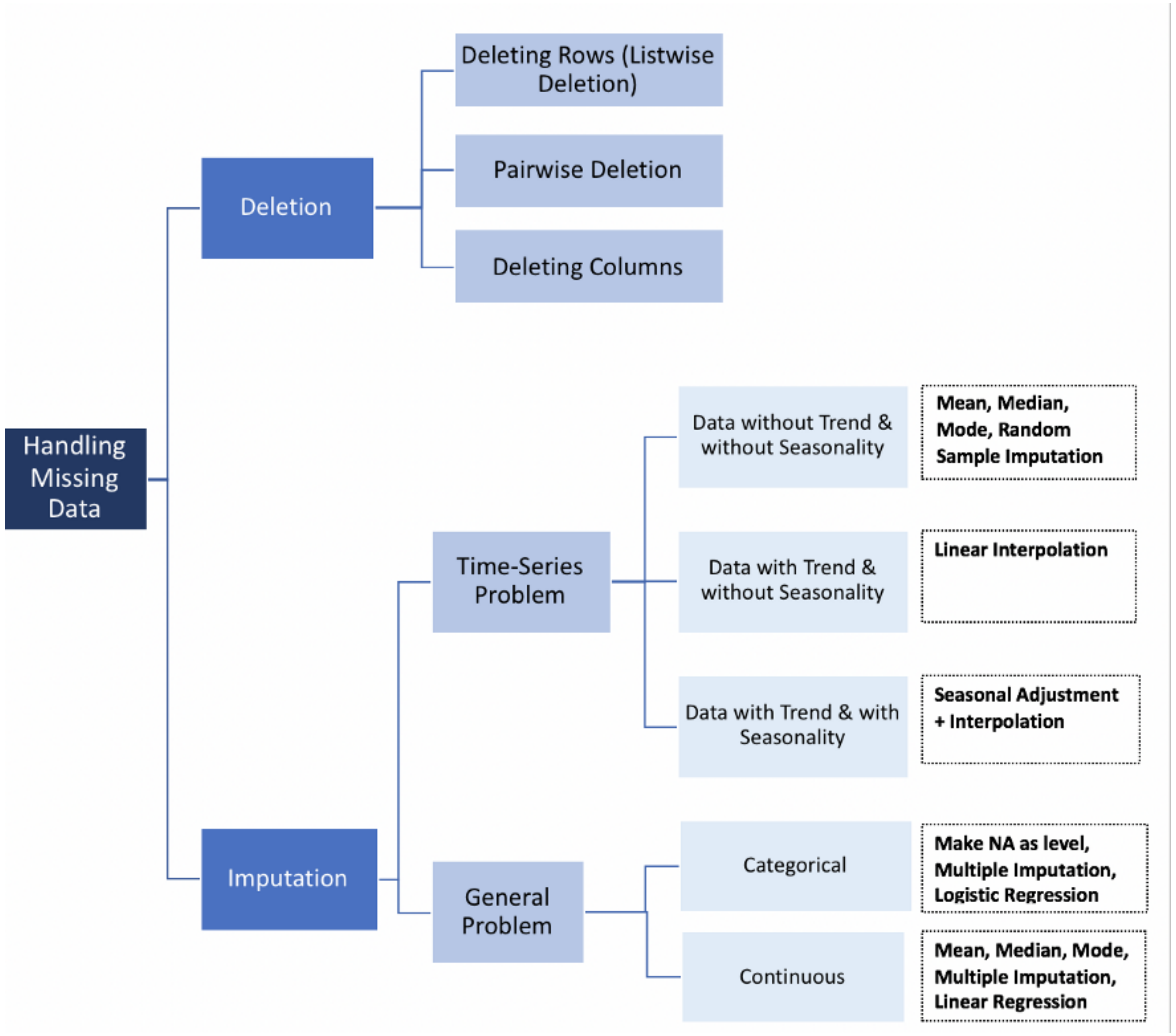
Regression Imputation

Regress the missing variable on other variables.

The imputed value is the predicted value for the missing variable.

Can use any regression (or classification if categorical) method. In practice CART works well.

Summary


Handling Missing Data
Deletion
Deleting Rows (Listwise Deletion)
Pairwise Deletion
Deleting Columns
Imputation
Time-Series Problem
Data without Trend & without Seasonality
Mean, Median, Mode, Random Sample Imputation
Data with Trend & without Seasonality
Linear Interpolation
Data with Trend & with Seasonality
Seasonal Adjustment + Interpolation
General Problem
Categorical
Make NA as level, Multiple Imputation, Logistic Regression
Continuous
Mean, Median, Mode, Multiple Imputation, Linear Regression