RESAMPLING METHODS

Prof. Mehrdad Pirnia

This content is based on concepts from "Introduction to Statistical Learning" by James, Witten, Hastie, and Tibshirani.

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Lecture Outcome

By the end of this lecture, you will be able to:

- Recognize the resampling method to evaluate the performance of ML models
- Perform k-fold cross validation
- Execute bootstrapping to create more data

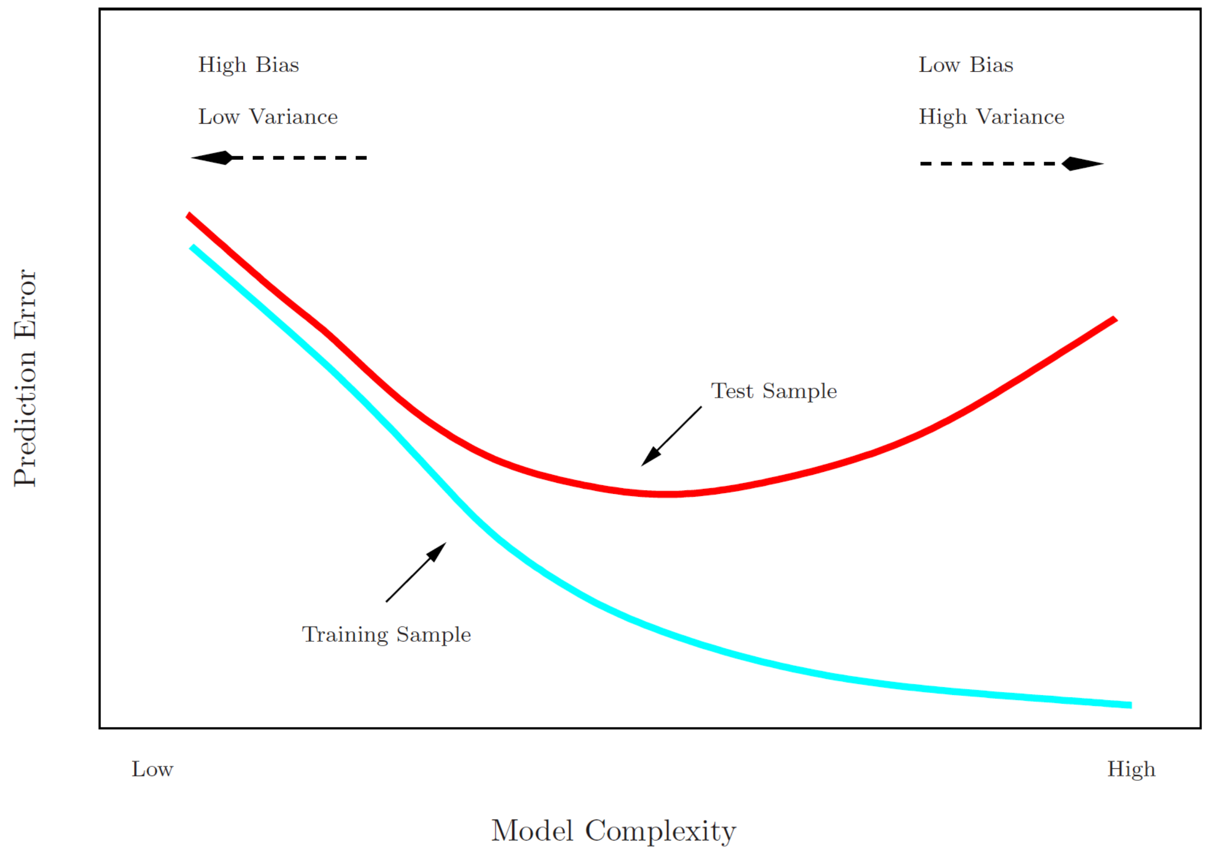
Cross-validation and the Bootstrap

- In the section we discuss two **resampling** methods: cross-validation and the bootstrap.
- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.
- For example, they provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates.

Training Error versus Test Error

- Recall the distinction between the test error and the training error:
  - The **test error** is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
  - In contrast, the **training error** can be easily calculated by applying the statistical learning method to the observations used in its training.
  - But the training error rate often is quite different from the test error rate, and in particular the former can **dramatically underestimate** the latter.

Training- versus Test-Set Performance



More on Prediction-error Estimates

- Best solution: a large designated test set. Often not available.
- Some methods make a **mathematical adjustment** to the training error rate in order to estimate the test error rate. These include the **Cp statistic**, AIC and BIC. They are discussed elsewhere in this course.
- Here we instead consider a class of methods that estimate the test error by **holding out** a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

Validation-set Approach

- Here we randomly divide the available set of samples into two parts: a **training set** and a **validation** or **hold-out set**.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error. This is typically assessed using MSE in the case of a quantitative response and misclassification rate in the case of a qualitative (discrete) response.

•**The training set** is used to fit the model, and includes the majority of our observations. For example, we might set aside 70% of our original data to be used for estimating the parameters of our model.

•**The validation set** is used during parameter tuning to avoid overfitting the model, which is when the model fits the training data too well but performs poorly on new data. This set does not overlap with the training set or the test set.

•**The test set** only includes data that the model has never seen before. This set is used to evaluate the performance of the model.

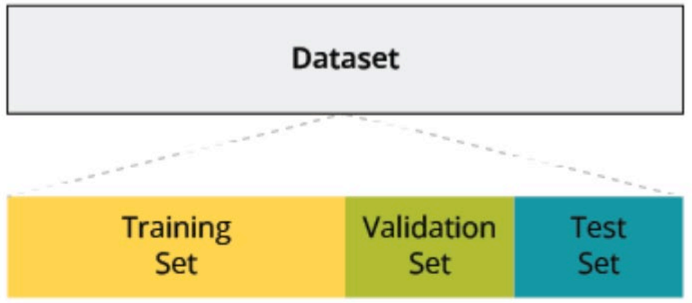
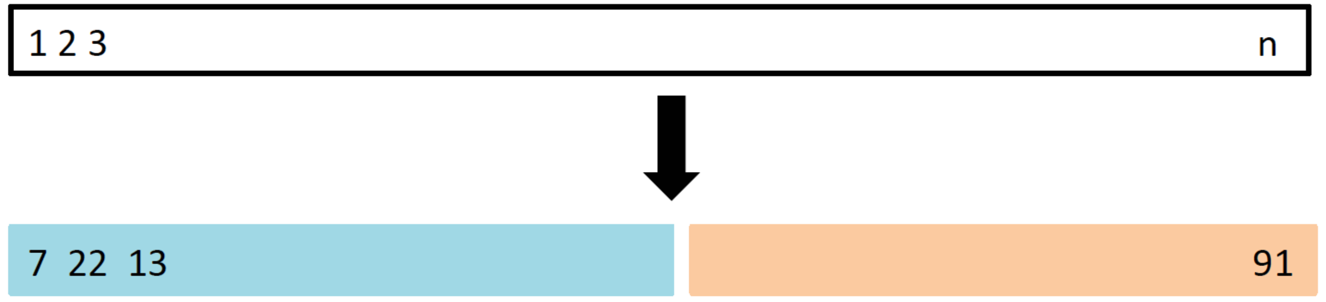


Figure 1: Training, Validation, and Test Sets

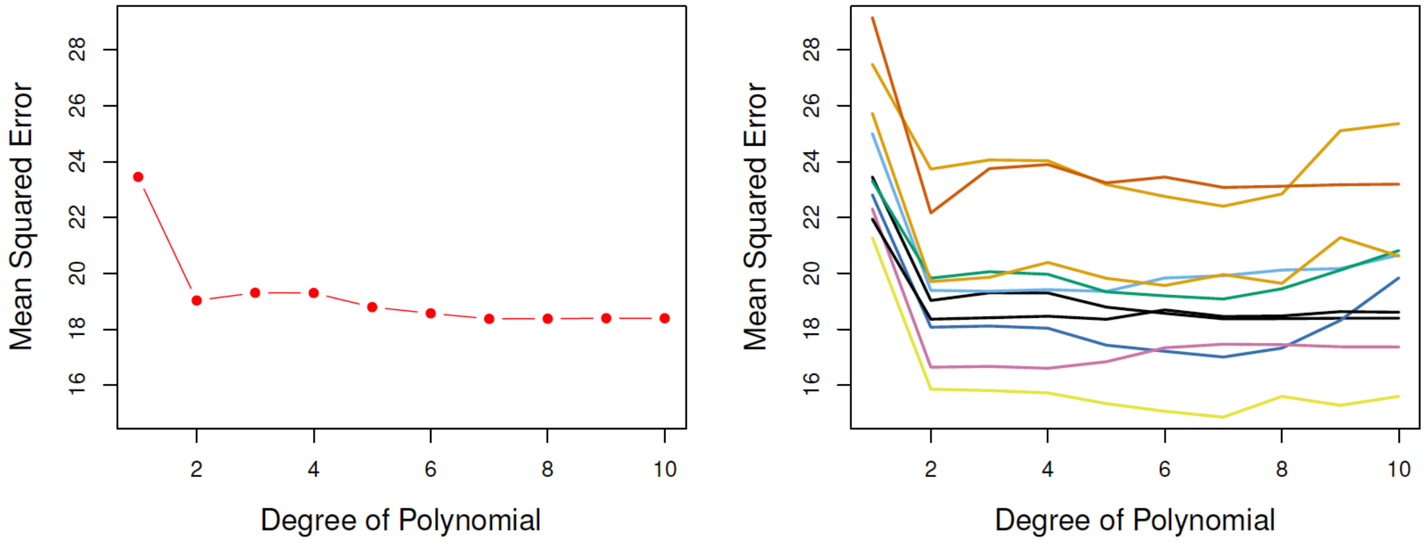
The Validation process



A random splitting into two halves: left part is training set, right part is validation set

Example: Automobile Data

- Want to compare linear vs higher-order polynomial terms in a linear regression
- We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.



Left panel shows single split; right panel shows multiple splits

Drawbacks of Validation Set Approach

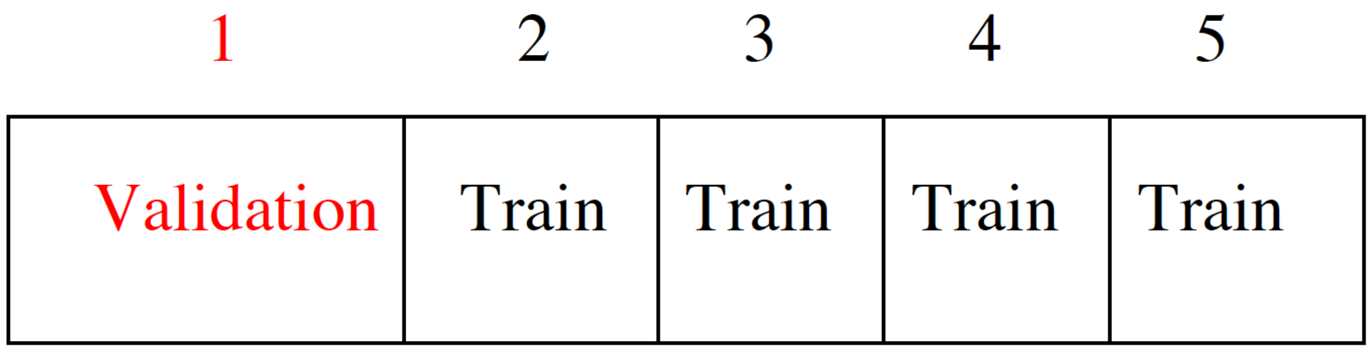
- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to **overestimate** the test error for the model fit on the entire data set. Why?

K-fold Cross-validation

- **Widely used approach** for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into **K** equal-sized parts. We leave out part **k**, fit the model to the other **K - 1** parts (combined), and then obtain predictions for the left-out **k**-th part.

This is done in turn for each part **k = 1, 2, ..., K**, and then the results are combined

K-fold Cross-validation in detail



Divide data into K roughly equal-sized parts (K = 5 here)

KFOLD

```
from sklearn.model_selection import KFold

kfold = KFold(n_splits=5, shuffle=False, random_state=None)
```

IMPORTANT PARAMETERS:

- `n_splits`: number of folds. This value should be at least 2, and the default is 5.
- `shuffle`: set to `True` to shuffle the data before splitting into batches. The default is `False`.
- `random_state`: an integer that affects the ordering of the indices if `shuffle` is set to `True`.

The Details

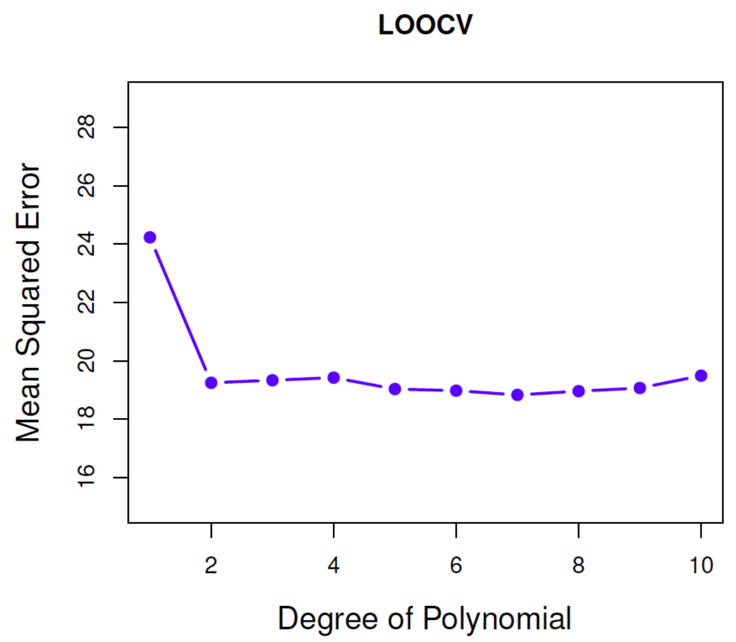
- Let the **K** parts be **C1, C2, ..., CK**, where **Ck** denotes the indices of the observations in part **k**. There are **nk** observations in part **k**: if **N** is a multiple of **K**, then **nk = n/K**.
- Compute

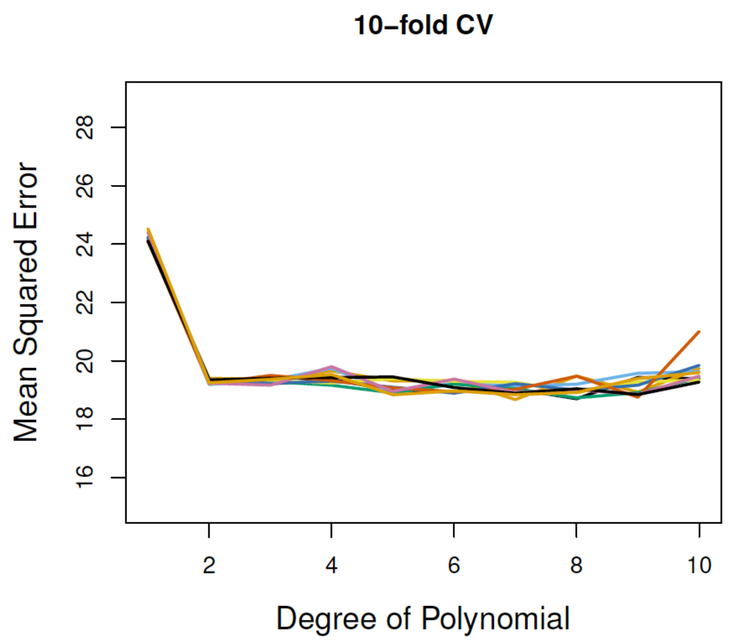
$$CV(K) = \sum_{k=1}^{K} \frac{nk}{n} MSE_k$$

where **MSEk** = $\sum_{i \in Ck} (y_i - \hat{y}_i)^2 \, / nk$, and $\hat{\mathbf{y}}_{\boldsymbol{i}}$ is the fit for observation i, obtained from the data with part **k** removed.

- Setting **K = n** yields **n**-fold or **leave-one-out cross-validation** (LOOCV).

Auto data revisited





Other Issues with Cross-validation

- Since each training set is only **(K - 1)/K** as big as the original training set, the estimates of prediction error will typically be biased upward. **Why?**
- This bias is minimized when **K = n** (LOOCV), but this estimate has high variance, as noted earlier.
- **K = 5 or 10** provides a good compromise for this bias-variance tradeoff.

Cross-Validation for Classification Problems

- We divide the data into **K** roughly equal-sized parts **C1, C2, …, CK**. **Ck** denotes the indices of the observations in part **k**. There are **nk** observations in part **k**: if **n** is a multiple of **K**, then **nk = n/K**.
- Compute

$$CV_K = \sum_{k=1}^{K} \frac{nk}{n} Err_k$$

where **Errk** = $\sum_{i \in Ck} I\,(y_i \neq \hat{y}_i)/nk.$

- The estimated standard deviation of **CV_K** is

$$\widehat{SE}(CV_K) = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\frac{\left(Err_k - \bar{Err_k}\right)^2}{K-1}}$$

This is a useful estimate, but strictly speaking, not quite valid. Why not?

Numerical Example

Assume the following dataset, with one feature and one target variable, and 5 observations, and obtain k fold CV for $k = 2$:

Index	Feature (X)	Target (Y)
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6

- **Step 1: Split the data into 2 folds.**

  Fold 1: (Training: Index 3, 4, 5; Validation: Index 1, 2)

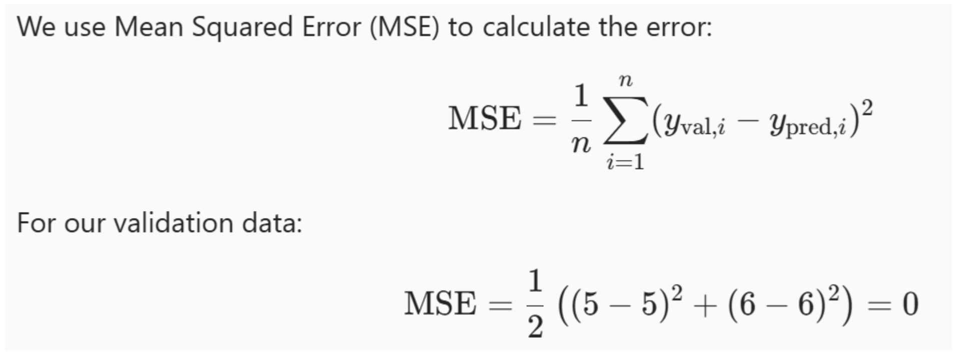
  Fold 2: (Training: Index 1, 2, 3; Validation: Index 4, 5)

Example --- Continue

Step 2: Train and validate the model on each fold.

Fold 2:

- Training Data: (1, 2), (2, 3), (3, 4)
- Validation Data: (4, 5), (5, 6)
- Train a simple linear regression model on the training data. (y=x+1)
- Predict on the validation data and calculate the error.



We use Mean Squared Error (MSE) to calculate the error:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{val},i} - y_{\text{pred},i})^2$$

For our validation data:

$$\text{MSE} = \frac{1}{2}\left((5-5)^2 + (6-6)^2\right) = 0$$

Step 3: Average the validation errors from both folds to get the final error estimate.

Cross-validation: Right and Wrong

- Consider a simple classifier applied to some two-class data:
  a. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
  b. We then apply a classifier such as logistic regression, using only these 100 predictors.

  How do we estimate the test set performance of this classifier?

  Can we apply cross-validation in step 2, forgetting about step 1?

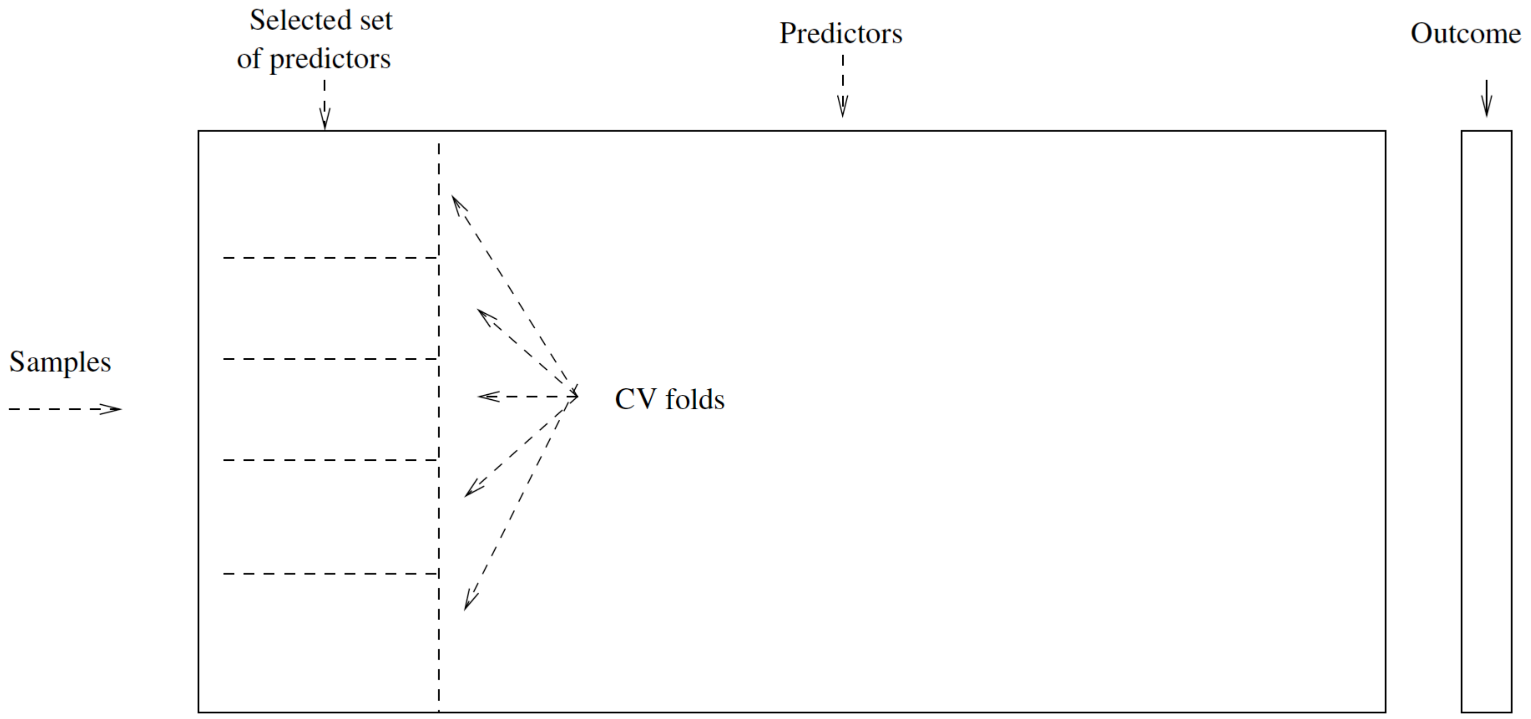
NO!

- This would ignore the fact that in Step 1, the procedure **has already seen the labels of the training data**, and made use of them. This is a form of training and must be included in the validation process.
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error = 50%, but the **CV error estimate** that ignores Step 1 is zero! **Try to do this yourself**.
- We have seen this error made in many high profile genomics papers.

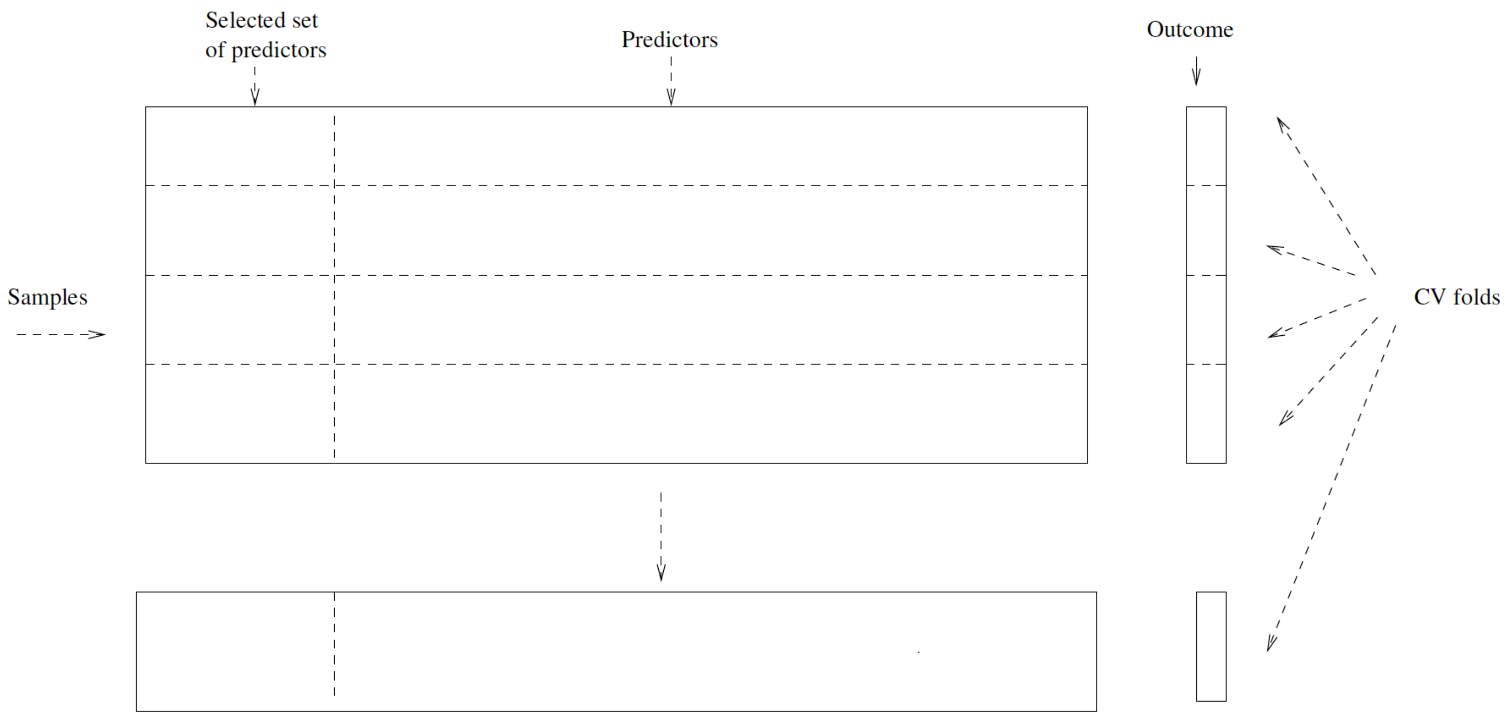
The Wrong and Right Way

- Wrong: Apply cross-validation in step 2.
- Right: Apply cross-validation to steps 1 and 2.

Wrong Way


Selected set of predictors
Predictors
Outcome
Samples
CV folds

Right Way


Selected set of predictors
Predictors
Outcome
Samples
CV folds

HYPERPARAMETER TUNING

Cross-validation can also be used for hyperparameter tuning. Let's start by reviewing the difference between parameters and hyperparameters:

•Model **parameters** are estimated from the training data, not manually set. For example, the coefficients of a regression model are parameters.

•Model **hyperparameters** are not learned from the data. We manually set them when initializing the model to optimize the model's performance. For example, the number of neighbors in KNN is a hyperparameter.

The Bootstrap

- The **bootstrap** is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

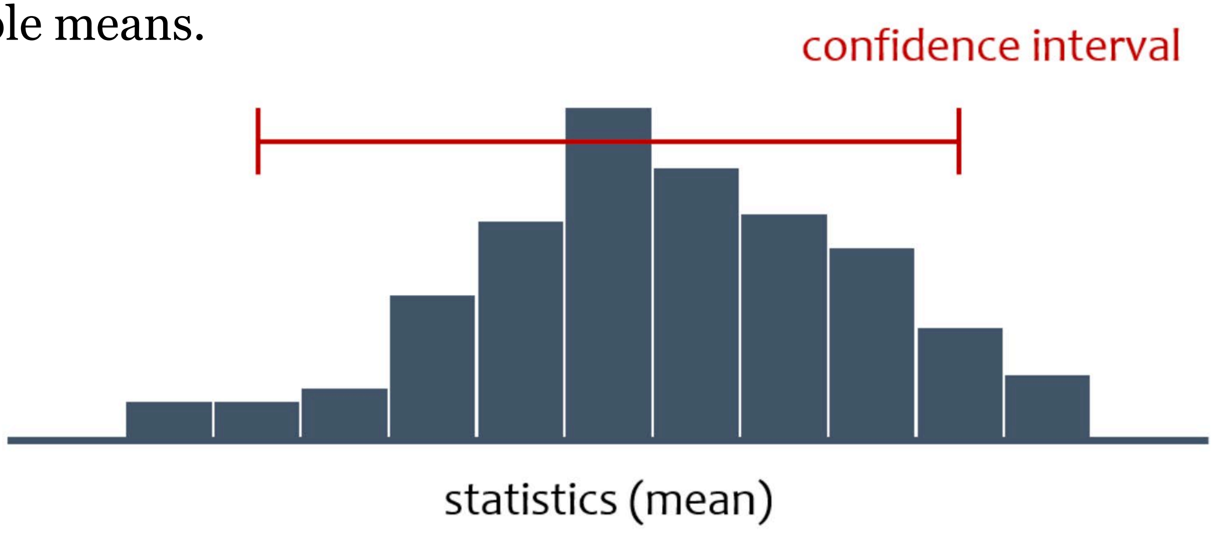
BOOTSTRAPPING STEPS

1. Construct an original sample by sampling from the population.
2. Create a bootstrap data set by sampling with replacement from the original sample. The size of the samples is typically the same size as your original data set but can be adjusted according to any specific requirements.
3. Measure the statistic of interest for each bootstrap sample (e.g., mean, median, variance, precision, or accuracy of the model).
4. Record the measured statistics.
5. Build a histogram of measured statistics to see which events are more likely to occur and which are rarer

BOOTSTRAP FOR CONFIDENCE INTERVALS

- One significant advantage of bootstrapping is its ability to determine the confidence intervals of the statistics of our interest once we've obtained the bootstrap samples. As we saw previously, the bootstrap repeats an experiment many times, often in the thousands, to create a distribution of statistics, such as a mean of the bootstrap sample. We can use the histogram to estimate the distribution of the mean.

- For example, let's say we have a histogram of 1000 bootstrap sample means below. If we take a new sample from the original population, the new sample mean will likely fall within the range given by the histogram of bootstrap sample means.



UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Where does the name come from?

- The use of the term **bootstrap** derives from the phrase *to pull oneself up by one's bootstraps*, widely thought to be based on one of the eighteenth century "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:

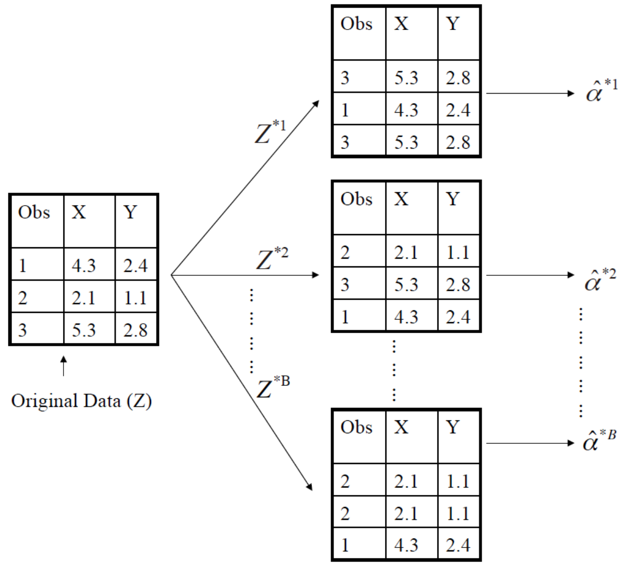
  The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

- It is not the same as the term **"bootstrap"** used in computer science meaning to "boot" a computer from a set of core instructions, though the derivation is similar.

Why Bootstrapping

- The **bootstrap** approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set **with replacement**.
- Each of these "bootstrap data sets" is created by sampling **with replacement**, and is the **same size** as our original dataset. As a result, some observations may appear more than once in a given bootstrap data set and some not at all.

Example with Just 3 Observations



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

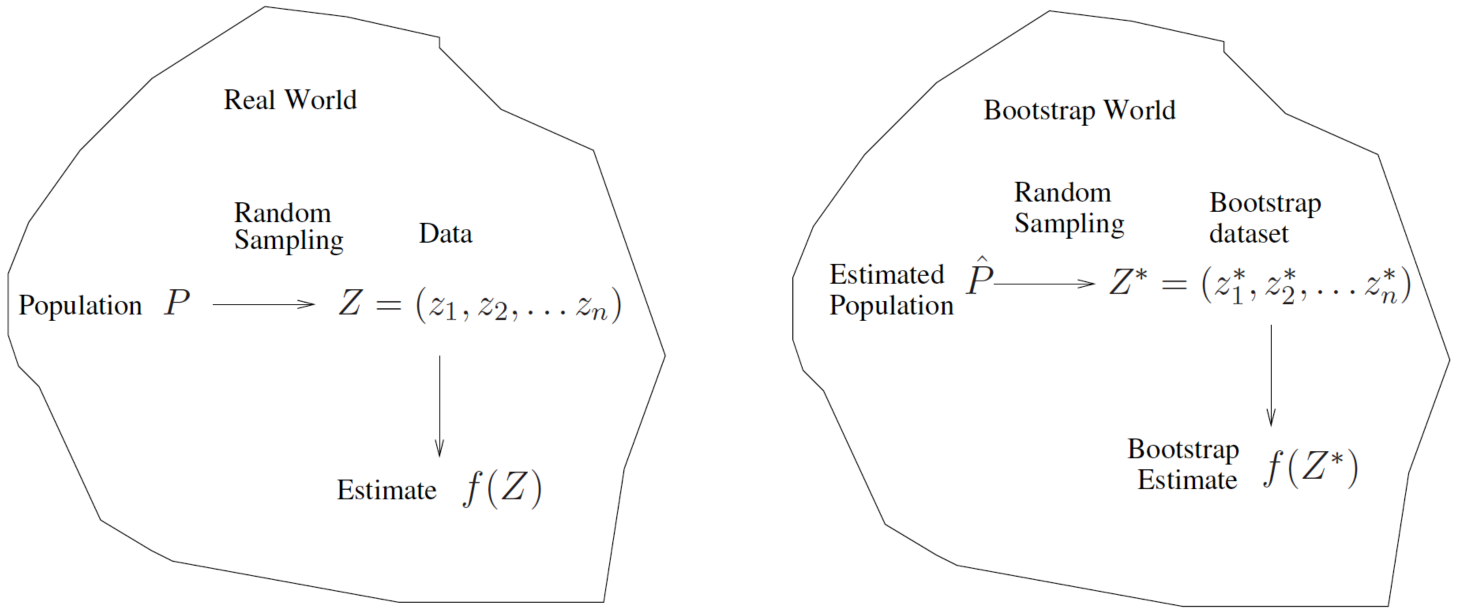
Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

- A graphical illustration of the bootstrap approach on a small sample containing **n = 3** observations. Each bootstrap data set contains **n** observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of **α**.

A general picture for the bootstrap


Real World
Random Sampling
Data
Population $P \longrightarrow Z = (z_1, z_2, \dots z_n)$
Estimate $f(Z)$
Bootstrap World
Random Sampling
Bootstrap dataset
Estimated Population $\hat{P} \longrightarrow Z^* = (z_1^*, z_2^*, \dots z_n^*)$
Bootstrap Estimate $f(Z^*)$

The Bootstrap in General

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can’t simply sample the observations with replacement (**why not?**).
- We can instead create blocks of consecutive observations, and sample those with replacements. Then we paste together sampled blocks to obtain a bootstrap dataset.

Other Uses of the Bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter.
- The interval is called a **Bootstrap Percentile** confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING