Principles of Machine Learning

Lecture 7: Model Evaluation and Validation

Sharif University of Technology
Dept. of Aerospace Engineering

April 27, 2025

Multioutput Classification

- Multioutput-multiclass classification is a generalization of multilabel classification where each label can be multiclass
- Example. (MNIST) Image Noise Removal

Multioutput Classification - (MNIST) Image Noise Removal

```
np.random.seed(42) # to make this code example reproducible
noise = np.random.randint(0, 100, (len(X_train), 784))
X_train_mod = X_train + noise
noise = np.random.randint(0, 100, (len(X_test), 784))
X_test_mod = X_test + noise
y_train_mod = X_train
y_test_mod = X_test

```

Multioutput Classification - (MNIST) Image Noise Removal

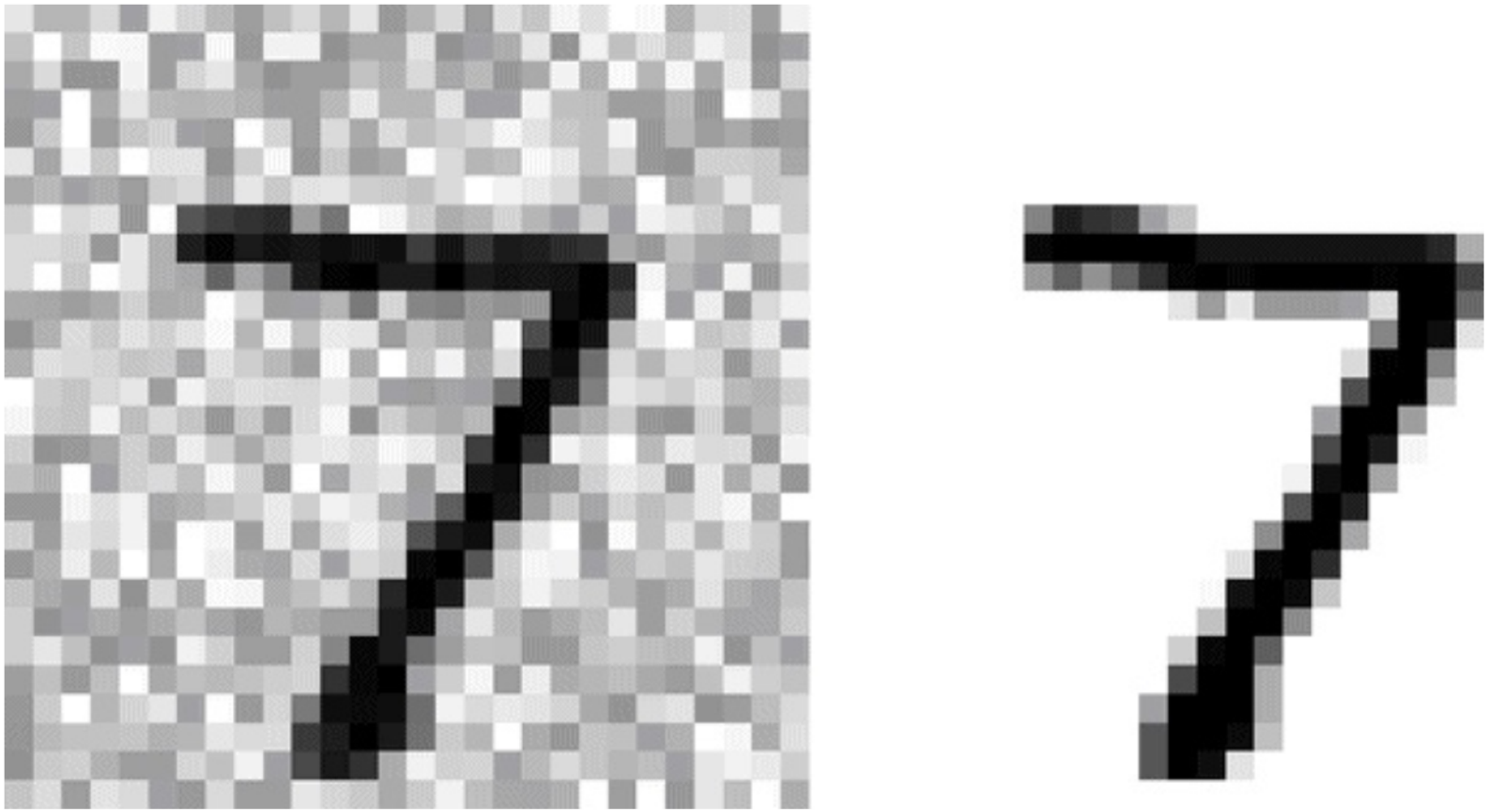


Figure: A noisy image (left) and the target clean image (right)

Multioutput Classification - (MNIST) Image Noise Removal

```
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train_mod, y_train_mod)
clean_digit = knn_clf.predict([X_test_mod[0]])
plot_digit(clean_digit)
plt.show()

```

Multioutput Classification - (MNIST) Image Noise Removal

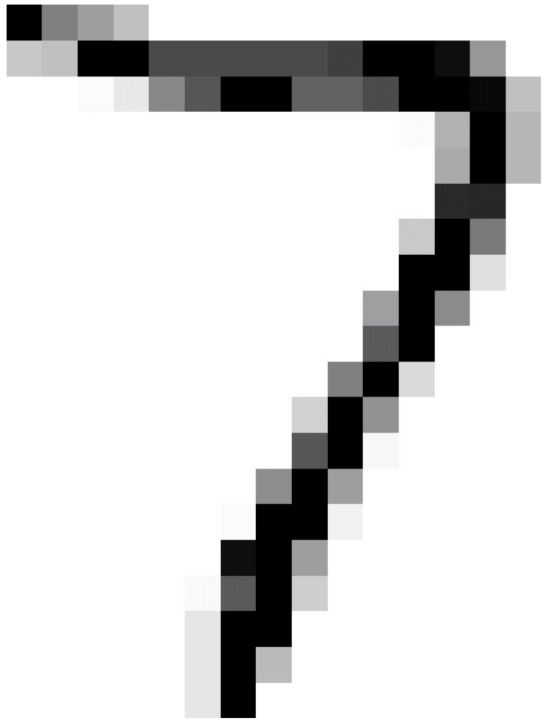


Figure: The cleaned-up image

Table of Contents

Error Analysis – Confusion Matrices

Let's say you have found a promising model for your project.
One way to improve it: **analyze the types of errors it makes**

Our focus of the *error analysis* will be on the following

- Confusion Matrices
- Data Augmentation

Error Analysis – Confusion Matrices

A colored diagram of the confusion matrix for better visualization

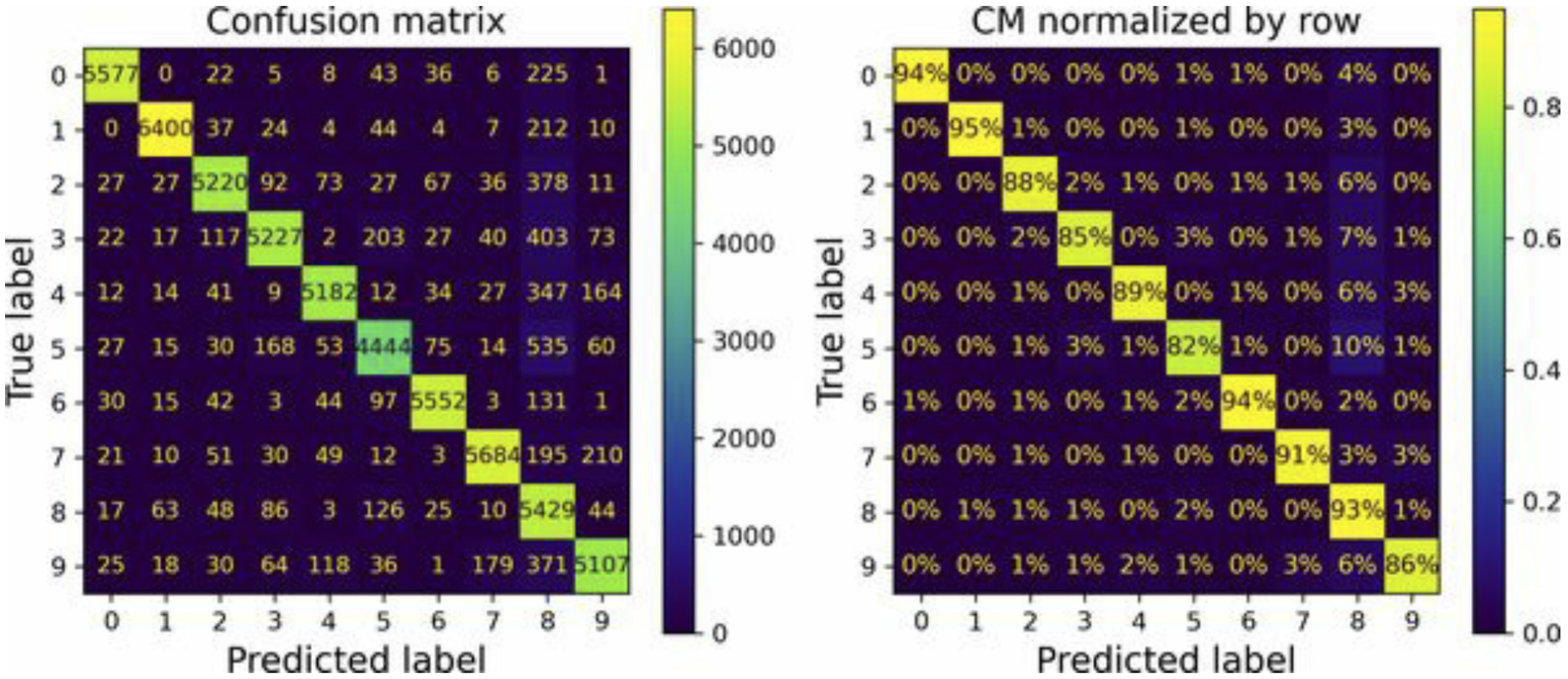


Figure: Confusion matrix (left) and the same CM normalized by row (right)

Error Analysis – Confusion Matrices

- **Correct Classifications**:
  - Most images lie along the main diagonal of the CM, indicating they were classified correctly.
- **Importance of Normalizing the CM**:
  - Normalization involves dividing each value in the CM by the total number of images in the true class. This ensures that misclassification rates can be compared across classes, regardless of class imbalance.
  - Example: The diagonal in **row #5 and column #5** appears slightly darker compared to other digits. This could be due to:
    - The model making more errors on digit "5".
    - Fewer "5s" being present in the dataset compared to other digits.

Error Analysis – Confusion Matrices

- **Correct Classifications**:
  - Most images lie along the main diagonal of the CM, indicating they were classified correctly.
- **Importance of Normalizing the CM**:
  - Normalization involves dividing each value in the CM by the total number of images in the true class. This ensures that misclassification rates can be compared across classes, regardless of class imbalance.
  - Example: The diagonal in **row #5 and column #5** appears slightly darker compared to other digits. This could be due to:
    - The model making more errors on digit "5".
    - Fewer "5s" being present in the dataset compared to other digits.
- **Performance on Specific Digits**:
  - **Accuracy on Digit '5'**: Only **82%** of images of "5" were classified correctly.
  - **Asymmetry in the CM**:
    - The most common error for digit "5" was misclassification as "8", which occurred in **10%** of all cases involving "5".
    - Conversely, only **2%** of images of "8" were misclassified as "5".

Error Analysis – Confusion Matrices

To make the errors stand out more $\rightarrow$ put zero weight on the correct predictions:

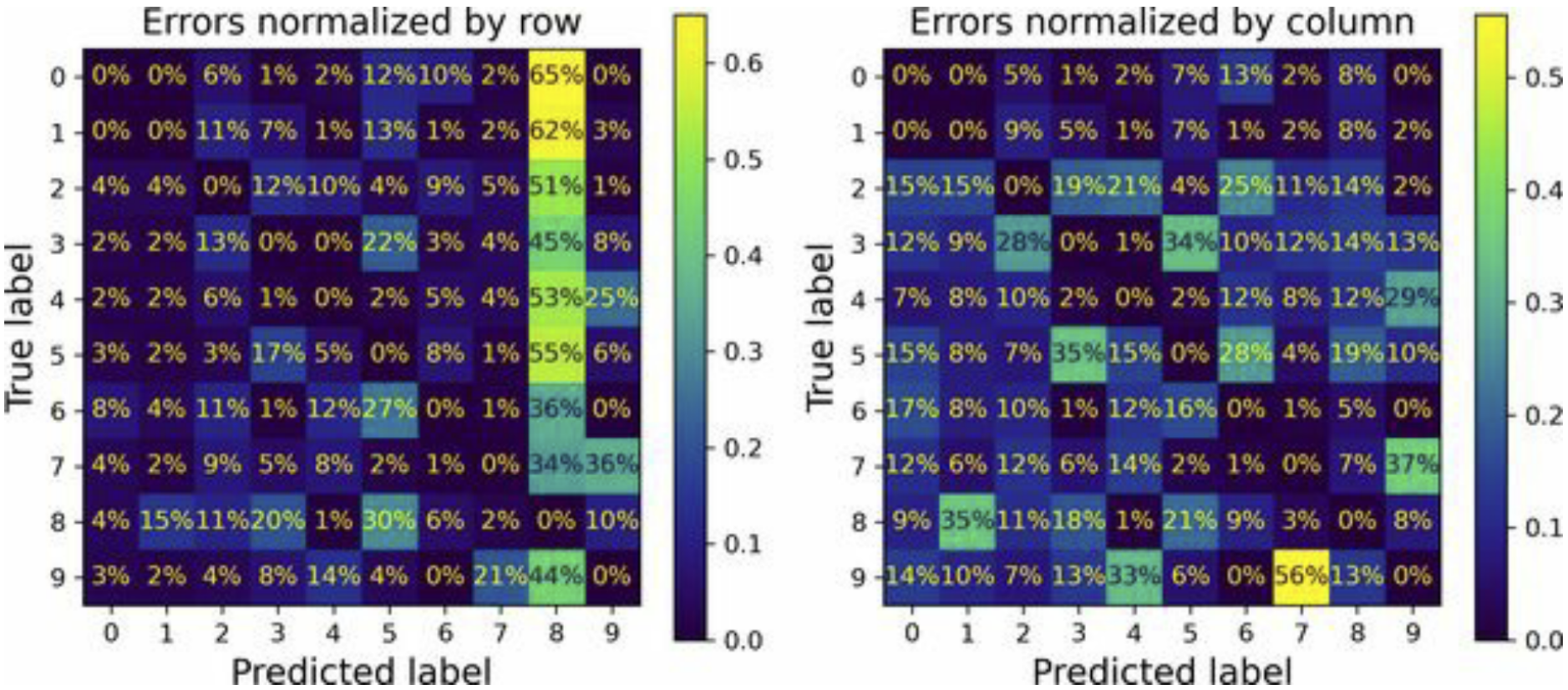


Figure: Confusion matrix with errors only, normalized by row (left) and by column (right)

Error Analysis – Confusion Matrices

Analyzing the CM to improve your classifier:

- Should try reducing the false 8s
  - try to gather more training data for digits that look like 8s (but are not) $\rightarrow$ the classifier can learn to distinguish them from real 8s
  - Or engineer new features that would help the classifier
    - writing an algorithm to count the number of closed loops (e.g., 8 has two, 6 has one, 5 has none)
  - Or preprocess the images (e.g., using `Scikit-Image`, `Pillow`, or `OpenCV`) $\rightarrow$ make some patterns, such as closed loops, stand out more
- ...

Error Analysis – Confusion Matrices

Or analyze individual errors rather than the whole CM:

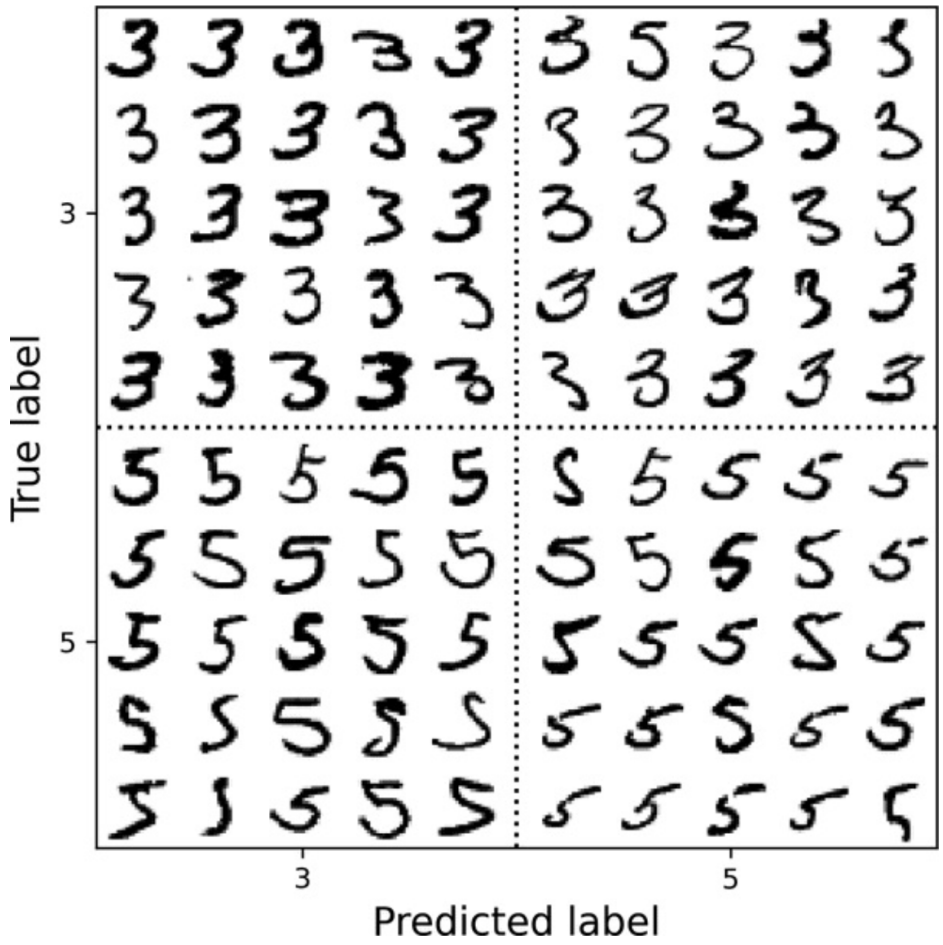


Figure: Some images of 3s and 5s organized like a confusion matrix

Error Analysis – Data Augmentation

When the model has a 3/5 confusion, it may be *too sensitive* to image *shifting and rotation*.

Idea: preprocess the images to ensure that they are well centered and not too rotated

BUT, this may not be easy since it requires predicting the correct rotation of each image!

Error Analysis – Data Augmentation

When the model has a 3/5 confusion, it may be *too sensitive* to image *shifting and rotation*.

Idea: preprocess the images to ensure that they are well centered and not too rotated

BUT, this may not be easy since it requires predicting the correct rotation of each image!

A better idea:

- Augment the training set with slightly shifted and rotated variants of the training images $\rightarrow$ forces the model to learn to be more tolerant to such variations
- This method is called **Data Augmentation** (to be discussed in details further in the further lectures)

Table of Contents

Resampling methods

- repeatedly drawing samples from a training set and refitting a model of interest on each sample $\rightarrow$ obtain additional information about the fitted model
- Allow us to obtain info that would not be available from fitting the model only once using the original training sample

1. Cross-validation
   - estimate the test error associated with a given statistical learning method to
     - evaluate its performance $\rightarrow$ Model assessment
     - select the appropriate level of flexibility $\rightarrow$ Model selection
2. Bootstrap
   - a measure of accuracy of a parameter estimate
   - a measure of accuracy of a statistical learning method

Cross-Validation

- Methods that estimate the test error rate by *holding out* a subset of the training observations from the fitting process
  - The Validation Set Approach
  - Leave-One-Out Cross-Validation
  - k-Fold Cross-Validation
    - Bias-Variance Trade-Off for k-Fold Cross-Validation
  - Cross-Validation on Classification Problems

The Validation Set Approach

Splitting your data into three sets: training, validation, and test sets

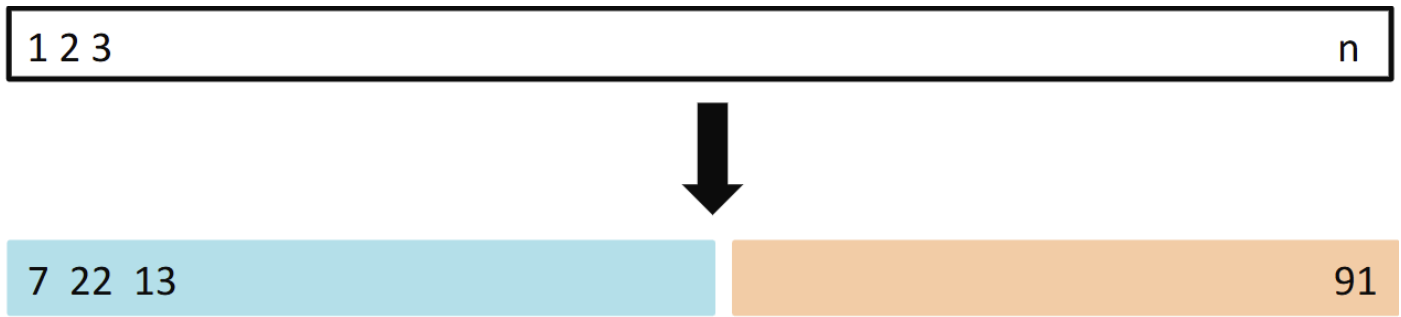


Figure: A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The learning method is fit on the training set, and its performance is evaluated on the validation set.

The Validation Set Approach

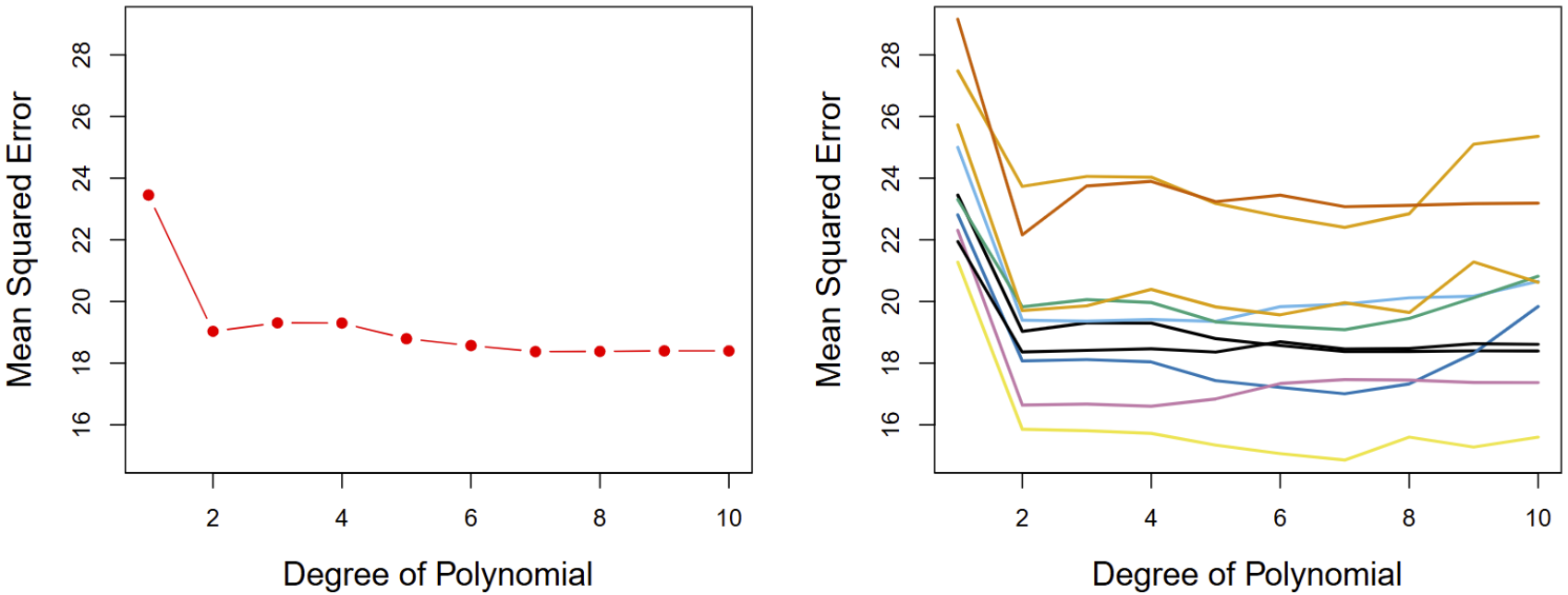


Figure: Left: Validation error estimates for a single split into training and validation data sets. Right: The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.

The Validation Set Approach Drawbacks

1. The validation estimate of the test error rate can be highly variable (as is shown in the right-hand panel of previous figure)

2. The validation set error rate may tend to *overestimate* the test error rate for the model fit on the entire data set (because it is now trained on fewer data)

Leave-One-Out Cross-Validation (LOOCV)

Like the validation set approach, LOOCV involves splitting the set of observations into two parts. BUT,

- Instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used for the validation set, and the remaining observations $\{(x_2, y_2), \cdots, (x_n, y_n)\}$ make up the training set

Leave-One-Out Cross-Validation (LOOCV)

Like the validation set approach, LOOCV involves splitting the set of observations into two parts. BUT,

- Instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used for the validation set, and the remaining observations $\{(x_2, y_2), \cdots, (x_n, y_n)\}$ make up the training set
- Repeat the procedure by selecting (x_2, y_2) for the validation data, training the statistical learning procedure on the $n-1$ observations $(x_1, y_1), (x_3, y_3), \cdots, (x_n, y_n)$, and computing $\mathrm{MSE}_2 = (y_2 - \hat{y_2})^2$

Leave-One-Out Cross-Validation (LOOCV)

Like the validation set approach, LOOCV involves splitting the set of observations into two parts. BUT,

- Instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used for the validation set, and the remaining observations $\{(x_2, y_2), \cdots, (x_n, y_n)\}$ make up the training set
- Repeat the procedure by selecting (x_2, y_2) for the validation data, training the statistical learning procedure on the $n-1$ observations $(x_1, y_1), (x_3, y_3), \cdots, (x_n, y_n)$, and computing $\mathrm{MSE}_2 = (y_2 - \hat{y_2})^2$
- The LOOCV estimate for the test MSE is the average of these n test error estimates:

$$\mathrm{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{MSE}_i$$

Leave-One-Out Cross-Validation (LOOCV)

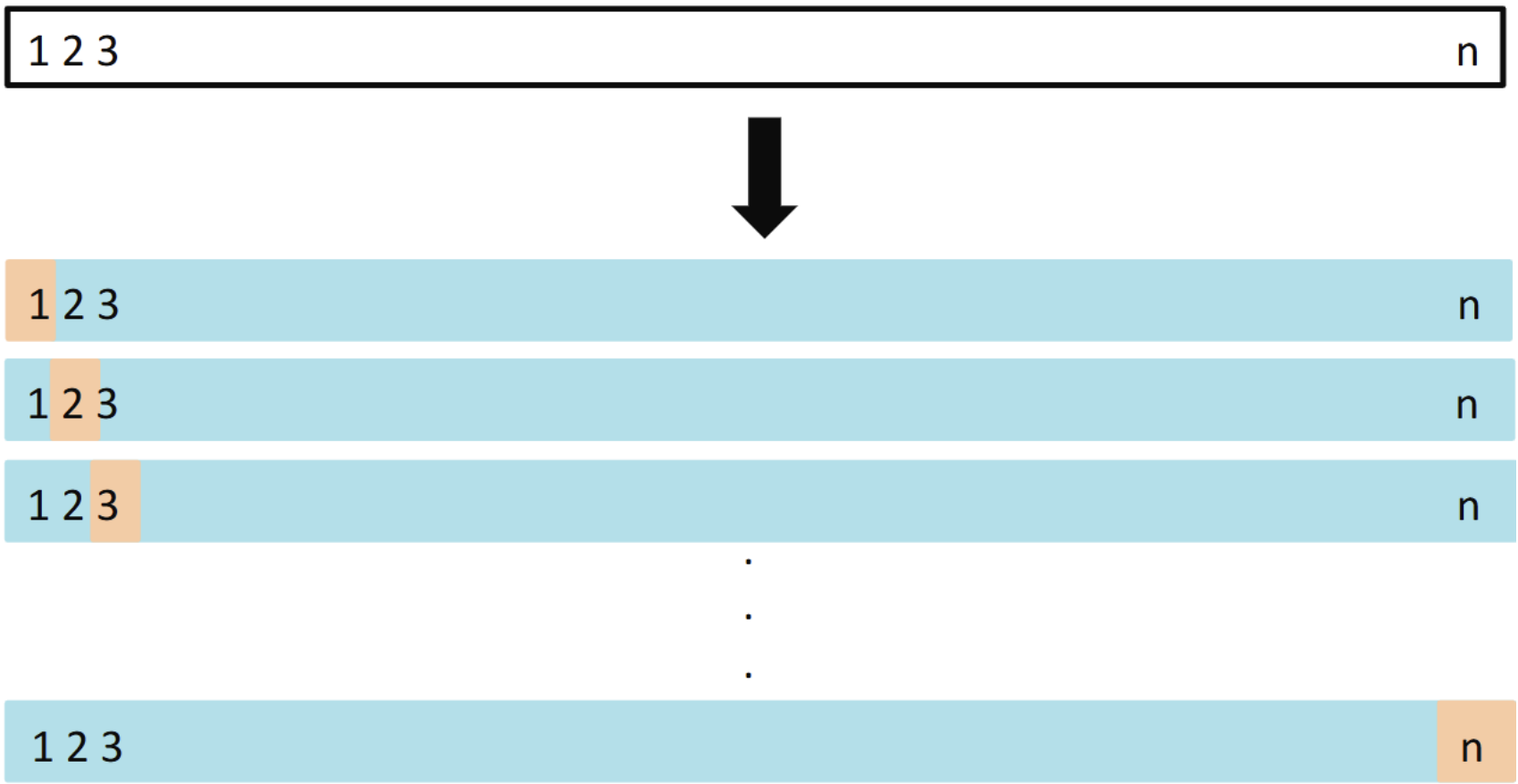


Figure: A schematic display of LOOCV.

Leave-One-Out Cross-Validation (LOOCV)

LOOCV vs. The Validation Set Approach

- LOOCV has far less bias since it repeatedly fit the learning method using training sets that contain $n-1$ observations
- LOOCV tends not to overestimate the test error rate
- Performing LOOCV multiple times will always yield the same results: there is no randomness in the training/validation set splits
- HOWEVER,

Leave-One-Out Cross-Validation (LOOCV)

LOOCV vs. The Validation Set Approach

- LOOCV has far less bias since it repeatedly fit the learning method using training sets that contain $n-1$ observations
- LOOCV tends not to overestimate the test error rate
- Performing LOOCV multiple times will always yield the same results: there is no randomness in the training/validation set splits
- HOWEVER,
  - LOOCV has the potential to be expensive to implement since the model has to be fit n times
  - and can be very time consuming if n is large, and if each individual model is slow to fit

k-Fold Cross-Validation

An alternative to LOOCV

- randomly dividing the set of observations into k groups, or folds, of approximately equal size
- The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ fold
- The MSE_1 is then computed on the observations in the held-out fold
- Repeat the procedure k times
  - each time, a different group of observations is treated as a validation set
- The k-fold CV estimate is then:

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$

k-Fold Cross-Validation

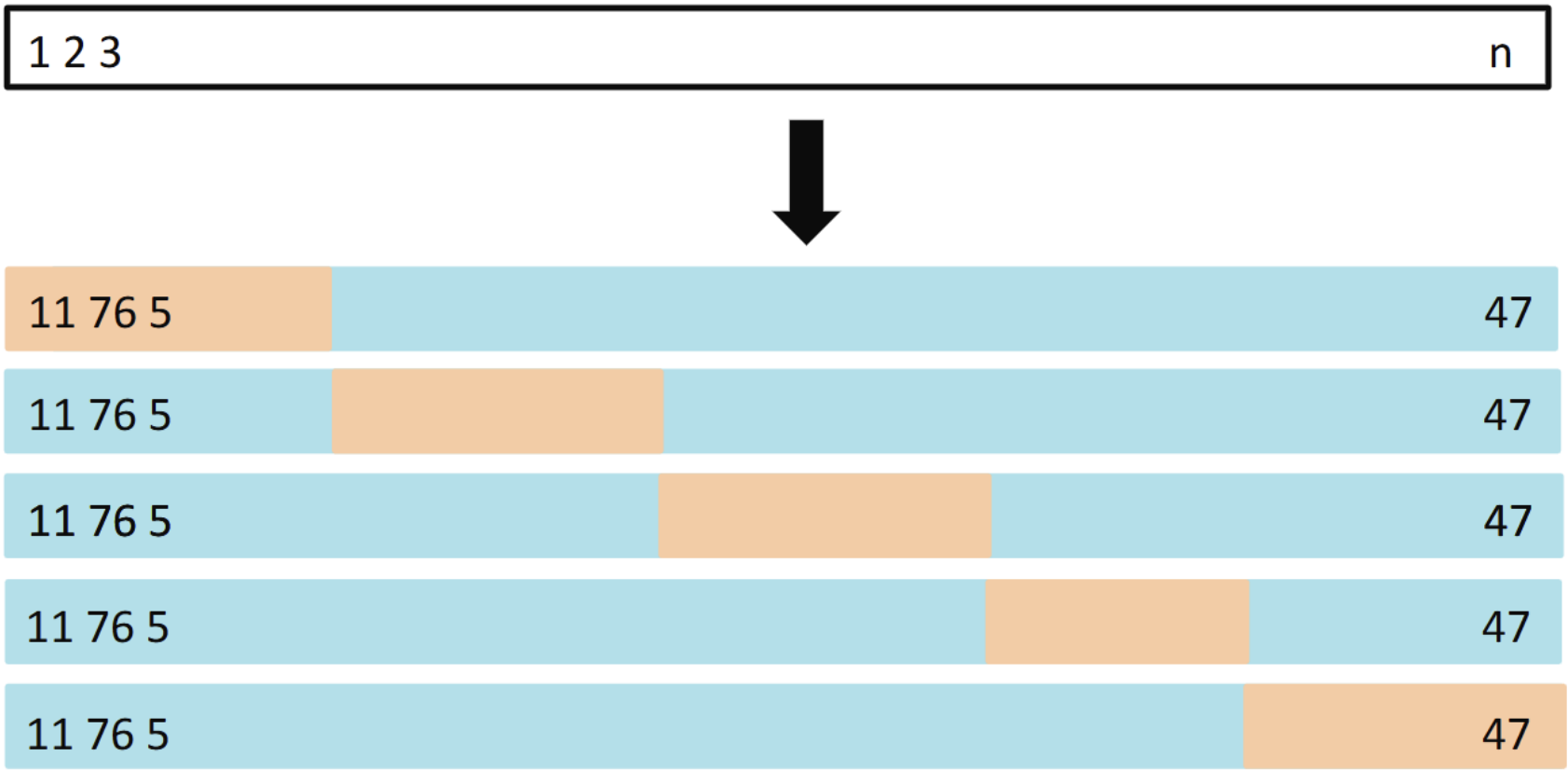


Figure: A schematic display of 5-fold CV.

k-Fold CV vs. LOOCV

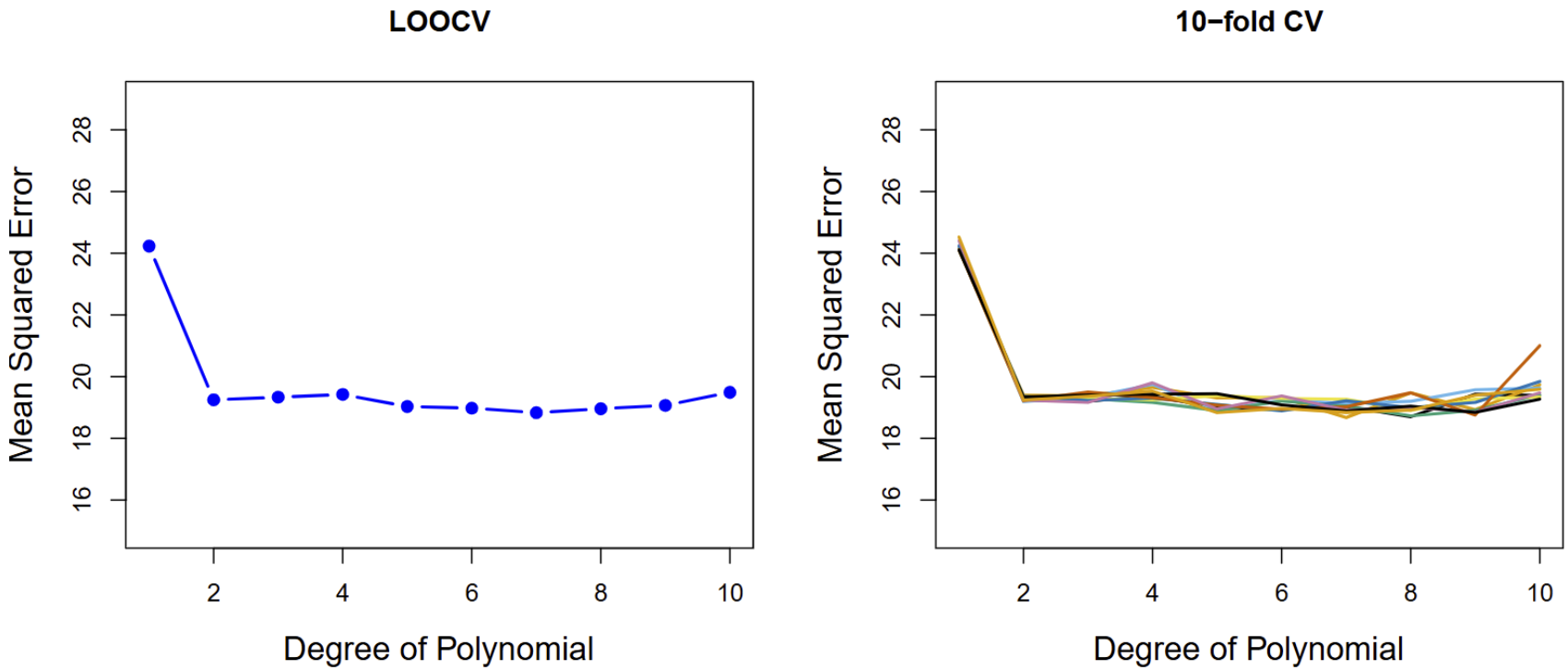


Figure: Left: The LOOCV error curve. Right: 10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.

k-Fold CV vs. LOOCV

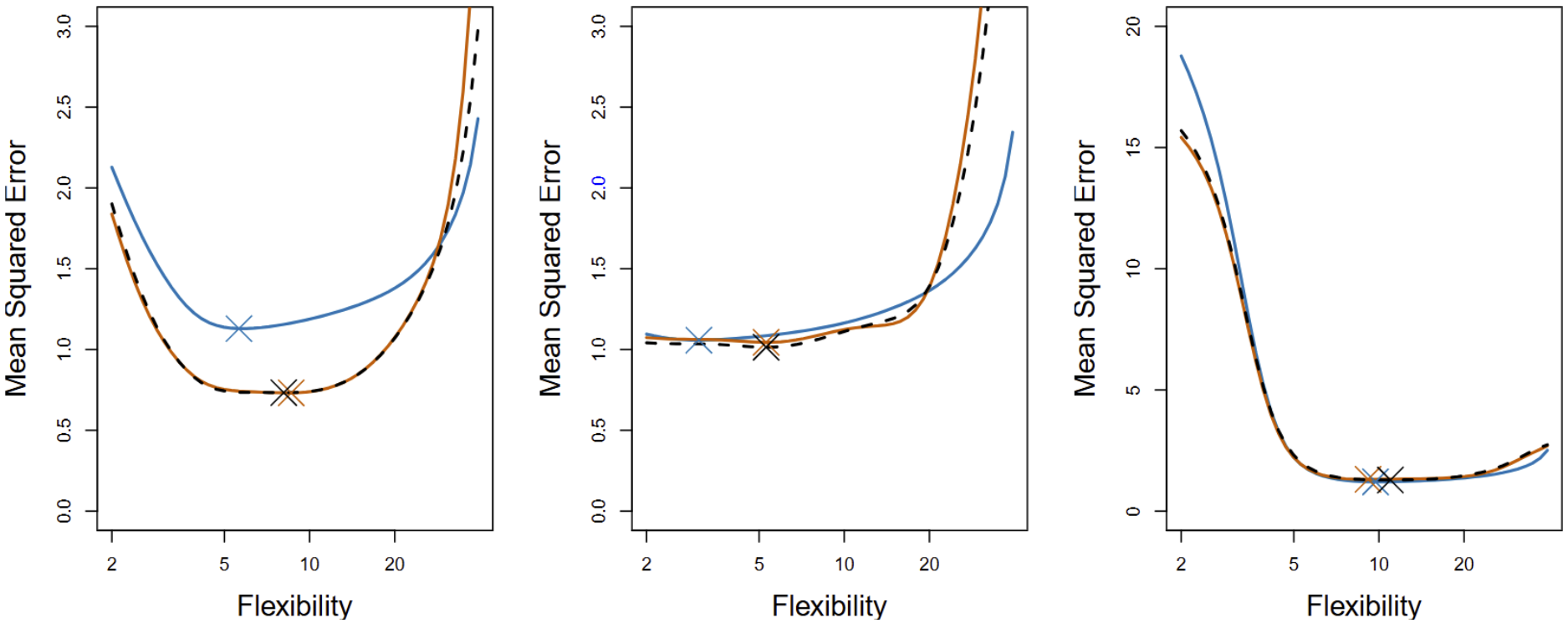


Figure: True and estimated test MSE for the simulated data sets in left, center, and right. The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.

Bias-Variance Trade-Off for k-Fold Cross-Validation

- k-fold CV with $k < n$ has a computational advantage to LOOCV
- From the perspective of *bias* reduction:
  it is clear that LOOCV is to be preferred to k-fold CV
- To consider the procedure's variance:
  LOOCV has higher variance than does k-fold CV with $k < n$
  (averaging the outputs of n fitted models, having highly correlated outputs)

Given the bias-variance trade-off considerations, one should performs k-fold cross-validation using $k = 5$ or $k = 10$

- these values have been shown empirically to yield test error rate estimates
- suffer neither from excessively high bias nor from very high variance

Cross-Validation on Classification Tasks

So far: regression settings $\rightarrow$ quantitative outcome $\rightarrow$ MSE

Cross-Validation on Classification Tasks

So far: regression settings $\rightarrow$ quantitative outcome $\rightarrow$ MSE

For classification tasks:
the number of misclassified observation (rather than MSE) is used to quantify error:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{Err}_i$$

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{Err}_i$$

where $\mathrm{Err}_i = \mathrm{I}(y_i \neq \hat{y}_i)$

Cross-Validation on Classification Tasks

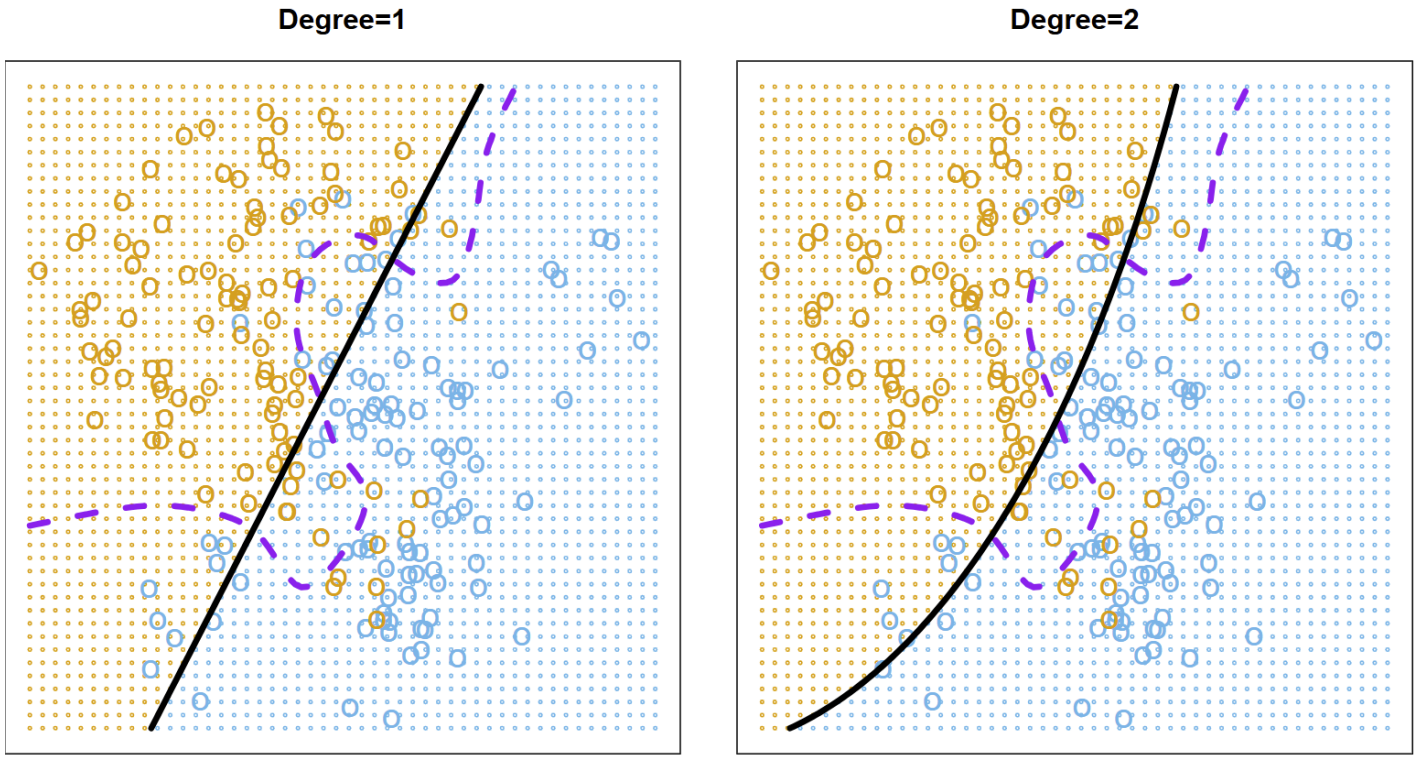


Figure: Logistic regression fits on the 2d classification data. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from degrees 1–4 logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively 0.201, 0.197, 0.160, and 0.162.

Cross-Validation on Classification Tasks

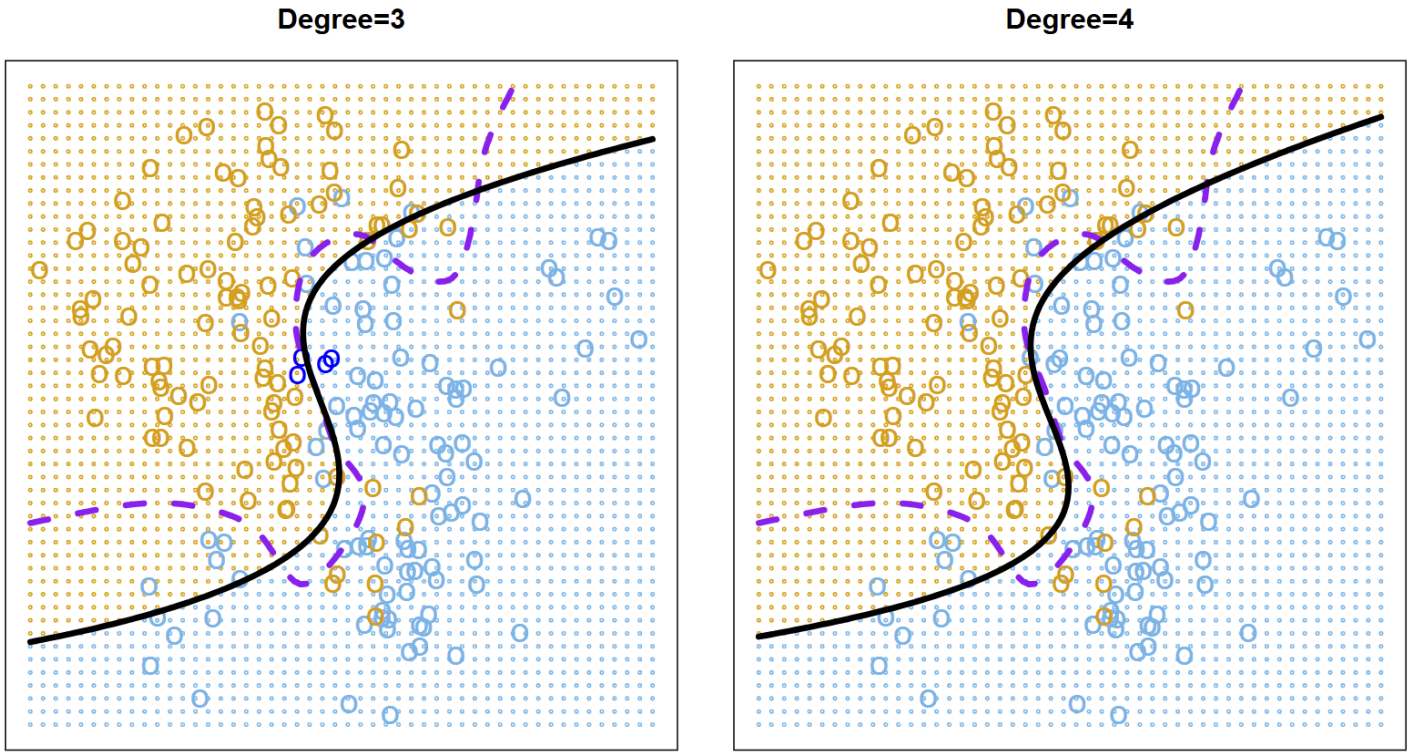


Figure: Logistic regression fits on the 2d classification data. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from degrees 1–4 logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively 0.201, 0.197, 0.160, and 0.162.

Cross-Validation on Classification Tasks

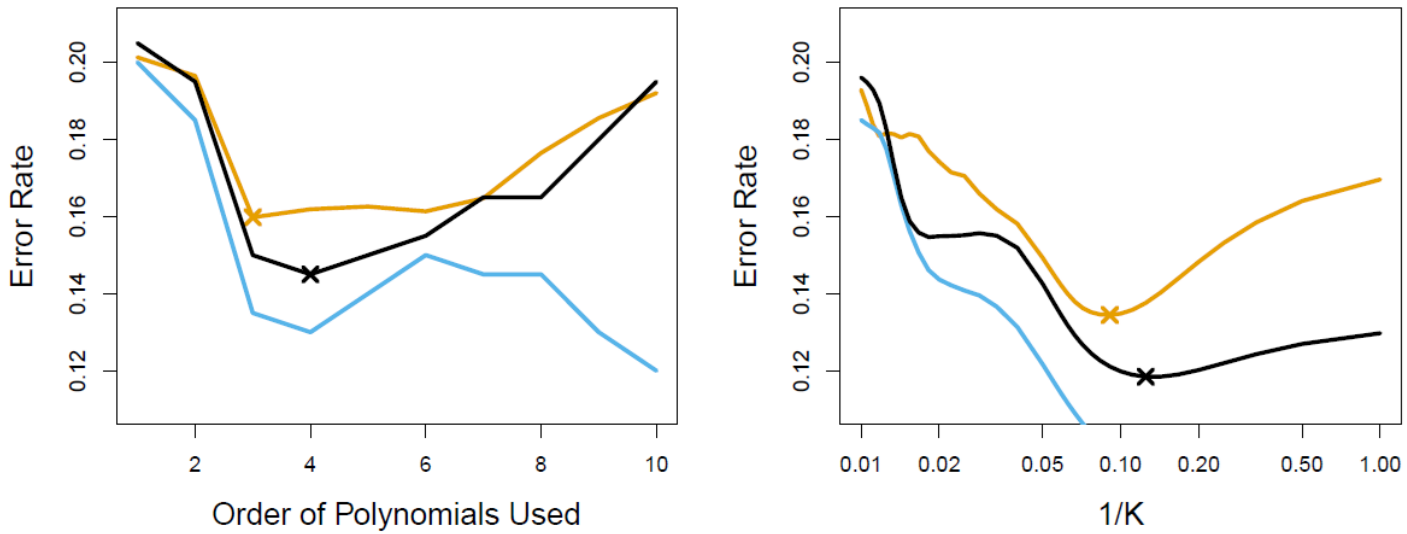


Figure: Test error (brown), training error (blue), and 10-fold CV error (black) on the two-dimensional classification data. Left: Logistic regression using polynomial functions of the predictors. The order of the polynomials used is displayed on the x-axis. Right: The KNN classifier with different values of K, the number of neighbors used in the KNN classifier.

Bootstrap

What is The Bootstrap?

- A powerful resampling technique to quantify uncertainty.
- Used to estimate the variability (e.g., standard error) of a statistic.
- Applicable even when analytical solutions are hard or unavailable.
- Relies on sampling **with replacement** from the original data.

Goal: Assess accuracy of estimators like the mean, regression coefficients, or optimal investment fractions.

Toy Example: Optimal Investment Allocation

Suppose we invest a fraction α of wealth in asset X and $1-\alpha$ in asset Y:

$$\text{Var}(\alpha X + (1-\alpha)Y) \Rightarrow \text{minimize risk}$$

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}} \quad \text{(true)}$$

Since σ's are unknown, we estimate:

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

Use Bootstrap to estimate the variability of $\hat{\alpha}$!

Toy Example continued

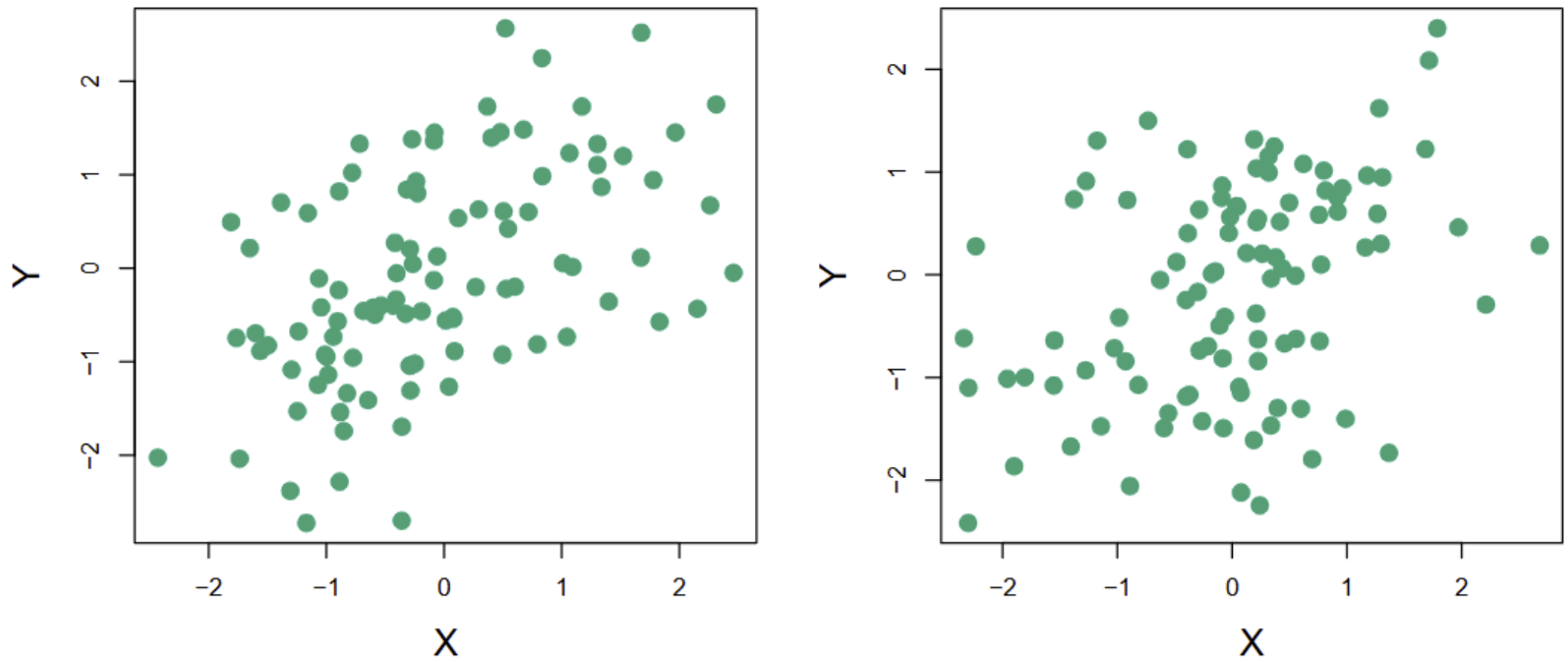


Figure: Each panel displays 100 simulated returns (obtain the σ_X^2, σ_Y^2, and σ_{XY}) for investments X and Y. From left to right, the resulting estimates for α are 0.576 and 0.532

Toy Example continued

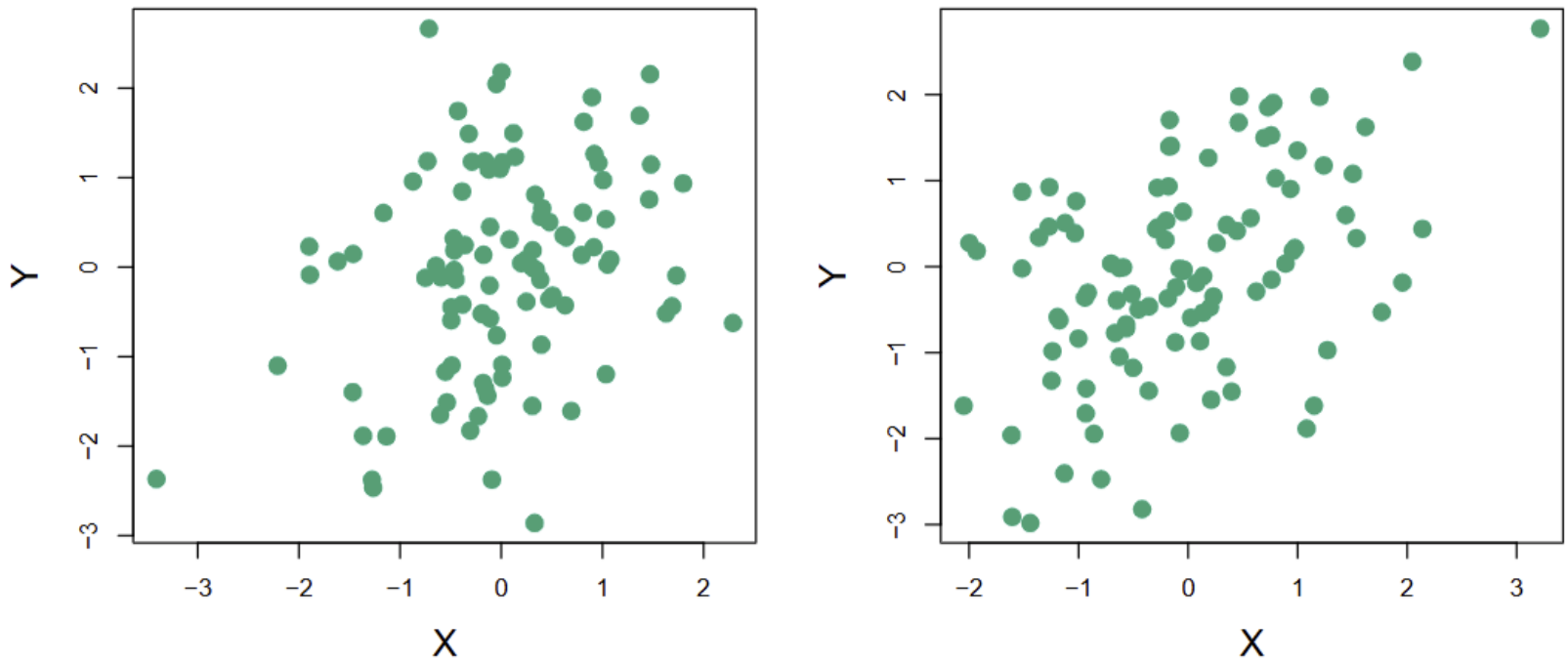


Figure: Each panel displays 100 simulated returns (obtain the σ_X^2, σ_Y^2, and σ_{XY}) for investments X and Y. From left to right, the resulting estimates for α are 0.657 and 0.651

Toy Example continued

- The mean over all $1,000$ estimates for α:

$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996,$$

- The standard deviation of the estimates is

$$\mathrm{SE}(\hat{\alpha}) = \sqrt{\frac{1}{1000-1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} \approx 0.083$$

Note that the above procedure is for estimating $\mathrm{SE}(\hat{\alpha})$ cannot be applied in practice! (why?)

Bootstrap Estimation Procedure

1. From a dataset of n observations, sample n points with replacement $\Rightarrow$ bootstrap dataset Z^*.
2. Compute $\hat{\alpha}^*$ for each Z^*.
3. Repeat for B bootstrap samples $\Rightarrow \hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \ldots, \hat{\alpha}^{*B}$.
4. Compute bootstrap standard error:

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}$$

Bootstrap Estimation Procedure – Toy Example

- True value: $\alpha = 0.6$
- 1,000 estimates from simulated data: $\bar{\alpha} = 0.5996$, SE ≈ 0.083
- Bootstrap estimate: $\text{SE}_B(\hat{\alpha}) = 0.087$

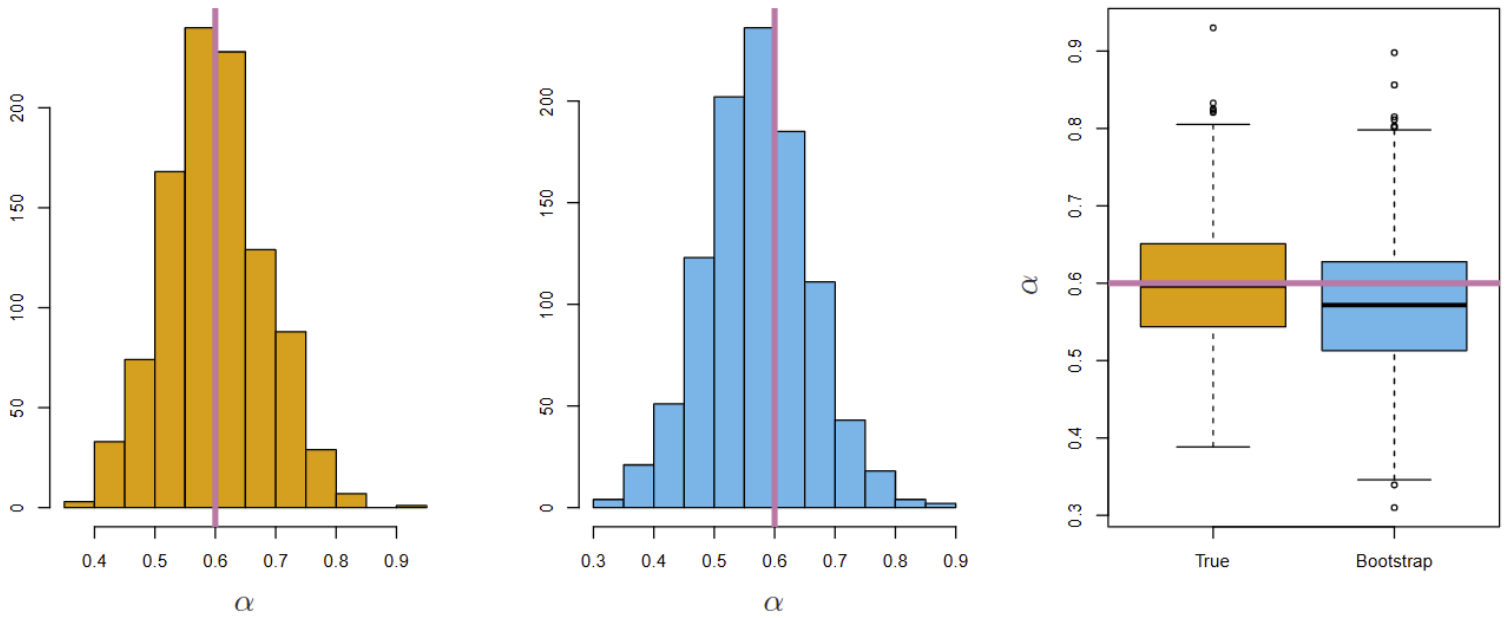


Figure: Histograms & boxplots of $\hat{\alpha}$ using simulation and bootstrap

Visualizing the Bootstrap

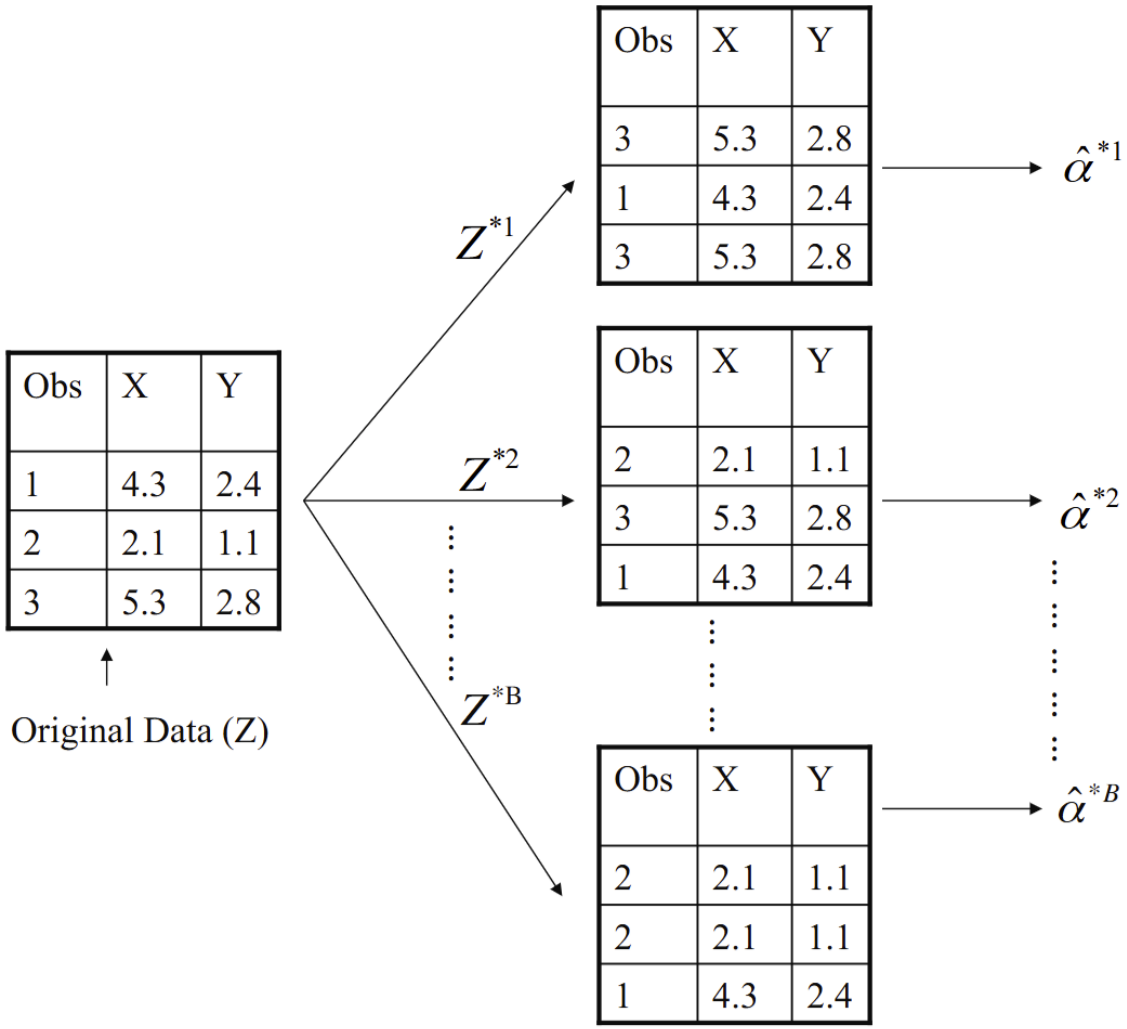


Figure: Each bootstrap sample (with replacement) yields an estimate of $\hat{\alpha}^*$

Principles of Machine Learning

Lecture 8: Data Preparation

Sharif University of Technology
Dept. of Aerospace Engineering

April 27, 2025

Data Preparation

Feature Engineering

To handle irrelevant features

- *Feature selection* (selecting the most useful features)
- *Feature extraction* (combining existing features to produce a useful one)
- Gathering new data $\longrightarrow$ Creating new features

Overfitting the Training Data

Performing well on training data but not generalizing well
Complex relative to the amount and noisiness of the training data
Possible solutions:

- Simplifying or constraining the model
- Gather more training data
- Reduce the noise in the training data

Underfitting the Training Data

Too simple to learn the underlying structure of data
Possible solutions:

- Select a model with more parameters
- Feed better features (feature engineering)
- Reduce the constraints

Data Preparation

Testing and Validating

Hyperparameter Tuning and Model Selection

- Train the model on the *training set*
- Test the model on the *test set*
- *Generalization error*: The error rate on new cases
- Evaluate the model on the test set $\longrightarrow$ Estimating the generalization error
- The value above indicates how good your model performs on unseen data
- Common 80-20 splitting

But, what if the model is the best model only for *that particular set?*

Testing and Validation

Hyperparameter Tuning and Model Selection

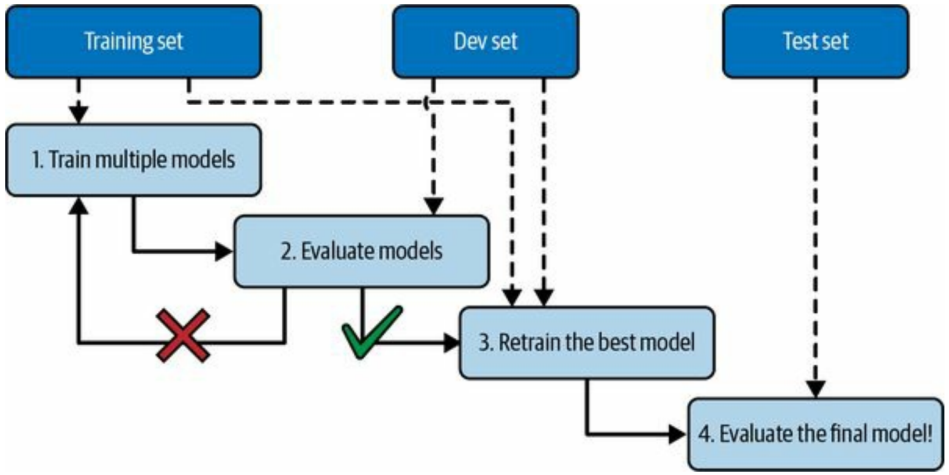


Figure: Model selection using holdout validation

Validation set, too small or too large $\longrightarrow$ cross validation

Data Mismatch

- Representativity of the test and validation (development) sets
- How to decide the poor performance is due to model overfitting or data mismatch $\longrightarrow$ using train-dev set

Overview of an ML Project

1. Look at the big picture & define the problem
2. Get the data
3. Explore and visualize the data $\longrightarrow$ gain insights
4. Prepare and clean data for the ML algorithm
5. Select and train a model (or models)
6. Evaluate the model & fine-tune
7. Present your solution
8. Deploy, monitor, and maintain your system

Working with Real Data

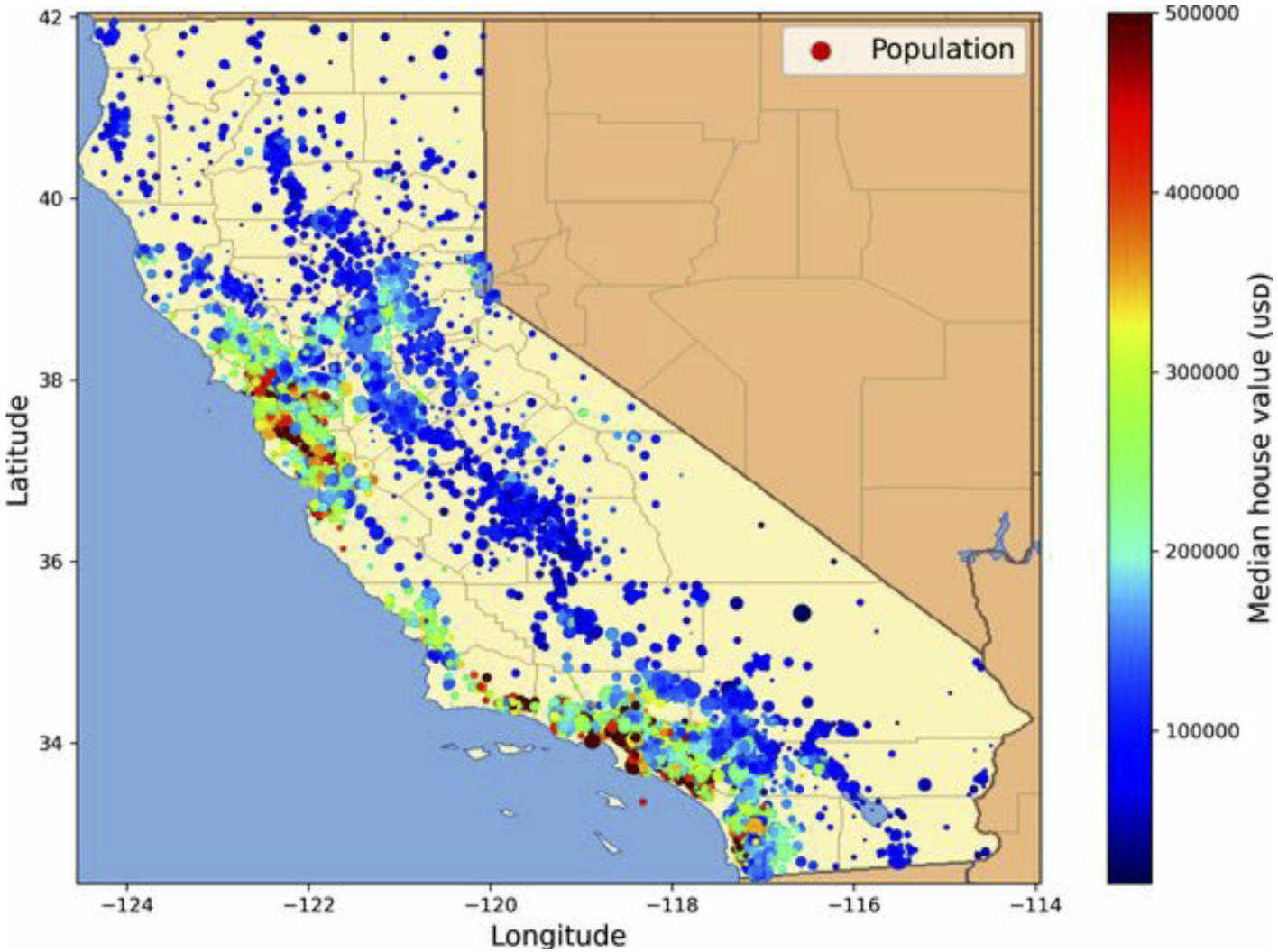
- **Popular open data repositories:**
  - OpenML.org
  - Kaggle.com
  - PapersWithCode.com
  - UC Irvine Machine Learning Repository
  - Amazon's AWS datasets
  - TensorFlow datasets
- **Meta portals (they list open data repositories):**
  - DataPortals.org
  - OpenDataMonitor.eu
- **Other pages listing many popular open data repositories:**
  - Wikipedia's list of machine learning datasets
  - Quora.com
  - The datasets subreddit

Example of a Project

The California Housing Prices dataset from the StatLib repository

- Based on data from the 1990 California census
- Not exactly recent (a nice house in the Bay Area was still affordable at the time)
- Will pretend it is recent data for learning purposes

Example of a Project

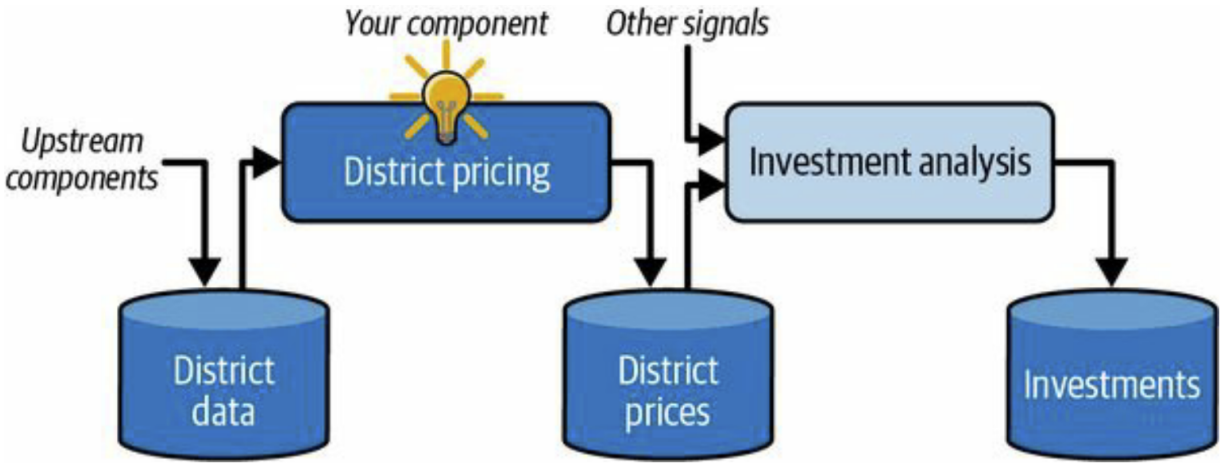


Look at The Big Picture

- Use California census data $\longrightarrow$ build a model of housing prices in the state
- This data includes, for each distinct,
  - Population
  - Median income
  - Median housing price
- The model should learn from this data $\longrightarrow$ predict the median housing price in any district

Frame the Problem

- What is the objective
- Knowing the objective is important because it will determine
  - how you frame the problem
  - which algorithms you will select
  - which performance measure you will use to evaluate your model
  - how much effort you will spend tweaking it
- Your model's output may be fed to another ML system:



Check the Assumptions

- List and verify the assumptions that have been made so far (by you or others)
- Example:
  - The output is the price $\longrightarrow$ regression task
    OR
  - But what if the downstream system converts the prices into categories (e.g., “cheap”, “medium”, or “expensive”) and uses those categories instead of the prices themselves?
    Your system just needs to get the category right.
    The problem should have been framed as a classification task, not a regression task.

Explore and Visualize the Data

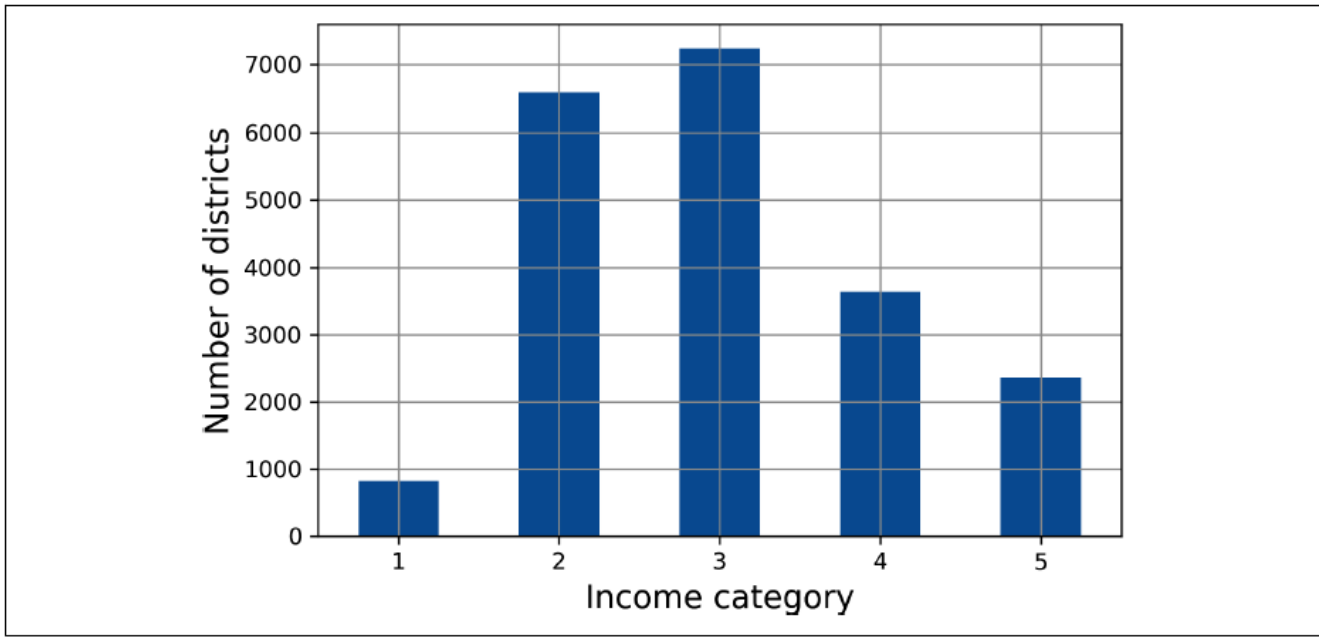
- Use histograms, scatter plots, correlation matrices
- Look for:
  - Skewed distributions
  - Outliers
  - Missing values

Data Cleaning and Preparation

- Handle missing values
- Encode categorical features
- Feature scaling (standardization)

Train-Test Splits

- Use random versus stratified splits
  Suppose that the median *income* is a very important attribute to predict median housing prices. You want to ensure that the test set is representative of the various categories of incomes in the whole dataset.
- Stratify by income category, then split them



Feature Engineering

- Create new informative features:
  - Rooms per household
  - Population per household
  - Income category

Select and Train Models

- Start with baseline models:
  - Linear regression
  - Decision trees
  - Random forests
- Use Scikit-Learn pipelines for efficiency

Evaluate and Fine-Tune

- Use cross-validation for reliable metrics
- Tune Hyper-parameters using GridSearchCV or RandomizedSearchCV
  - Randomized search explores a wide range of values for hyperparameters (e.g., 1,000 iterations can test 1,000 values), while grid search limits exploration to predefined values.
  - If a hyperparameter has little effect on performance, grid search still trains for all its possible values (e.g., 10 values = 10x longer training), but randomized search ignores its impact.
  - For large parameter spaces (e.g., 6 hyperparameters with 10 values each), grid search requires exhaustive training (1 million combinations), whereas randomized search allows flexible control over the number of iterations.
- Feature importance analysis

Launch and Monitor

- Deploy to production
- Monitor data drift and model performance
- Periodically retrain on fresh data

Some Visualizations of the Data

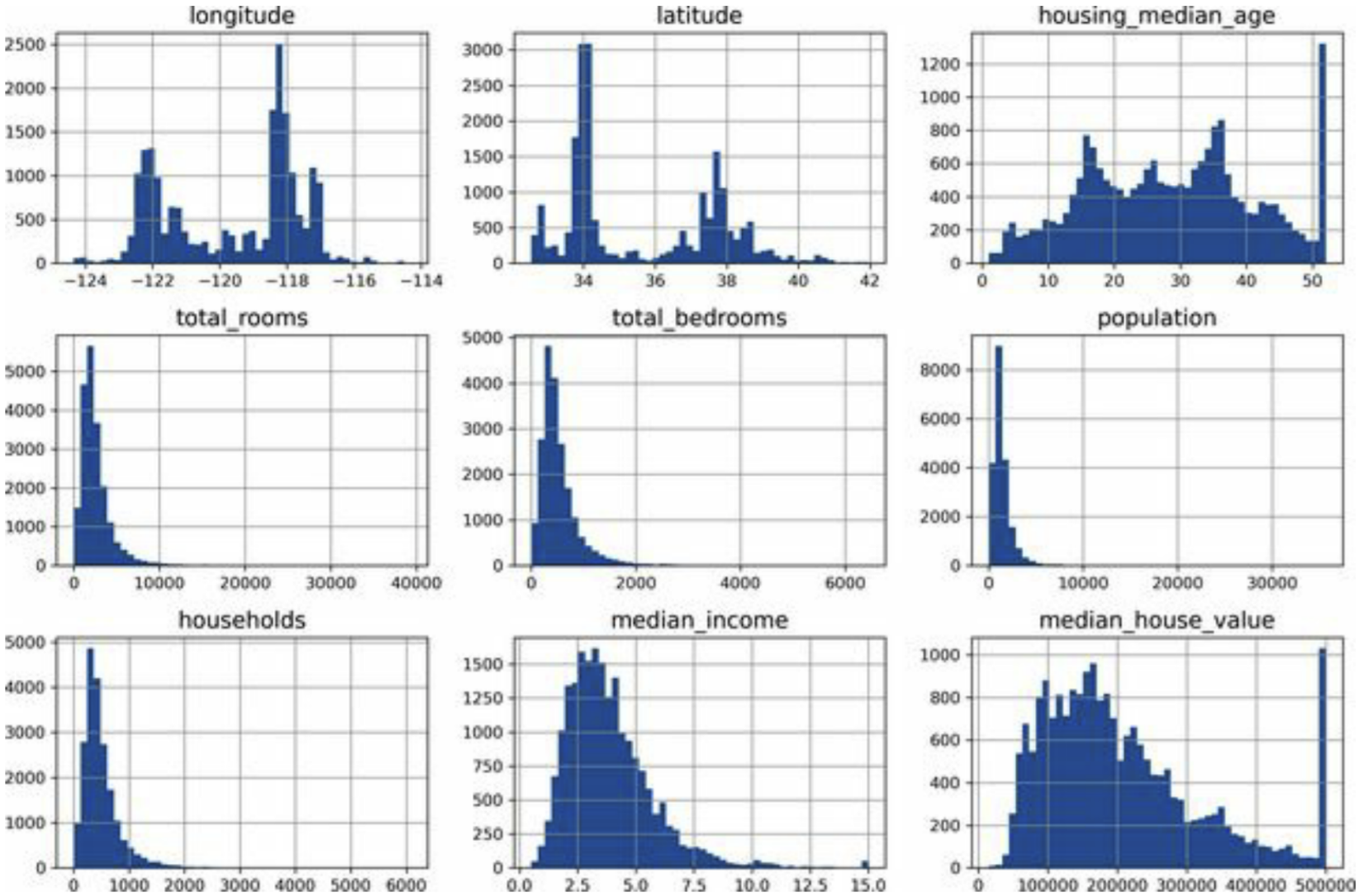
Quick Look at the Data

```
housing.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	500001.000000

Some Visualizations of the Data

Quick Look at the Data



Some Visualizations of the Data

Correlation Determination using Scatter Matrices