

# Statistical Computing

Michael Mayer

March 2023



# Statistical Computing: What will we do?

## Chapters

1. R in Action
2. Statistical Inference
3. Linear Models
4. Model Selection and Validation
5. Trees
6. Neural Nets

## Remarks

- ▶ Chapters 3 to 6:  
Statistical ML in Action
- ▶ Two weeks per chapter
- ▶ Exercises at end of chapter notes

# Model Selection and Validation

## Two Questions

- ▶ “How good is our model?”
- ▶ “Which model to choose among alternatives?”

### Problem and solution

- ▶ “In-sample” performance is biased
- ▶ Overfitting should not be rewarded
- ▶ Use data splitting to get fair results

### Notation

- ▶ Total loss  $Q(f, D) = \sum_{(y_i, \mathbf{x}_i) \in D} L(y_i, f(\mathbf{x}_i))$
- ▶ Average loss  $\bar{Q}(f, D) = Q(f, D)/|D|$
- ▶ Performance measure or evaluation metric  $S(f, D)$  of interest, often  $S = \bar{Q}$  or a function of it

# Outline

- ▶ Nearest-Neighbor
- ▶ Simple Validation
- ▶ Cross-Validation
- ▶ Test Data and Final Workflow
- ▶ Excursion: SQL and Spark

## Excursion: $k$ -Nearest-Neighbor ( $k$ -NN)

- ▶ Alternative to linear model
- ▶ How does it work?
- ▶ Classification and regression
- ▶ Standardization?

### Example

# Simple Validation

- ▶ In-sample, 1-NN would win any comparison!?
- ▶ Split data into training and validation sets  $D_{\text{train}}$  and  $D_{\text{valid}}$ , e.g., 80%/20%
- ▶ Use performance  $S(\hat{f}, D_{\text{valid}})$  on validation set to make decisions (choose models, choose parameters like  $k$ )
- ▶ Measure amount of overfitting/optimism by

$$S(\hat{f}, D_{\text{valid}}) - S(\hat{f}, D_{\text{train}})$$

## Example

# K-fold Cross-Validation (CV)

Simple validation is neither economic nor robust, except for large data

## Algorithm

1. Split the data into  $K$  pieces  $D = \{D_1, \dots, D_K\}$  called “folds”. Typical values for  $K$ ?
2. Set aside one of the pieces ( $D_k$ ) for validation
3. Fit model  $\hat{f}_k$  on  $D \setminus D_k$
4. Calculate performance  $\hat{S}_k = S(\hat{f}_k, D_k)$
5. Repeat Steps 2 – 4 for each  $k$
6. Calculate **CV performance**  $\hat{S}_{CV} = \frac{1}{K} \sum_{k=1}^K \hat{S}_k$

## Remarks

- ▶ How to choose and fit best/final model?
- ▶ What means «best»?
- ▶ Stability of results?
- ▶ Repeated CV?

## Example



# Hyperparameter Tuning

- ▶ Choosing  $k$  in  $k$ -NN is example of “hyperparameter tuning”
- ▶ Algorithms with more than 1 hyperparameter?
- ▶ Grid Search CV
- ▶ Randomized Search CV

# Test Data and Final Workflow

## Problematic consequence of model tuning?

- ▶ **Overfitting** on validation data or on CV!
- ▶ Performance of final model? → **Test data**

## Workflow A

1. Split data into train/valid/test, e.g., by ratios 60%/20%/20%
2. Train different models on training data and assess performance on validation data. Choose best model, re-train on training + validation data, and call it “final model” (Simplification?)
3. Assess performance of final model on test data

## Workflow B

1. Split data into train/test, e.g., by ratios 80%/20%.
2. Evaluate and tune different models by *K*-fold CV on training data. Choose best model, re-train on full training data
3. Assess performance of final model on test data

## Example of Workflow B

When test data not necessary?

## Ridge Regression

- ▶ Example of **penalized** regression
- ▶ Model equation similar to usual linear regression

$$\mathbb{E}(Y \mid \mathbf{x}) = f(\mathbf{x}) = \beta_0 + \beta_1 x^{(1)} + \dots + \beta_p x^{(p)}$$

- ▶ But with penalized least-squares objective

$$Q(f, D_{\text{train}}) = \sum_{(y_i, \mathbf{x}_i) \in D_{\text{train}}} (y_i - f(\mathbf{x}_i))^2 + \lambda \sum_{j=1}^p \beta_j^2$$

- ▶  $L_2$  penalty pulls coefficients slightly towards 0, fighting overfitting
- ▶  $\lambda_{\text{opt}} \geq 0$  with best (cross-)validation result  $\rightarrow$  use to fit final model
- ▶ Intercept? Standardization?  $L_1$ ? Elastic-net?

## Example

**Random splits**

**Grouped splits**

**Time-Series splits**

**Stratified splits**

## Excursion: SQL and Spark

*Data science is 80% preparing data, 20% complaining about preparing data.*

### Typical preprocessing steps?

#### Good moment to learn

- ▶ data structure
- ▶ meaning of columns
- ▶ sources of bias

### How to do preprocessing?

Data = files on disk or tables in database

- ▶ If small → R/Python
- ▶ If large? → Database Management System (DBMS) or Spark
- ▶ Communication via SQL

# SQL

## Structured Query Language

- ▶ Pronounced?
- ▶ Important in data science
- ▶ In DBMS or R/Python
- ▶ ISO norm  $\leftrightarrow$  dialects
- ▶ SQL queries

## Learn SQL with examples

- ▶ Diamonds (from memory)
- ▶ Taxi (from Parquet)

## DuckDB (since 2018)

- ▶ In-process, open-source DBMS
- ▶ Easy to install in R/Python
- ▶ No dependencies (Java etc.)
- ▶ Fast
- ▶ Out-of-core capabilities

# Apache Spark

- ▶ Distributed, open-source cluster computing system for big data
- ▶ Apache project since 2013
- ▶ Heavily used in industry
- ▶ Written in Scala
- ▶ Contains SQL engine
- ▶ Can be used from R/Python

## Examples

- ▶ Diamonds (from memory)
- ▶ Taxi (from Parquet)