Statistics and Computing

Series Editor

Wolfgang Karl Härdle, Humboldt-Universität zu Berlin, Berlin, Germany

Statistics and Computing (SC) includes monographs and advanced texts on statistical computing and statistical packages.

Giovanni Cerulli

Fundamentals of Supervised Machine Learning

With Applications in Python, R, and Stata

Springer

Giovanni Cerulli
IRCRES-CNR, Research Institute for
Sustainable Economic Growth
National Research Council of Italy
Rome, Italy

ISSN 1431-8784 ISSN 2197-1706 (electronic)
Statistics and Computing
ISBN 978-3-031-41336-0 ISBN 978-3-031-41337-7 (eBook)
https://doi.org/10.1007/978-3-031-41337-7

© The Editor(s) (if applicable) and The Author(s), under exclusive licence to Springer Nature
Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

To the memory of my grandmother Esterina and aunt Letizia who silently but constantly accompanied the writing of this book. For the good, the wisdom, and the knowledge they pass on to me. For their priceless and everlasting love.

Preface

Scope of the Book

This is my second book published with Springer. The first, *Econometrics of Program Evaluation: Theory and Applications* (second edition), deals with causal inference for policy impact assessment and has a strong inferential character. This second book, dealing with supervised Machine Learning, largely complements the first in many regards, in that it privileges predictive analytics over inferential analysis. Inference and prediction are however both essential for data science and analysis, as they shed light on two different, although increasingly linked, modes of scientific investigation.

This book presents the fundamental theoretical notions of supervised Machine Learning along with a wide range of applications with Python, R, and Stata software. The book is dedicated to Ph.D. students, academics, and practitioners in various fields of study (social sciences, medicine, epidemiology, as well as hard sciences) who intend to learn the fundamentals of supervised Machine Learning to apply it to concrete case studies.

The book assumes the reader to have a good understanding of basic statistics (both descriptive and inferential), the meaning and the writing of algorithms, and a working knowledge of Python, R, or Stata software. Mathematics is used only when strictly necessary, and a large focus is paid to graphical explanations instead of analytic proofs.

Why should the reader consider this book as a valuable source of knowledge for learning theoretical and applied supervised Machine Learning? Here are three compelling reasons:

1. *Comprehensive coverage of theoretical foundations.* The book starts by presenting the fundamental theoretical notions of supervised Machine Learning. It covers key concepts such as regression, classification, ensemble methods, and evaluation metrics, providing a solid foundation for understanding the principles and techniques behind supervised learning. This theoretical grounding ensures that readers gain a deep understanding of the subject matter before moving on to practical applications.

2. *Diverse applications and case studies.* The book goes beyond theory and offers a wide range of applications using three popular software packages: Python, R, and Stata. By including multiple software options, the book caters to the diverse needs and preferences of readers. Python, known for its versatility and extensive libraries like scikit-learn, allows readers to implement Machine Learning algorithms and explore advanced techniques. R, a widely used language in statistics and data analysis, provides readers with additional tools and packages—such as CARET—specifically tailored for statistical learning modeling and results' visualization. Stata, a statistical software widely used in social sciences, medicine, and epidemiology, offers readers an alternative perspective with its unique features and capabilities. By including all three software packages, this book ensures that readers can leverage the software they are most comfortable with or choose the one best suited to their specific domain.
3. *Differentiating factors among other books.* While there are indeed several good books available on Statistical Learning and software, this book stands out by addressing the specific needs of Ph.D. students, academics, and practitioners in various fields of study. It bridges the gap between theory and application, making it particularly relevant for those who wish to apply supervised Machine Learning techniques to concrete case studies in their respective disciplines. The inclusion of multiple software packages further enhances the book's value, as readers can gain exposure to different tools and methodologies and develop a versatile skill set.

In summary, this book offers a comprehensive introduction to supervised Machine Learning, catering to the needs of Ph.D. students, academics, and practitioners in diverse fields of study. Its inclusion of Python, R, and Stata software packages allows readers to choose the tool that aligns with their preferences and domain-specific requirements. By combining theoretical foundations with practical applications and addressing the specific needs of its target audience, this book provides a unique and valuable resource for anyone looking to master supervised Machine Learning for real-world scenarios.

Every chapter presents an initial theoretical part, where the basics of the methodologies are explained, followed by an applicative part, where the methods are applied to real-world datasets. Each chapter is self-contained, but the reader is invited to consider reading the two introductory chapters before going through the study of chapters dealing with single methods.

For ease of reproducibility, the Python, R, and Stata codes used in the book for carrying out the applications, along with the related datasets, are available at this GitHub link: https://github.com/GioCer73/Book_FSML_Ed1.

Organization of the Book

The book is organized into eight chapters for a structured and cohesive learning experience.

Chapter 1. The first chapter introduces the rationale and ontology of Machine Learning. Its purpose is to pave the way for the subsequent chapters and facilitate the understanding of the material covered.

Chapter 2. This chapter focuses on the statistics of Machine Learning, which serves as a crucial foundation for comprehending the following chapters. Understanding this section will ensure a smooth transition into the presentation and application of individual Machine Learning methods.

Chapter 3. Model selection and regularization are the main topics discussed in this chapter. Special emphasis is placed on lasso and elastic-net regression and classification, subset selection models, and the lasso's role in inferential analysis.

Chapter 4. The fourth chapter delves into discriminant analysis, nearest neighbor methods, and support vector machines. Although primarily utilized for classification purposes, it is worth noting that regression analysis can also be performed using the nearest neighbor and support vector machine methods.

Chapter 5. In this chapter, tree modeling is explored in both classification and regression scenarios. After outlining the tree-building algorithm, the chapter introduces three ensemble methods: bagging, random forests, and boosting machines, which are valuable for prediction and feature-importance detection.

Chapter 6. The sixth chapter provides an introduction to artificial neural networks. It covers fully connected neural networks, explaining their construction logic and demonstrating software applications for intelligent tasks such as image recognition.

Chapter 7. Building upon the foundation laid in the previous chapter, the seventh chapter delves into deep learning modeling. This section focuses on special artificial neural networks that leverage specific data ordering to enhance predictability and computation efficiency. Two deep learning architectures are presented and discussed: convolutional neural networks and recurrent neural networks.

Chapter 8. The concluding chapter serves as a primer on sentiment analysis. This approach involves developing a predictive mapping between human textual documents and the corresponding human polar sentiment associated with those documents. The chapter explores textual feature engineering and its application in predictive analytics.

By structuring the book into these eight chapters, readers can systematically progress from understanding the fundamentals to exploring advanced topics and applications in Machine Learning.

Rome, Italy Giovanni Cerulli

Acknowledgments

Writing a book is a challenging and arduous experience. In this regard, writing this book made no exception. I want to thank all my colleagues at IRCRES-CNR for their support, in particular, Emanuela Reale (our current director), Edoardo Lorenzetti (responsible for the research unit of Rome, where I work), Antonio Zinilli, and Lucio Morettini.

A special thank goes to Timberlake Ltd and Tstat Srl for the opportunity they gave me to run several courses on Machine Learning with Stata and Python.

I want to thank my lovely wife Rossella, and my sweet daughters Marta and Emma, for sustaining me during the time I was writing this book.

Another special thank goes to my parents, Felice and Barbara, my parents-in-law, Maria and Franco, and to my brother Riccardo.

Some fellow colleagues read this book. I want to thank them all.

All the errors are mine.

January 2023 Giovanni Cerulli

Contents

1 The Basics of Machine Learning 1
1.1 Introduction 1
1.2 Machine Learning: Definition, Rationale, Usefulness 2
1.3 From Symbolic AI to Statistical Learning 5
1.3.1 Numeral Recognition: Symbolic AI Versus Machine Learning 9
1.4 Non-identifiability of the Mapping: The *Curse of Dimensionality* 12
1.5 Conclusions 16
References 17

2 The Statistics of Machine Learning 19
2.1 Introduction 19
2.2 ML Modeling Trade-Offs 20
2.2.1 Prediction Versus Inference 20
2.2.2 Flexibility Versus Interpretability 21
2.2.3 Goodness-of-Fit Versus Overfitting 22
2.3 Regression Setting 23
2.4 Classification Setting 25
2.5 Training and Testing Error in Parametric and Nonparametric Regression: an Example 26
2.6 Measures for Assessing the Goodness-of-Fit 28
2.6.1 Dealing with Unbalanced Classes for Classification 34
2.7 Optimal Tuning of Hyper-Parameters 38
2.7.1 Information Criteria 39
2.7.2 Bootstrap 40
2.7.3 *K*-fold Cross-validation (CV) 42
2.7.4 Plug-in Approach 48
2.8 Learning Modes and Architecture 48

2.9 Limitations and Failures of Statistical Learning 51
2.9.1 ML Limitations 51
2.9.2 ML Failure 52
2.10 Software 56
2.11 Conclusions 57
References 57

3 Model Selection and Regularization 59
3.1 Introduction 59
3.2 Model Selection and Prediction 60
3.3 Prediction in High-Dimensional Settings 61
3.4 Regularized Linear Models 61
3.4.1 Ridge 62
3.4.2 Lasso 63
3.4.3 Elastic-Net 64
3.5 The Geometry of Regularized Regression 64
3.6 Comparing Ridge, Lasso, and Best Subset Solutions 66
3.7 Choosing the Optimal Tuning Parameters 67
3.7.1 Adaptive Lasso 67
3.7.2 Plugin Estimation 68
3.8 Optimal Subset Selection 69
3.8.1 Best (or Exhaustive) Subset Selection 70
3.8.2 Forward Stepwise Selection 71
3.8.3 Backward Stepwise Selection 73
3.9 Statistical Properties of Regularized Regression 74
3.9.1 Rate of Convergence 75
3.9.2 Support Recovery 76
3.9.3 Oracle Property 76
3.10 Causal Inference 77
3.10.1 Partialing-Out 78
3.10.2 Double-Selection 79
3.10.3 Cross-Fit Partialing-Out 79
3.10.4 Lasso with Endogenous Treatment 80
3.11 Regularized Nonlinear Models 82
3.12 Stata Implementation 83
3.12.1 The `lasso` Command 83
3.12.2 The `lassopack` Suite 85
3.12.3 The `subset` Command 86
3.12.4 Application S1: Linear Regularized Regression 87
3.12.5 Application S2: Nonlinear Lasso 93
3.12.6 Application S3: Multinomial Lasso 98
3.12.7 Application S4: Inferential Lasso Under Exogeneity 104

3.12.8 Application S5: Inferential Lasso Under Endogeneity 110
3.12.9 Application S6: Optimal Subset Selection 115
3.12.10 Application S7: Regularized Regression with Time-Series and Longitudinal Data 119
3.13 R Implementation 128
3.13.1 Application R1: Fitting a Gaussian Penalized Regression 128
3.13.2 Application R2: Fitting a Multinomial Ridge Classification 131
3.14 Python Implementation 134
3.14.1 Application P1: Fitting Lasso Using the `LassoCV()` Method 135
3.14.2 Application P2: Multinomial Regularized Classification in Python 138
3.15 Conclusion 143
References 144

4 Discriminant Analysis, Nearest Neighbor, and Support Vector Machine 147
4.1 Introduction 147
4.2 Classification 148
4.2.1 Linear and Logistic Classifiers 148
4.2.2 Discriminant Analysis 150
4.2.3 Nearest Neighbor 153
4.2.4 Support Vector Machine 157
4.3 Regression 169
4.3.1 SVM Regression 169
4.3.2 KNN Regression 171
4.4 Applications in Stata, R, and Python 172
4.4.1 Application S1: Linear Discriminant Analysis Classification in Stata 172
4.4.2 Application S2: Quadratic Discriminant Analysis Classification in Stata 177
4.4.3 Application S3: Nearest Neighbor Classification in Stata 179
4.4.4 Application S4: Tuning the Number of Nearest Neighbors in Stata 182
4.4.5 Application R1: K-Nearest Neighbor Classification in R 184
4.4.6 Application R2: Tuning the Number of Nearest Neighbors in R 185

4.4.7 Application P1: Support Vector Machine Classification in Python 191
4.4.8 Application P2: Support Vector Machine Regression in Python 195
4.5 Conclusions 199
References 200

5 Tree Modeling 201
5.1 Introduction 201
5.2 The Tree Approach 202
5.3 Fitting a Tree via Recursive Binary Splitting 203
5.4 Tree Overfitting 205
5.5 Tree Optimal Pruning 206
5.6 Classification Trees 208
5.7 Tree-Based Ensemble Methods 210
5.7.1 Bagging 211
5.7.2 Test-Error Estimation for Bagging 212
5.7.3 Measuring Feature Importance for Bagging 213
5.7.4 Random Forests 214
5.7.5 Boosting 215
5.7.6 Generalized Gradient Boosting 217
5.7.7 Logit Boosting 220
5.7.8 Multinomial Boosting 222
5.7.9 Tuning Boosting Hyper-Parameters 224
5.7.10 Feature Importance 225
5.8 R Implementation 225
5.8.1 Application R1: Fitting a Classification Tree Using `tree` 226
5.8.2 Application R2: Fitting Bagging and Random–Forests Using `randomForest` 231
5.8.3 Application R3: Fitting Multinomial Boosting Using `gbm` 234
5.8.4 Application R4: Tuning Random Forests and Boosting Using CARET 236
5.9 Stata Implementation 241
5.9.1 The `srtree` and `sctree` Commands 242
5.9.2 Application S1: Fitting a Regression Tree 243
5.9.3 Application S2: Fitting a Classification Tree 247
5.9.4 Application S3: Random Forests with `rforest` 250
5.10 Python Implementation 253
5.10.1 Application P1: Fitting and Plotting a Decision Tree in Python 254
5.10.2 Application P2: Tuning a Decision Tree in Python 255

5.10.3 Application P3: Random Forests with Python 259
5.10.4 Application P4: Boosting in Python 261
5.11 Conclusions 266
References 266

6 Artificial Neural Networks 269
6.1 Introduction 269
6.2 The ANN Architecture 270
6.2.1 ANN Imaging Recognition: An Illustrative Example 273
6.3 Fitting an ANN 276
6.3.1 The Gradient Descent Approach 277
6.3.2 The Back-Propagation Algorithm 278
6.3.3 Specific Issues Arising when Fitting an ANN 281
6.4 Two Notable ANN Architectures 283
6.4.1 The Perceptron 283
6.4.2 The Adaline 286
6.5 Python Implementation 289
6.5.1 Application P1: Implementing the Perceptron in Python 289
6.5.2 Application P2: Fitting ANNs with Scikit-Learn 292
6.5.3 Application P3: Fitting ANNs with Keras 296
6.6 R Implementation 299
6.6.1 Application R1: Fitting an ANN in R Using the mlp() Function 299
6.6.2 Application R2: Fitting and Cross-Validating an ANN in R Using the neuralnet() Function 303
6.6.3 Application R3: Tuning ANN Parameters Using CARET 307
6.7 Stata Implementation 312
6.7.1 Application S1: Fitting an ANN in Stata Using the mlp2 Command 312
6.7.2 Application S2: Comparing an ANN and a Logistic Model 315
6.7.3 Application S3: Recognizing Handwritten Numerals Using mlp2 317
6.8 Conclusion 321
References 321

7 Deep Learning 323
7.1 Introduction 323
7.2 Deep Learning and Data Ordering 324
7.3 Convolutional Neural Networks 327
7.3.1 The CNN Architecture 328
7.3.2 The Convolutional Operation 328
7.3.3 Pooling 333

7.3.4 Full Connectivity 334
7.3.5 Multi-channel/Multi-filter CNNs 334
7.3.6 Tuning a CNN by Dropout 335
7.3.7 Application P1: Fitting a CNN Using Keras—The Lenet-5 Architecture 336
7.4 Recurrent Neural Networks 342
7.4.1 Back-Propagation for an RNN 344
7.4.2 Long Short-Term Memory Networks 345
7.4.3 Application P2: Univariate Forecasting Using an LSTM Network 348
7.4.4 Application P3: Multivariate Forecasting Using an LSTM Network 353
7.4.5 Application P4: Text Generation with an LSTM Network 358
7.5 Conclusion 364
References 364

8 Sentiment Analysis 365
8.1 Introduction 365
8.2 Sentiment Analysis: Definition and Logic 366
8.2.1 Step 1. Textual Feature Engineering 367
8.2.2 Step 2. Statistical Learning 370
8.2.3 Step 3. Quality Assessment 371
8.3 Applications 371
8.3.1 Application S1: Classifying the Topic of Newswires Based on Their Text 371
8.3.2 Application R1: Sentiment Analysis of IMDB Movie Reviews 374
8.3.3 Application P1: Building a “Spam” Detector in Python 381
8.4 Conclusions 383
References 383

Author Index 385

Index 389

List of Figures

Fig. 1.1 Machine Learning (ML) standard taxonomy 3
Fig. 1.2 Deep learning (DL) taxonomy 4
Fig. 1.3 Process carrying out an intelligent task 5
Fig. 1.4 Timeline development of artificial intelligence paradigms and methods 6
Fig. 1.5 Marta and Emma Cerulli when they were six years old. Examples of intelligent systems able to learn from experience. Photographs courtesy of Giovanni Cerulli 7
Fig. 1.6 Examples of two numerals to be recognized by a machine 9
Fig. 1.7 Examples of two numerals to be recognized by a machine exploiting one angular measurement 9
Fig. 1.8 Examples of two numerals to be recognized by a machine exploiting two angular measurements. Numeral seven is written with a steeper vertical bar 10
Fig. 1.9 Prediction of two numerals to be recognized by a machine exploiting two angular measurements 10
Fig. 1.10 Some handwritten numerals 11
Fig. 1.11 Data representation of the two handwritten numerals 11
Fig. 1.12 Data representation of the two handwritten numerals 12
Fig. 1.13 Face completion. Predicting the lower half of a face by knowing the upper half. Olivetti faces dataset. The dataset contains a set of freely available face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge 13
Fig. 1.14 Identification of the conditional mean with no sparse data 13
Fig. 1.15 Non identification of the conditional mean with sparse data 14
Fig. 1.16 Taxonomy of parametric and nonparametric statistical learning methods 15
Fig. 1.17 How Machine Learning and big data impact sampling and specification errors 16

Fig. 2.1 Trade-off between bias and variance as functions of model complexity . . . 24
Fig. 2.2 Visualizing training and testing errors in a parametric regression (least squares) . . . 27
Fig. 2.3 Visualizing training and test error in a nonparametric regression (nearest neighbor) . . . 28
Fig. 2.4 Example of a confusion matrix for a three-class classification, where classes are labeled as A, B, and C . . . 30
Fig. 2.5 Example of a two-class confusion matrix for the Covid-19 viral infection. TP = number of true positives; FP = number of false positives; FN = number of false negatives; TN = number of true negatives; $N = TP + FP + FN + TN$ . . . 30
Fig. 2.6 Example of imbalance data. Precision and recall are better goodness-of-fit measures than the accuracy when positives are much lower than negative cases in the initial dataset . . . 32
Fig. 2.7 Example of a receiver operator characteristics (ROC) curve . . . 33
Fig. 2.8 Example of non-classification paradox . . . 34
Fig. 2.9 Oversampling procedure . . . 36
Fig. 2.10 Under-sampling procedure . . . 36
Fig. 2.11 Graphical representation of the ROSE (Random oversampling examples) procedure . . . 37
Fig. 2.12 Graphical representation of the SMOTE (Synthetic minority oversampling technique) procedure . . . 38
Fig. 2.13 Flow-diagram of the bootstrap procedure for computing the empirical distribution of a statistic for the population parameter a . . . 41
Fig. 2.14 Visualizing bootstrap overlapping in a sample with $N = 10$ observations and $B = 2$ bootstrap replications . . . 42
Fig. 2.15 Visualizing the K-fold cross-validation with $K = 5$ . . . 43
Fig. 2.16 Visualizing the cross-validation for time-series data. Varying window setting . . . 45
Fig. 2.17 Visualizing the cross-validation for time-series data. Fixed window setting . . . 46
Fig. 2.18 Example of cross-validation for time-series data, where an auto-regressive structure of order 2–AR(2)–is considered. Varying window setting . . . 47
Fig. 2.19 Cross-validation optimal hyper-parameter's tuning scheme for a generic ML method . . . 47
Fig. 2.20 The meta-learning machine architecture. This figure has been drawn from Cerulli (2021) . . . 50
Fig. 2.21 Example of how data sparseness can weaken prediction ability . . . 54
Fig. 2.22 Example of how data ordering can help prediction. Sequential and spatial data . . . 55

Fig. 3.1 Comparison of Lasso, Ridge, and Elastic-net penalization solutions 65
Fig. 3.2 Lasso, Ridge, and Subset selection solutions in the case of orthonormal features 66
Fig. 3.3 Adaptive Lasso solution 68
Fig. 3.4 Lasso feature selection in a simple structural model. Left-hand panel: given its low indirect effect on y (due to set $b = 1$), x_1 is correctly not selected. Right-hand panel: given its higher indirect effect on y (due to set $b = 2$), x_1 is erroneously selected. All errors and variables are standard normal. Errors are mutually uncorrelated, and uncorrelated with all the observed variables 77
Fig. 3.5 Lasso cross-validation plot. Linear Lasso of the median value of the houses as a function of various housing characteristics 89
Fig. 3.6 Lasso coefficients path-diagram. Linear adaptive Lasso of the median value of the houses as a function of various housing characteristics 93
Fig. 3.7 Lasso Logit cross-validation plot. Logit Lasso of the probability to survive the Titanic disaster as a function of various passengers' characteristics 96
Fig. 3.8 Lasso Logit coefficients path-diagram. Logit Lasso of the probability to survive the Titanic disaster as a function of various passengers' characteristics 96
Fig. 3.9 The cancer/genes coefficients heatmap. OR: odds ratios 103
Fig. 3.10 Lasso treatment effect of union membership on the log-salary by occupation type. Occupation baseline category: "managers and administratives" 107
Fig. 3.11 Optimal number of features using exhaustive (or best) subset selection according to different information criteria. The residual sum of squares (RSS) graph shows the training error behavior 118
Fig. 3.12 Rolling 1-step ahead cross-validation 121
Fig. 3.13 Minimum MSPE at different forecasting horizons 123
Fig. 3.14 Out-of-sample Lasso forecasting 124
Fig. 3.15 1-step ahead cross-validation plot in a panel data setting 127
Fig. 3.16 Lasso coefficients' path using `glmnet()` 131
Fig. 3.17 Lasso tenfold cross-validation results using `glmnet()` 132
Fig. 3.18 Lasso coefficients' plot as function of log of lambda. Dataset: `diabetes`. Target variable: `diabprog` 137
Fig. 3.19 Plot of actual and predicted values. Dataset: `diabetes`; Target variable: `diabprog` 138
Fig. 3.20 Plot of training and test error as a function of the grid index. Dataset: `diabetes`. Target variable: `diabprog` 142
Fig. 4.1 Example of a linear and logistic probability fit 149

Fig. 4.2 Example of linear and logistic decision boundaries 150
Fig. 4.3 Discriminant analysis classification logic. We classify the new observation with $x = x_0$ as *red*, as $f_{red}(x_0) > f_{green}(x_0)$. Observe that, in this case, we assume same priors, that is, $q_{red} = q_{green}$ 151
Fig. 4.4 Discriminant analysis classification logic. By increasing the prior of class *green*, we change the decision boundary by increasing the likelihood to classify a new observation as *green* 152
Fig. 4.5 Nearest-neighbor classifier with neighborhood of size $M = 3$. The decision boundary is built based on the class obtaining—within the neighborhood of the point at stake—the largest frequency. This approach uses a pure data-driven approach to classify units, based on the *closeness* principle 154
Fig. 4.6 Simulation of decision boundary adaptation: comparison between a linear discriminant classifier (LDA) and a Nearest-neighbor classifier with neighborhood of size $M = 3$ (3-NN). The true decision boundary is a quadratic function 155
Fig. 4.7 M-nearest neighbor decision boundary at different number of nearest neighbors. The largest model complexity is reached when $M = 1$, the smallest when $M = 50$. The parameter M is a tuning parameter, entailing a trade-off between prediction bias and prediction variance. As such, M can be optimally tuned, for example, by cross-validation 156
Fig. 4.8 Example of a 5-Nearest neighbor algorithm with $N = 24$ training observations, and two classes—*blue*. For a new instance $i =$ x, we impute the class using the Bayes conditional probability formula (4.17) 157
Fig. 4.9 Example of a separating hyperplane with two classes 158
Fig. 4.10 Non-uniqueness of a separating hyperplane 159
Fig. 4.11 Computation of the *margin* 159
Fig. 4.12 Computation of the *maximal margin hyperplane* 160
Fig. 4.13 Maximal margin hyperplane with perfectly separable classes. No observation is allowed to lie between the hyperplane and the marginal hyperplane 161
Fig. 4.14 Location of observations within a support vector classification, based on system (4.19) 162
Fig. 4.15 Support vector classifier for non-separable classes. Observations are allowed to lie both in the wrong side of the margin or of the hyperplane. These observations—known as support vectors—are those responsible for classification 163

Fig. 4.16 Support vector classifier for non-separable classes at different levels of the cost parameter C (or τ). Larger values of C (i.e., smaller values of τ) produce tighter margins. Variations of C induce a trade-off between prediction bias and prediction variance 164
Fig. 4.17 Support vector classifier for non-separable classes where a linear boundary is unable to classify well 165
Fig. 4.18 Radial kernel. The kernel weight decreases exponentially with an increasing distance between point i and point i' 167
Fig. 4.19 Radial Basis Function (RBF) kernel as function of the γ parameter 168
Fig. 4.20 Numerical example of computation of the Support Vector Machine (SVM) classifier with 3 training observations and 2 features 168
Fig. 4.21 OLS and SVM regression loss functions: a comparison 170
Fig. 4.22 Example of a SVM regression fit 171
Fig. 4.23 Illustrative example of the k-nearest neighbor regression algorithm with only one feature, and $k = 3$ 172
Fig. 4.24 Scatterlplot of the three groups defined within the feature space 173
Fig. 4.25 Comparison of two classifiers, C-1 and C-2, performing respectively better and worse than the "No Information Rate" (NIR) classifier 190
Fig. 4.26 Graphical representation of the SVM confusion matrix 195
Fig. 4.27 Support vector machine regression. Actual versus fitted outcomes on the test dataset. Dataset: `Boston`. Target variables: Median value of homes in 1000s dollars 199
Fig. 5.1 An illustrative tree plot based on binary splitting 202
Fig. 5.2 Regression tree plot predicting the median value of owner-occupied housing units (variable `medv`), as a function of the average number of rooms per dwelling (variable `rm`), and the percentage of population that is lower status (variable `lstat`). Dataset: `boston_house` 203
Fig. 5.3 Growing a tree by recursive binary splitting and best–first approach 204
Fig. 5.4 Sequence of trees indexed by the penalization parameter α 207
Fig. 5.5 Plot of the classification error rate, Gini, and cross-entropy metrics in a two-class setting as a function of the proportion of class 1 210
Fig. 5.6 Plot of a pruned tree of size $M = 5$ over the training sample. Dataset: `Eter`. Target variable: `pub_cat` (categorical variable for publications) 229
Fig. 5.7 Cross-validated optimal pruned tree. Dataset: `Eter`. Target variable: `pub_cat` (categorical variable for publications) 231

Fig. 5.8 Random forest feature importance indexes. Dataset: `Eter`. Target variable: `pub_cat` (categorical variable for publications) 233

Fig. 5.9 Multinomial boosting feature importance indexes. Dataset: `Eter`. Target variable: `pub_cat` (categorical variable for publications) 236

Fig. 5.10 Boosting cross-validation results over the parameters `interaction.depth` and `n.trees`. Dataset: `Eter`. Target variable: `pub_cat` (categorical variable for publications) 241

Fig. 5.11 Plot of the unpruned regression tree over the training sample. Dataset: `boston_house`. Target variable: `medv` (median value of owner-occupied homes in $1000s) 245

Fig. 5.12 Plot of the 4-leave pruned regression tree over the training sample. Dataset: `boston_house`. Target variable: `medv` (median value of owner-occupied homes in $1000s) 246

Fig. 5.13 Plot of cross–validation results for tree optimal pruning the training sample. The minimized objective function is the deviance. Dataset: `carseats`. Target variable: `High` (`Yes` if Sales > 8; `No` otherwise) 249

Fig. 5.14 Plot of the optimal classification tree over the training sample. Dataset: `carseats`. Target variable: `High` (`Yes` if Sales > 8; `No` otherwise) 250

Fig. 5.15 Plot of a classification tree in Python. Dataset: `iris`. Features: length and width of sepals and petals. Target variable: three species of iris (setosa, virginica, and versicolor) 255

Fig. 5.16 Plot of random forests feature importance measures. Dataset: `boston` 260

Fig. 6.1 Simplified representation of a human brain's neuron 270

Fig. 6.2 Activation functions. Plot of the *sigmoid* $\sigma(t) = 1/(1+e^{s \cdot t})$ and *rectified linear unit* (*ReLU*) $max[0, s \cdot t]$ functions. at different scale value s, controlling for the so-called *activation rate*. Observe that a larger value of s entails harder activation 272

Fig. 6.3 A feedforward artificial neural network architecture characterized by: (i) two inputs, (ii) one hidden layer, (iii) four hidden units (or neurons), and (iv) two outcomes 273

Fig. 6.4 Images A and B and their data representation 274

Fig. 6.5 Imaging recognition process carried out by a feedforward ANN 274

Fig. 6.6 Representation of the gradient descent algorithm 278

Fig. 6.7 Workflow of the back–propagation algorithm based on gradient descent to fit an ANN 280

Fig. 6.8 The Perceptron classification rule 284

Fig. 6.9 Weight's updating rule for the Perceptron as illustrated in Table 6.1. *Note* we consider only one feature x. For the sake of simplicity, we consider all instances with $x = 1$, so that $\theta x = \theta$. The initial weight is equal to 8 286

Fig. 6.10 The Perceptron learning architecture. *Note* the learning process via gradient descent takes place when the error is different from zero 287

Fig. 6.11 The Adeline learning architecture. *Note* the learning process via gradient descent takes place using a linear prediction. The final continuous predicted outcome is then transformed into an integer label 288

Fig. 6.12 Perceptron updates as a function of the epochs 292

Fig. 6.13 Plot of the Iris dataset used by fitting the Perceptron 292

Fig. 6.14 Plot of a 2-layer ANN using the R function `plotnet()` on the Iris dataset 302

Fig. 6.15 Graph of feature-importance measures for the output "Irsis-setosa" using the Iris dataset. Method: Olden et al. (2004) 302

Fig. 6.16 Plot of a 2-layer ANN using the `plot()` function with respectively 5 and 3 neurons. Dataset: `Boston`; target variable: `medv`: the median value of a house. Features 13 house characteristics 306

Fig. 6.17 Plot of actual versus predicted target values in the ANN and LM. Dataset `Boston`; target variable: `medv`: the median value of a house. Features: 13 house characteristics 306

Fig. 6.18 Graphical representation of a numeral 320

Fig. 6.19 Graphical representation of a series of numerals 320

Fig. 7.1 A stylized versus a real cat. Stylized cat drawing: courtesy of Giovanni Cerulli 324

Fig. 7.2 Example of negative, positive, and zero spatial autocorrelation 326

Fig. 7.3 Example of a feature map projection in a convolutional neural network 328

Fig. 7.4 Examples of a padding of size 3 (for a vector x), and size 1 (for a matrix M) 329

Fig. 7.5 Example of a stride set either to 1 and to 2 using a 2×2 filter along the matrix M 329

Fig. 7.6 1D sliding window approach with size $m = 3$ and stride $s = 2$ to build the matrix $\mathbf{X}$ 331

Fig. 7.7 2D sliding window approach with size $m = 3$ and stride $s = 2$ to build the matrix $\mathbf{X}$ 331

Fig. 7.8 A comparison of full, same, and valid pudding in convolutional operations 332

Fig. 7.9 Example of max-pooling using a $\mathbf{P}_{2\times2}$ pooling matrix 333

Fig. 7.10 Example of a multi-channel/multi-filter CNN for colored images recognition . . . 335
Fig. 7.11 The Lenet-5 convolutional neural network . . . 337
Fig. 7.12 Graphical representation of the digits in the MNIST dataset . . . 341
Fig. 7.13 Patterns of the training and testing *loss* and *accuracy* as function and the epochs for the Lenet-5 convolutional neural network estimated using Keras . . . 342
Fig. 7.14 Feedforward versus recurrent neural network architectures . . . 343
Fig. 7.15 The *unfolded* structure of a recurrent neural network (RNN) . . . 343
Fig. 7.16 Back-propagation within a recurrent neural network (RNN) . . . 344
Fig. 7.17 Information flow within a Long Short-Term Memory network (LSTM) . . . 346
Fig. 7.18 Structure of a *Tensor* used as input in a Long Short-Term Memory (LSTM) networks. A tensor is a 3D array with three axes: *observations*, *time-steps*, *features* . . . 349
Fig. 7.19 Long Short-Term Memory networks (LSTM) univariate in-sample and out-of-sample forecasting. Dataset: `hardware`. Variable: `dim`: *dimensional lumber sales.* Number of lags: 3 . . . 353
Fig. 7.20 Patterns of the test and train loss function (mean absolute error, MAE) in an LSTM network fit as a function of the epochs . . . 357
Fig. 7.21 Patterns of the actual and forecast (both in-sample and out-of-sample) time-series values for a multivariate LSTM network fit using as loss function the mean absolute error (MAE). Dataset: `lutkepohl2`. Target variable: `cons` (consumption) . . . 358
Fig. 8.1 Sentiment analysis of the IMDB movie reviews: plot of the ROC curve for the GBM (Gradient Boosting Machine), RF (Random Forests), and NN (Neural Network) . . . 380

List of Tables

Table 1.1 A one-to-one word translation of the statistical and the Machine Learning jargon 8

Table 2.1 Flexibility measures for a parametric model like the ordinary least squares (OLS), and for a nonparametric model like the K-nearest neighbor (KNN) 22

Table 2.2 Main Machine Learning methods and associated tuning hyper-parameters. GLM: Generalized Linear Models 49

Table 3.1 Description for the Exhaustive subset selection algorithm 70

Table 3.2 Description for the Forward stepwise selection algorithm 72

Table 3.3 Comparison of the computational burden of optimal subset selection algorithms 72

Table 3.4 Example of a Forward stepwise selection procedure with $p = 4$ features 72

Table 3.5 Description for the Backward stepwise selection procedure 73

Table 3.6 Example of a Backward stepwise selection procedure with $p = 4$ features 74

Table 3.7 Description of the partialing-out procedure 79

Table 3.8 Description of the double-selection procedure 79

Table 3.9 Description of the Cross-fit partialing-out procedure 80

Table 3.10 Lasso IV procedure with z high-dimensional and x low-dimensional 81

Table 3.11 Lasso IV procedure with both z and x high-dimensional 81

Table 3.12 Lasso and related estimators available in Stata 17 84

Table 3.13 Description of the `boston_house` dataset 87

Table 3.14 Description of the `titanic` dataset. Number of passengers 891; number of variables 11 (excluding passengers' id) 94

Table 3.15 Dataset `nlsw88`: National Longitudinal Survey of Mature and Young Women. Year 1988 105

Table 3.16 Description of the Hitters dataset. Number of observations: 322. Number of features: 21 115
Table 3.17 Rolling h-step ahead cross-validation 120
Table 3.18 Description of the `diabetes` dataset. Number of variables: 11. Sample size: 442 135
Table 5.1 Algorithm for optimal tree pruning via K–fold cross-validation 207
Table 5.2 Example of a tree cross-validation MSEs calculation with a 5-fold split and a grid of three values of α 208
Table 5.3 Algorithm for the computation of the bagging ensemble prediction 212
Table 5.4 Algorithm for the computation of the out-of-bag (OOB) test error for regression and classification bagging 213
Table 5.5 Random forest implementation algorithm 215
Table 5.6 Implementation of the boosting–tree algorithm for regression 216
Table 5.7 Forward stagewise implementation algorithm 218
Table 5.8 Gradient tree boosting algorithm for a generic loss function $L(\cdot)$ 220
Table 5.9 Implementation of the Logit gradient tree boosting algorithm 223
Table 5.10 Implementation of the multinomial gradient tree boosting algorithm 223
Table 5.11 Description of the Eter dataset 226
Table 5.12 Hyper-parameters tuning with cross-validation using of the CARET package. Note: one can refer to the vignette to see the other arguments of the function 237
Table 5.13 Description of `carseat` dataset containing sales of child car seats at 400 different stores 247
Table 6.1 A numerical example of the weights' updating rule for the Perceptron, when we consider only one feature $x = 1$ 285
Table 6.2 The Perceptron algorithm 287
Table 6.3 The Adaline learning algorithm 289
Table 6.4 Example of a Keras modeling workflow 297
Table 7.1 Example of a univariate time-series dataset with three lags . . . 349
Table 7.2 Example of a multivariate time-series dataset with two features and two lags 354
Table 7.3 Text sequence data representation. This dataset is obtained by generating sequences of *length* equal to 6 and *pace* equal to 1 starting from the initial text: "Gio does good" 359
Table 7.4 Text sequence data representation. The input data X of Table 7.3 can be represented as a 2D array with shape [*observations*, *sequence-length*] 359

Table 8.1 Example of how to generate features from texts using a uni-gram algorithm 368
Table 8.2 Example of how to generate features form texts using a bi-gram algorithm. Single-token variables are not showed for simplicity 369