Hands-On Machine Learning with R, 2ed

Brad Boehmke & Brandon Greenwell

2023-07-28

Table of contents

Welcome

Welcome to the second edition of *Hands-On Machine Learning with R*. This book provides hands-on modules for many of the most common machine learning methods to include:

- Generalized low rank models
- Clustering algorithms
- Autoencoders
- Regularized models
- Random forests
- Gradient boosting machines
- Deep neural networks
- Stacking / super learners
- and more!

You will learn how to build and tune these various models with R packages that have been tested and approved due to their ability to scale well. However, our motivation in almost every case is to describe the techniques in a way that helps develop intuition for its strengths and weaknesses. For the most part, we minimize mathematical complexity when possible but also provide resources to get deeper into the details if desired.

> **i** Note
>
> This book is undergoing heavy restructuring and may be confusing or incomplete; however, we hope for the first draft to be completed by the fall of 2023. You can find the complete first edition at bradleyboehmke.github.io/HOML.

Who should read this

We intend this work to be a practitioner's guide to the machine learning process and a place where one can come to learn about the approach and to gain intuition about the many commonly used, modern, and powerful methods accepted in the machine learning community. If you are familiar with the analytic methodologies, this book may still serve as a reference for how to work with the various R packages for implementation.

This book is not meant to be an introduction to R or to programming in general; as we assume the reader has familiarity with the R language to include defining functions, managing R objects, controlling the flow of a program, and other basic tasks. If not, we would refer you to R for Data Science (Wickham and Grolemund 2016) to learn the fundamentals of data science with R such as importing, cleaning, transforming, visualizing, and exploring your data. For those looking to advance their R programming skills and knowledge of the language, we would refer you to Advanced R (Wickham 2014). Nor is this book designed to be a deep dive into the theory and math underpinning machine learning algorithms. Several books already exist that do great justice in this arena (i.e. Elements of Statistical Learning (Hastie, Tibshirani, and Friedman 2009), Computer Age Statistical Inference (Efron and Hastie 2016), Deep Learning (Goodfellow, Bengio, and Courville 2016)).

Instead, this book is meant to help R users learn to use the machine learning stack within R, which includes using various R packages such as the **tidymodels** ecosystem of packages for model development, **vip** and **pdp** for model interpretation, TODO (add others as we develop) and others to effectively model and gain insight from your data. The book favors a hands-on approach, growing an intuitive understanding of machine learning through concrete examples and just a little bit of theory. While you can read this book without opening R, we highly recommend you experiment with the code examples provided throughout.

Why R

In this book we focus on implementing machine learning tasks with R. R has emerged over the last couple decades as a first-class tool for scientific computing tasks, and has been a consistent leader in implementing statistical methodologies for analyzing data. The usefulness of R for data science stems from the large, active, and growing ecosystem of third-party packages. We are not ignoring other languages such as Python or Julia because we think these tools are inferior. They're not! And in practice, most organizations and data science teams use a mix of languages. In fact, throughout this book we may reference methods or implementations in other languages and we may even provide a few examples in Python. However, we strongly believe that it's best to master one tool at a time, and R is a great place to start.

Conventions used in this book

The following typographical conventions are used in this book:

- *strong italic*: indicates new terms,
- **bold**: indicates package & file names,
- `inline code`: monospaced highlighted text indicates functions or other commands that could be typed literally by the user,
- code chunk: indicates commands or other text that could be typed literally by the user

```
1 + 2
## [1] 3
```

In addition to the general text used throughout, you will notice the following code chunks with images:

> 💡 Tip
>
> Signifies a tip or suggestion

> ℹ️ Note
>
> Signifies a general note

> ⚠️ Warning
>
> Signifies a warning or caution

Additional resources

There are many great resources available to learn about machine learning. Throughout the chapters we try to include many of the resources that we have found extremely useful for digging deeper into the methodology and applying with code. However, due to print restrictions, the hard copy version of this book limits the concepts and methods discussed. Online supplementary material exists at https://koalaverse.github.io/homlr/. The additional material will accumulate over time and include extended chapter material (i.e., random forest package benchmarking) along with brand new content we couldn't fit in (i.e., random hyperparameter search). In addition, you can download the data used throughout the book, find teaching resources (i.e., slides and exercises), and more.

Acknowledgments

We'd like to thank everyone who contributed feedback, typo corrections, and discussions while the book was being written. GitHub contributors included @agailloty, @asimumba, @benprew, @bfgray3, @bragks, @cunningjames, @DesmondChoy, @erickeniuk, @j-ryanhart, @lcreteig, @liangwu82, @Lianta, @mccurcio, @mmelcher76, @MMonterosso89, @nsharkey, @raycblai, @schoonees, @tpristavec and @william3031. We'd also like to thank folks such as Alex Gutman, Greg Anderson, Jay Cunningham, Joe Keller, Mike Pane, Scott Crawford, and several other co-workers who provided great input around much of this machine learning content.

Software information

This book was built with the following packages and R version. All code was executed on 2019 MacBook Pro with a 2.6 GHz 6-Core Intel Core i7 processor, 16 GB of memory, 2667 MHz speed, and double data rate synchronous dynamic random access memory (DDR4).

```r
# packages used
pkgs <- c(
    "tidymodels",
    "vip"
)

# package & session info
sessioninfo::session_info(pkgs)
#> - Session info -----------------------------------------------------------------
#>  setting  value
#>  version  R version 4.2.0 (2022-04-22)
#>  os       Ubuntu 22.04.2 LTS
#>  system   x86_64, linux-gnu
#>  ui       X11
#>  language (EN)
#>  collate  C.UTF-8
#>  ctype    C.UTF-8
#>  tz       UTC
#>  date     2023-07-28
#>  pandoc   2.9.2.1 @ /usr/bin/ (via rmarkdown)
#>
#> - Packages ---------------------------------------------------------------------
#>  package      * version    date (UTC) lib source
#>  backports      1.4.1      2021-12-13 [1] CRAN (R 4.2.0)
#>  broom          1.0.5      2023-06-09 [1] CRAN (R 4.2.0)
#>  cachem         1.0.8      2023-05-01 [1] CRAN (R 4.2.0)
#>  class          7.3-20     2022-01-16 [3] CRAN (R 4.2.0)
#>  cli            3.6.1      2023-03-23 [1] CRAN (R 4.2.0)
#>  clock          0.7.0      2023-05-15 [1] CRAN (R 4.2.0)
#>  codetools      0.2-18     2020-11-04 [3] CRAN (R 4.2.0)
#>  colorspace     2.1-0      2023-01-23 [1] CRAN (R 4.2.0)
#>  conflicted     1.2.0      2023-02-01 [1] CRAN (R 4.2.0)
#>  cpp11          0.4.5      2023-07-20 [1] CRAN (R 4.2.0)
#>  data.table     1.14.8     2023-02-17 [1] CRAN (R 4.2.0)
#>  diagram        1.6.5      2020-09-30 [1] CRAN (R 4.2.0)
#>  dials          1.2.0      2023-04-03 [1] CRAN (R 4.2.0)
```

```
#>   DiceDesign    1.9         2021-02-13 [1] CRAN (R 4.2.0)
#>   digest        0.6.33      2023-07-07 [1] CRAN (R 4.2.0)
#>   dplyr         1.1.2       2023-04-20 [1] CRAN (R 4.2.0)
#>   ellipsis      0.3.2       2021-04-29 [1] CRAN (R 4.2.0)
#>   fansi         1.0.4       2023-01-22 [1] CRAN (R 4.2.0)
#>   farver        2.1.1       2022-07-06 [1] CRAN (R 4.2.0)
#>   fastmap       1.1.1       2023-02-24 [1] CRAN (R 4.2.0)
#>   foreach       1.5.2       2022-02-02 [1] CRAN (R 4.2.0)
#>   furrr         0.3.1       2022-08-15 [1] CRAN (R 4.2.0)
#>   future        1.33.0      2023-07-01 [1] CRAN (R 4.2.0)
#>   future.apply  1.11.0      2023-05-21 [1] CRAN (R 4.2.0)
#>   generics      0.1.3       2022-07-05 [1] CRAN (R 4.2.0)
#>   ggplot2       3.4.2       2023-04-03 [1] CRAN (R 4.2.0)
#>   globals       0.16.2      2022-11-21 [1] CRAN (R 4.2.0)
#>   glue          1.6.2       2022-02-24 [1] CRAN (R 4.2.0)
#>   gower         1.0.1       2022-12-22 [1] CRAN (R 4.2.0)
#>   GPfit         1.0-8       2019-02-08 [1] CRAN (R 4.2.0)
#>   gtable        0.3.3       2023-03-21 [1] CRAN (R 4.2.0)
#>   hardhat       1.3.0       2023-03-30 [1] CRAN (R 4.2.0)
#>   infer         1.0.4       2022-12-02 [1] CRAN (R 4.2.0)
#>   ipred         0.9-14      2023-03-09 [1] CRAN (R 4.2.0)
#>   isoband       0.2.7       2022-12-20 [1] CRAN (R 4.2.0)
#>   iterators     1.0.14      2022-02-05 [1] CRAN (R 4.2.0)
#>   KernSmooth    2.23-20     2021-05-03 [3] CRAN (R 4.2.0)
#>   labeling      0.4.2       2020-10-20 [1] CRAN (R 4.2.0)
#>   lattice       0.20-45     2021-09-22 [3] CRAN (R 4.2.0)
#>   lava          1.7.2.1     2023-02-27 [1] CRAN (R 4.2.0)
#>   lhs           1.1.6       2022-12-17 [1] CRAN (R 4.2.0)
#>   lifecycle     1.0.3       2022-10-07 [1] CRAN (R 4.2.0)
#>   listenv       0.9.0       2022-12-16 [1] CRAN (R 4.2.0)
#>   lubridate     1.9.2       2023-02-10 [1] CRAN (R 4.2.0)
#>   magrittr      2.0.3       2022-03-30 [1] CRAN (R 4.2.0)
#>   MASS          7.3-56      2022-03-23 [3] CRAN (R 4.2.0)
#>   Matrix        1.4-1       2022-03-23 [3] CRAN (R 4.2.0)
#>   memoise       2.0.1       2021-11-26 [1] CRAN (R 4.2.0)
#>   mgcv          1.8-40      2022-03-29 [3] CRAN (R 4.2.0)
#>   modeldata     1.1.0       2023-01-25 [1] CRAN (R 4.2.0)
#>   modelenv      0.1.1       2023-03-08 [1] CRAN (R 4.2.0)
#>   munsell       0.5.0       2018-06-12 [1] CRAN (R 4.2.0)
#>   nlme          3.1-157     2022-03-25 [3] CRAN (R 4.2.0)
#>   nnet          7.3-17      2022-01-16 [3] CRAN (R 4.2.0)
```

```
#> numDeriv     2016.8-1.1 2019-06-06 [1] CRAN (R 4.2.0)
#> parallelly   1.36.0     2023-05-26 [1] CRAN (R 4.2.0)
#> parsnip      1.1.0      2023-04-12 [1] CRAN (R 4.2.0)
#> patchwork    1.1.2      2022-08-19 [1] CRAN (R 4.2.0)
#> pillar       1.9.0      2023-03-22 [1] CRAN (R 4.2.0)
#> pkgconfig    2.0.3      2019-09-22 [1] CRAN (R 4.2.0)
#> prettyunits  1.1.1      2020-01-24 [1] CRAN (R 4.2.0)
#> prodlim      2023.03.31 2023-04-02 [1] CRAN (R 4.2.0)
#> progressr    0.13.0     2023-01-10 [1] CRAN (R 4.2.0)
#> purrr        1.0.1      2023-01-10 [1] CRAN (R 4.2.0)
#> R6           2.5.1      2021-08-19 [1] CRAN (R 4.2.0)
#> RColorBrewer 1.1-3      2022-04-03 [1] CRAN (R 4.2.0)
#> Rcpp         1.0.11     2023-07-06 [1] CRAN (R 4.2.0)
#> recipes      1.0.6      2023-04-25 [1] CRAN (R 4.2.0)
#> rlang        1.1.1      2023-04-28 [1] CRAN (R 4.2.0)
#> rpart        4.1.16     2022-01-24 [3] CRAN (R 4.2.0)
#> rsample      1.1.1      2022-12-07 [1] CRAN (R 4.2.0)
#> rstudioapi   0.15.0     2023-07-07 [1] CRAN (R 4.2.0)
#> scales       1.2.1      2022-08-20 [1] CRAN (R 4.2.0)
#> shape        1.4.6      2021-05-19 [1] CRAN (R 4.2.0)
#> slider       0.3.0      2022-11-16 [1] CRAN (R 4.2.0)
#> SQUAREM      2021.1     2021-01-13 [1] CRAN (R 4.2.0)
#> stringi      1.7.12     2023-01-11 [1] CRAN (R 4.2.0)
#> stringr      1.5.0      2022-12-02 [1] CRAN (R 4.2.0)
#> survival     3.3-1      2022-03-03 [3] CRAN (R 4.2.0)
#> tibble       3.2.1      2023-03-20 [1] CRAN (R 4.2.0)
#> tidymodels   1.1.0      2023-05-01 [1] CRAN (R 4.2.0)
#> tidyr        1.3.0      2023-01-24 [1] CRAN (R 4.2.0)
#> tidyselect   1.2.0      2022-10-10 [1] CRAN (R 4.2.0)
#> timechange   0.2.0      2023-01-11 [1] CRAN (R 4.2.0)
#> timeDate     4022.108   2023-01-07 [1] CRAN (R 4.2.0)
#> tune         1.1.1      2023-04-11 [1] CRAN (R 4.2.0)
#> tzdb         0.4.0      2023-05-12 [1] CRAN (R 4.2.0)
#> utf8         1.2.3      2023-01-31 [1] CRAN (R 4.2.0)
#> vctrs        0.6.3      2023-06-14 [1] CRAN (R 4.2.0)
#> vip          0.4.0      2023-07-19 [1] CRAN (R 4.2.0)
#> viridisLite  0.4.2      2023-05-02 [1] CRAN (R 4.2.0)
#> warp         0.2.0      2020-10-21 [1] CRAN (R 4.2.0)
#> withr        2.5.0      2022-03-03 [1] CRAN (R 4.2.0)
#> workflows    1.1.3      2023-02-22 [1] CRAN (R 4.2.0)
#> workflowsets 1.0.1      2023-04-06 [1] CRAN (R 4.2.0)
```

```
#>  yardstick     1.2.0     2023-04-21 [1] CRAN (R 4.2.0)
#>
#>  [1] /home/runner/work/_temp/Library
#>  [2] /opt/R/4.2.0/lib/R/site-library
#>  [3] /opt/R/4.2.0/lib/R/library
#>
#> ----------------------------------------------------------------------
```

Preface to the second edition

Welcome to the second edition of *Hands-On Machine Learning with R*! This is a major re-working of the first edition, removing material we no longer think is useful, adding material we wish we included in the first edition, and generally updating the text and code to reflect changes in best practices.

A brief summary of the biggest changes follows:

- TBD
- TBD
- TBD

Part I

Fundamentals

1 Introduction to Machine Learning

Machine learning (ML) continues to grow in importance for many organizations across nearly all domains. Some example applications of machine learning in practice include:

- Predicting the likelihood of a patient returning to the hospital (*readmission*) within 30 days of discharge.
- Segmenting customers based on common attributes or purchasing behavior for targeted marketing.
- Predicting coupon redemption rates for a given marketing campaign.
- Predicting customer churn so an organization can perform preventative intervention.
- And many more!

In essence, these tasks all seek to learn from data. To address each scenario, we can use a given set of *features* to train an algorithm and extract insights. These algorithms, or *learners*, can be classified according to the amount and type of supervision needed during training. The two main groups this book focuses on are: ***supervised learners*** which construct predictive models, and ***unsupervised learners*** which build descriptive models. Which type you will need to use depends on the learning task you hope to accomplish.

1.1 Supervised learning

A ***predictive model*** is used for tasks that involve the prediction of a given output (or target) using other variables (or features) in the data set. Or, as stated by Kuhn and Johnson (2013, 26:2), predictive modeling is "...the process of developing a mathematical tool or model that generates an accurate prediction." The learning algorithm in a predictive model attempts to discover and model the relationships among the target variable (the variable being predicted) and the other features (aka predictor variables). Examples of predictive modeling include:

- using customer attributes to predict the probability of the customer churning in the next 6 weeks;
- using home attributes to predict the sales price;
- using employee attributes to predict the likelihood of attrition;
- using patient attributes and symptoms to predict the risk of readmission;
- using production attributes to predict time to market.

Each of these examples has a defined learning task; they each intend to use attributes (X) to predict an outcome measurement (Y).

> **i** Note
>
> Throughout this text we'll use various terms interchangeably for
>
> - X: "predictor variable", "independent variable", "attribute", "feature", "predictor"
> - Y: "target variable", "dependent variable", "response", "outcome measurement"

The predictive modeling examples above describe what is known as *supervised learning*. The supervision refers to the fact that the target values provide a supervisory role, which indicates to the learner the task it needs to learn. Specifically, given a set of data, the learning algorithm attempts to optimize a function (the algorithmic steps) to find the combination of feature values that results in a predicted value that is as close to the actual target output as possible.

> **i** Note
>
> In supervised learning, the training data you feed the algorithm includes the target values. Consequently, the solutions can be used to help *supervise* the training process to find the optimal algorithm parameters.

Most supervised learning problems can be bucketed into one of two categories, *regression* or *classification*, which we discuss next.

1.1.1 Regression problems

When the objective of our supervised learning is to predict a numeric outcome, we refer to this as a ***regression problem*** (not to be confused with linear regression modeling). Regression problems revolve around predicting output that falls on a continuum. In the examples above, predicting home sales prices and time to market reflect a regression problem because the output is numeric and continuous. This means, given the combination of predictor values, the response value could fall anywhere along some continuous spectrum (e.g., the predicted sales price of a particular home could be between \$80,000 and \$755,000). Figure 1.1 illustrates average home sales prices as a function of two home features: year built and total square footage. Depending on the combination of these two features, the expected home sales price could fall anywhere along a plane.

1.1.2 Classification problems

When the objective of our supervised learning is to predict a categorical outcome, we refer to this as a ***classification problem***. Classification problems most commonly revolve around

Figure 1.1: Average home sales price as a function of year built and total square footage.

predicting a binary or multinomial response measure such as:

- Did a customer redeem a coupon (coded as yes/no or 1/0)?
- Did a customer churn (coded as yes/no or 1/0)?
- Did a customer click on our online ad (coded as yes/no or 1/0)?
- Classifying customer reviews:
    - Binary: positive vs. negative.
    - Multinomial: extremely negative to extremely positive on a 0–5 Likert scale.

However, when we apply machine learning models for classification problems, rather than predict a particular class (i.e., "yes" or "no"), we often want to predict the *probability* of a particular class (i.e., yes: 0.65, no: 0.35). By default, the class with the highest predicted probability becomes the predicted class. Consequently, even though we are performing a classification problem, we are still predicting a numeric output (probability). However, the essence of the problem still makes it a classification problem.

Although there are machine learning algorithms that can be applied to regression problems but not classification and vice versa, most of the supervised learning algorithms we cover in this book can be applied to both. These algorithms have become the most popular machine learning applications in recent years.

Figure 1.2: Classification problem modeling 'Yes'/'No' response based on three features.

1.1.3 Knowledge check

1.2 Unsupervised learning

Unsupervised learning, in contrast to supervised learning, includes a set of statistical tools to better understand and describe your data, but performs the analysis without a target variable. In essence, unsupervised learning is concerned with identifying groups in a data set. The groups may be defined by the rows (i.e., *clustering*) or the columns (i.e., *dimension reduction*); however, the motive in each case is quite different.

The goal of **clustering** is to segment observations into similar groups based on the observed variables; for example, to divide consumers into different homogeneous groups, a process known as market segmentation. In **dimension reduction**, we are often concerned with reducing the number of variables in a data set. For example, classical linear regression models break down in the presence of highly correlated features. Some dimension reduction techniques can be used to reduce the feature set to a potentially smaller set of uncorrelated variables. Such a reduced feature set is often used as input to downstream supervised learning models (e.g., principal component regression).

Unsupervised learning is often performed as part of an exploratory data analysis (EDA). However, the exercise tends to be more subjective, and there is no simple goal for the analysis,

such as prediction of a response. Furthermore, it can be hard to assess the quality of results obtained from unsupervised learning methods. The reason for this is simple. If we fit a predictive model using a supervised learning technique (i.e., linear regression), then it is possible to check our work by seeing how well our model predicts the response Y on observations not used in fitting the model. However, in unsupervised learning, there is no way to check our work because we don't know the true answer—the problem is unsupervised!

Despite its subjectivity, the importance of unsupervised learning should not be overlooked and such techniques are often used in organizations to:

- Divide consumers into different homogeneous groups so that tailored marketing strategies can be developed and deployed for each segment.
- Identify groups of online shoppers with similar browsing and purchase histories, as well as items that are of particular interest to the shoppers within each group. Then an individual shopper can be preferentially shown the items in which he or she is particularly likely to be interested, based on the purchase histories of similar shoppers.
- Identify products that have similar purchasing behavior so that managers can manage them as product groups.

These questions, and many more, can be addressed with unsupervised learning. Moreover, the outputs of unsupervised learning models can be used as inputs to downstream supervised learning models.

1.2.1 Knowledge check

> **Caution**
>
> Identify the type of unsupervised model required for the following tasks:
>
> - Say you have a YouTube channel. You may have a lot of data about the subscribers of your channel. What if you want to use that data to detect groups of similar subscribers?
> - Say you'd like to group Ohio counties together based on the demographics of their residents.
> - A retailer has collected hundreds of attributes about all their customers; however, many of those features are highly correlated. They'd like to reduce the number of features down by combining all those highly correlated features into groups.

1.3 Machine Learning in ⬤

Historically, the R ecosystem provides a wide variety of ML algorithm implementations. This has its benefits; however, this also has drawbacks as it requires the users to learn many different formula interfaces and syntax nuances.

More recently, development on a group of packages called **Tidymodels** has helped to make implementation easier. The **tidymodels** collection allows you to perform discrete parts of the ML workflow with discrete packages:

- rsample for data splitting and resampling
- recipes for data pre-processing and feature engineering
- parsnip for applying algorithms
- tune for hyperparameter tuning
- yardstick for measuring model performance
- and several others!

Throughout this book you'll be exposed to several of these packages. Go ahead and make sure you have the following packages installed.

> **i** Note
>
> The **tidymodels** package is a meta package, or a package of packages, that will install several packages that exist in the **tidymodels** ecosystem.

```r
# data wrangling
install.packages(c("here", "tidyverse"))

# modeling
install.packages("tidymodels")

# model interpretability
install.packages(c("pdp", "vip"))
```

```r
packageVersion("tidymodels")
## [1] '1.1.0'

library(tidymodels)
## -- Attaching packages ------------------------------------- tidymodels 1.1.0 --
## v broom        1.0.5     v rsample      1.1.1
## v dials        1.2.0     v tibble       3.2.1
## v dplyr        1.1.2     v tidyr        1.3.0
```

```
## v infer        1.0.4     v tune         1.1.1
## v modeldata    1.1.0     v workflows    1.1.3
## v parsnip      1.1.0     v workflowsets 1.0.1
## v purrr        1.0.1     v yardstick    1.2.0
## v recipes      1.0.6
## -- Conflicts ------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks plotly::filter(), stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

1.3.1 Knowledge check

> **Caution**
>
> Check out the Tidymodels website: https://www.tidymodels.org/. Identify which packages can be used for:
>
> 1. Efficiently splitting your data
> 2. Optimizing hyperparameters
> 3. Measuring the effectiveness of your model
> 4. Working with correlation matrices

1.4 Roadmap

The goal of this book is to provide effective tools for uncovering relevant and useful patterns in your data by using R's ML stack. We begin by providing an overview of the ML modeling process and discussing fundamental concepts that will carry through the rest of the book. These include feature engineering, data splitting, model validation and tuning, and performance measurement. These concepts will be discussed in Chapters …

> ⚠️ **TODO**
>
> Fill out roadmap as we progress

1.5 Data sets

TBD

1.6 Exercises

1. Identify four real-life applications of supervised and unsupervised problems.

   - Explain what makes these problems supervised versus unsupervised.
   - For each problem identify the target variable (if applicable) and potential features.

2. Identify and contrast a regression problem with a classification problem.

   - What is the target variable in each problem and why would being able to accurately predict this target be beneficial to society?
   - What are potential features and where could you collect this information?
   - What is determining if the problem is a regression or a classification problem?

3. Identify three open source data sets suitable for machine learning (e.g., https://bit.ly/35wKu5c).

   - Explain the type of machine learning models that could be constructed from the data (e.g., supervised versus unsupervised and regression versus classification).
   - What are the dimensions of the data?
   - Is there a code book that explains who collected the data, why it was originally collected, and what each variable represents?
   - If the data set is suitable for supervised learning, which variable(s) could be considered as a useful target? Which variable(s) could be considered as features?

4. Identify examples of misuse of machine learning in society. What was the ethical concern?

2 Modeling Process

Much like EDA, the ML process is very iterative and heurstic-based. With minimal knowledge of the problem or data at hand, it is difficult to know which ML method will perform best. This is known as the *no free lunch* theorem for ML (Wolpert 1996). Consequently, it is common for many ML approaches to be applied, evaluated, and modified before a final, optimal model can be determined. Performing this process correctly provides great confidence in our outcomes. If not, the results will be useless and, potentially, damaging [1].

Approaching ML modeling correctly means approaching it strategically by spending our data wisely on learning and validation procedures, properly pre-processing the feature and target variables, minimizing *data leakage*, tuning hyperparameters, and assessing model performance. Many books and courses portray the modeling process as a short sprint. A better analogy would be a marathon where many iterations of these steps are repeated before eventually finding the final optimal model. This process is illustrated in Figure 2.1.
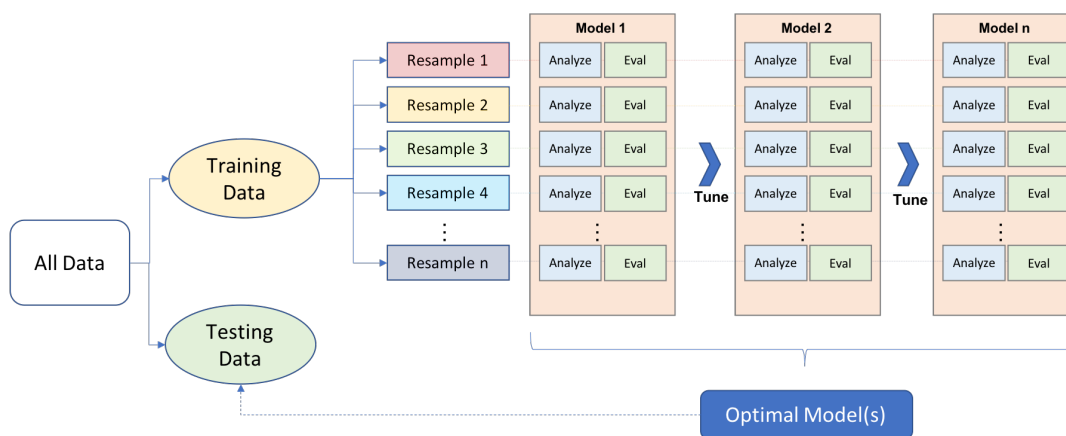

Figure 2.1: General predictive machine learning process.

Before introducing specific algorithms, this chapter, and the next, introduce concepts that are fundamental to the ML modeling process and that you'll see briskly covered in future modeling chapters. More specifically, this chapter is designed to get you acquainted with building predictive models using the Tidymodels construct. We'll focus on the process of splitting our

---

[1]See https://www.fatml.org/resources/relevant-scholarship for many discussions regarding implications of poorly applied and interpreted ML.

data and applying resampling procedures for improved generalizability, training a basic model, tuning a model's hyperparameter(s), and, finally, evaluating a model's performance.

> **ⓘ Note**
>
> Although the discussions in this chapter focus on supervised ML modeling, many of the topics also apply to unsupervised methods.

References

Efron, Bradley, and Trevor Hastie. 2016. *Computer Age Statistical Inference.* Vol. 5. Cambridge University Press.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* Vol. 1. MIT Press Cambridge.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Vol. 2. Springer Science+ Business Media.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling.* Vol. 26. Springer.

Wickham, Hadley. 2014. *Advanced r.* CRC Press.

Wickham, Hadley, and Garrett Grolemund. 2016. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* O'Reilly Media, Inc.

Wolpert, David H. 1996. "The Lack of a Priori Distinctions Between Learning Algorithms." *Neural Computation* 8 (7): 1341–90.