797ML Handbook

Steve Linberg

2022-04-06

Contents

Chapter 1

About

This book is being written as part of a final project for 797ML at UMass Amherst, spring 2022. It contains a simple reference and breakdown for a couple of dozen core methods used in machine learning.

The intent is twofold:

1. Serve as a reference for the basics of the material covered in the class, using language and examples that are as simple as possible to explain the core concepts and how to do them;
2. Force myself to learn these techniques better by carrying out the above.

The main purpose of this work is to be *simple*, not to be *comprehensive*. We won't cover every facet of every technique, or every possible permutations of outcomes. The goal is to simply express the broad strokes and core concepts in a way that can be easily remembered, and to serve as a jumping-off point when more information is needed.

1.1 Authoring guidelines

The goal is to have no more than a few short paragraphs for each section, and to keep each explanation of the meanings of outcome variables to one sentence each.

Each chapter will have a TL;DR section at the top with one-sentence answers to following questions:

- What it does
- When to do it
- How to do it
- How to assess it

and it will be possible to skim through the book, just surfing the TL;DR at the top of each chapter, and get a reasonable overview without reading the rest. The rest of each chapter will deliver more fleshed-out, but *short*, answers to the same questions, covering the basics and the most general concepts.

1.2   Contact

Steve Linberg: steve@slinberg.net || https://slinberg.net

Project home: https://stevelinberg.github.io/797MS-handbook/

Github repo: https://github.com/stevelinberg/797MS-handbook

1.3   Resources

A lot of the material from this work is from the class textbook, James et al. [2021]. I also find UNC geneticist Josh Starmer's StatQuest video series on YouTube to be immensely helpful for simple explanations of statistics and machine learning concepts.

Chapter 2

Simple Linear Regression

2.1 TL;DR

What it does Looks to see how well a single predictor variable predicts an outcome, like *how well do years of education predict salary?*

When to do it When you want to see if pretty much the simplest possible model provides enough of an explanation of variance for your purposes

How to do it With the `lm()` function, among other ways

How to assess it Look for a significant p-value for the predictor, and a reasonable R^2

Note "Linear" does not necessarily mean "straight" in this context; a linear regression line can curve.

2.2 What it does

Simple linear regression is where it all begins; among the simplest of all of the regression techniques in analysis, which attempts to estimate a slope and an intercept line for a set of observations using a single predictor variable X and an output variable Y. It uses ordinary least squares (OLS) to build its model, looking for the line through the mean of X and Y that has the smallest sum of squares between the predicted and observed values.

The figure above shows a plot of a simple linear regression, attempting to use the variable `TV` to predict `Sales.` The blue line is the line defined by the regression with its Y intercept and slope; the red dots are the actual observations of `Sales` for each measure of the predictor `TV` on the X axis. The thin blue lines are the error in the prediction.
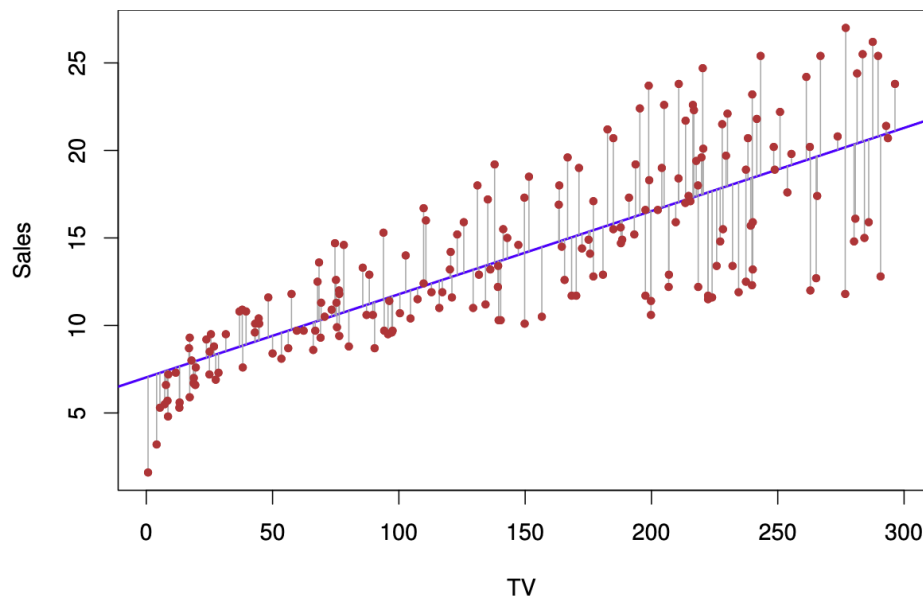
Figure 2.1: Simple linear regression plot (source: James et al. [2021], p. 62)

2.3 When to do it

It is a simple first step for looking at data to see if there is an easy single-variable model that does a reasonable job predicting outcomes using one predictor variable. Sometimes, it can be good enough! It has the advantage of being easy to execute, to understand and to communicate, and the value of these factors should not be underestimated. Communicating with non-specialists is an important aspect of a data scientist's job.

Linear regression requires a dataset with a continuous outcome variable; it is easiest and most effective if the predictor variable is also numeric, whether continuous or discrete. It is possible to do linear regression with non-numeric predictors, such as true/false or ordered responses, by converting the predictors to a numeric scale.

2.4 How to do it

This example from James et al. [2021] shows a simple linear regression of **medv** onto **lstat**, attempting to predict median housing prices from percentage of "lower-status" population in the Boston data set from the late 1970s.

```
lm.fit <- lm(medv ~ lstat, data = Boston)
plot(Boston$lstat, Boston$medv)
abline(lm.fit, lwd = 3, col = "red")
```
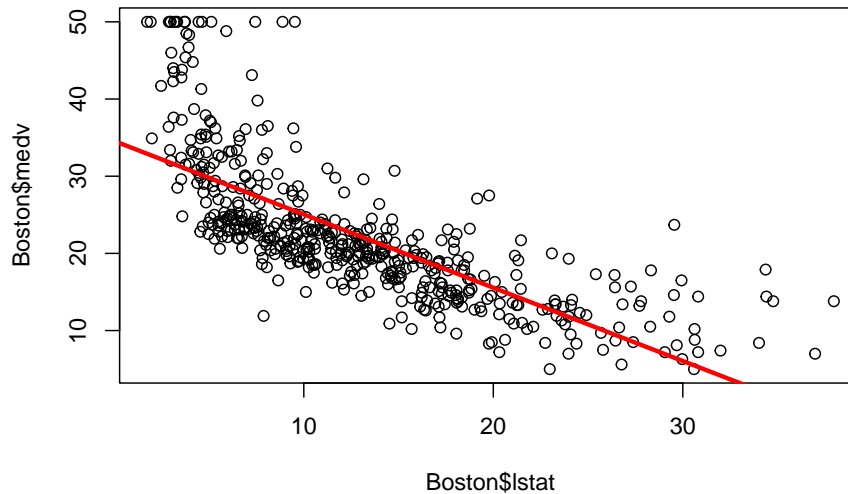
The results of the regression are stored in the output of `lm()`, and may be viewed with `summary()`:

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

2.5   How to assess it

The output contains a lot of information, but the key points are:

- `Adjusted R-squared`: 0.5432 is the percentage of the variation in the model that is explained by the fit's prediction, compared to just looking at how much the observations vary from their own average with no predictor variable.
- `Residual standard error`: 6.216 is "the average amount that the response will deviate from the true regression line", or the standard deviation of the error. In this case, since the unit of `medv` is in thousands of dollars, it means the average prediction will be off by \$6,216.
- `p-value`: < 2.2e-16 (basically zero) means that the model is (very) statistically significant, with a near-zero percent chance that the data in this set could have resulted from a random draw from the (unknown) population if there were no relationship between `medv` and `lstat`.
- `Estimated intercept` of 34.55 is the Y intercept on the graph, or the estimated value of `medv` when `lstat` is zero.
- `Estimated (lstat)`: -0.95 is the estimated coefficient of `lstat`, or the amount that `medv` changes by for each unit change in `lstat`.

The intercept and coefficient are β_0 and β_1, respectively, of the linear regression formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

Substituting the coefficients and variables, we transform this to:

$$\widehat{\text{medv}} = 34.55 - 0.95 \times \text{lstat}$$

The low p-value shows that there is (almost) definitely a relationship between `medv` and `lstat`, but the R^2 of 0.54 shows that the model only explains slightly more than half of the variation in the data. It's not a bad start, but we would probably want to find a model that explains more of it.

2.6   Where to learn more

- Chapter 3 - 3.1 in James et al. [2021]
- StatQuest: Linear Regression - this goes into good depth on the meaning of the F statistic as well, and how it used to calculate the p value.

2.7   Notes

You can also do a scatterplot in ggplot2 and add a regression line with `geom_smooth()`; it doesn't show the coefficients or other output information, but it gives a quick visual that's a little prettier than the base R plot:

```r
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x) +
  geom_hline(yintercept = mean(Boston$medv), linetype = "dashed", color = "red") +
  geom_vline(xintercept = mean(Boston$lstat), linetype = "dashed", color = "darkgreen") +
  theme_minimal()
```
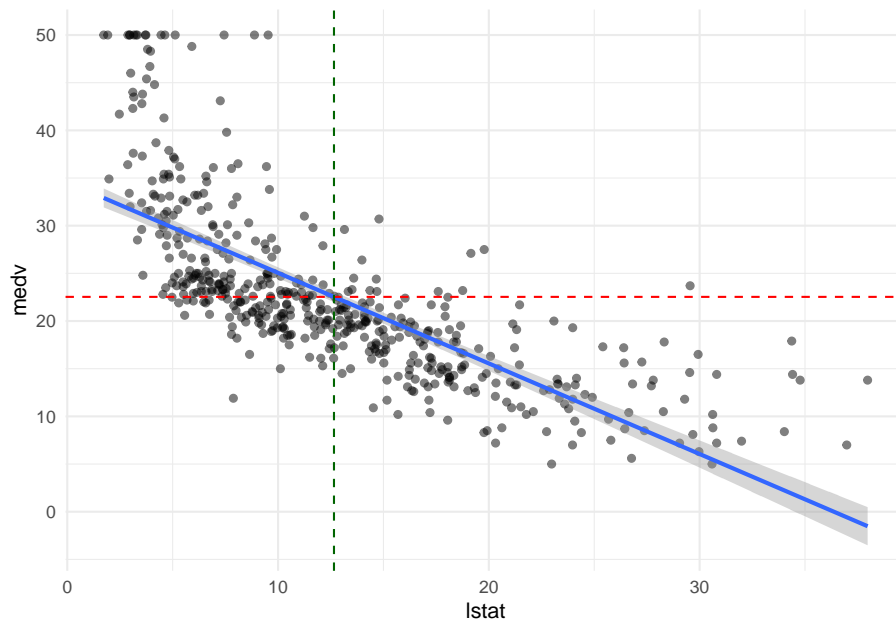

The means of `lstat` and `medv` are shown with dashed red and green lines, showing that the regression line goes through the center of the data.

Chapter 3

Multiple Linear Regression

3.1 TL;DR

What it does Looks to see how well multiple predictor variables predict an outcome, like *how well do years of education and age predict salary?*

When to do it When a simple linear regression doesn't provide a good enough explanation of variance, and you want to see if adding additional variables provides a better one

How to do it With the `lm()` function, utilizing more than one predictor

How to assess it Look for significant p-values for the predictors, and a reasonable adjusted-R^2

3.2 What it does

Multiple linear regression is the first natural extension of simple linear regression. It allows for more than one predictor variable to be specified. It is also possible to combine predictors in interactions, to find out if combinations of predictors have different effects than simply adding them to the model. XXX explain/demo

3.3 When to do it

Use multiple linear regression when a simple linear regression doesn't provide a good enough explanation of the variance you're observing, and you want to see if adding more predictors provides a better fit. Typically, this would be in response to either a low R^2 that leaves a lot of unexplained variance, or even just a visual conclusion drawn from seeing a plot of a linear model with an unsatisfactory regression line.

3.4   How to do it

Use the same `lm()` function, with more than one predictor in the formula.

```
lm.fit <- lm(medv ~ lstat + age, data = Boston)
plot(Boston$lstat, Boston$medv)
abline(lm.fit, lwd = 3, col = "red")
```
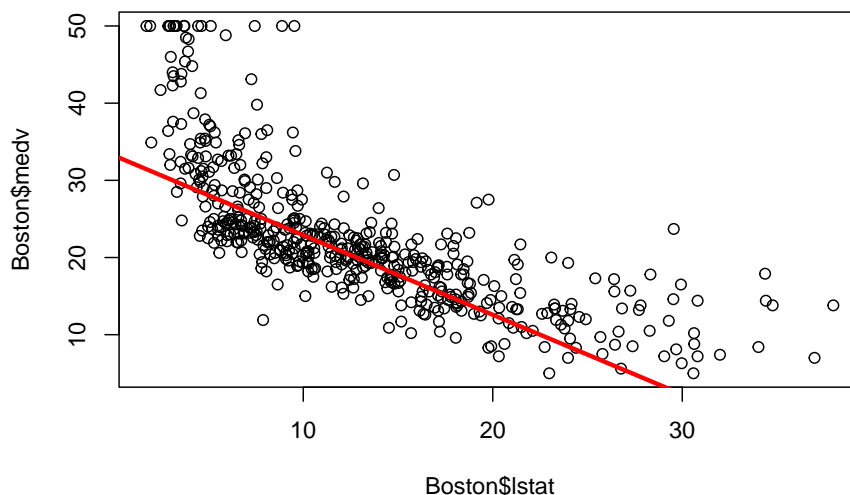

As with simplr linear regression, the results of the regression are stored in the output of `lm()`, and may be viewed with `summary()`:

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

All of the same parameters are there, with the addition of one coefficient line for each parameter specified in the model. Each coefficient has its own estimate, standard error, t value and p value, and should be evaluated independently.

The main task is to get the ideal subset of parameters that provide the best fit, which can involve a lot of trial and error. *Forward selection* involves starting with a null model (no parameters) and trying all available parameters to see which has the lowest p-value, and adding that and continuing until the parameters are no longer significant or don't reduce the adjusted R^2; *Backward selection* starts with all parameters and removes the least significant one at a time until all remaining parameters are significant; *Mixed selection* starts with forward selection but also removes parameters that lose their significance along the way.

Other tidbits:

- *interaction terms* can be specified with syntax like `lstat * age`, which adds `lstat`, `age` and `lstat * age` as predictors; if an interaction is significant but an individual predictor from the interaction isn't, it should still be left in the model
- *nonlinear transformations* like age^2 can be added by escaping them inside the I() function, like `I(age^2)`, to escape the `^` character inside the formula

3.5 How to assess it

With the exception of the additional information for each parameter, the assessment is the same as for simple linear regression.

In the example above, we see that the addition of `age` only added a tiny bit of improvement to the model from the simple linear regression on just `lstat`. Other variables, or combinations of variables, might do a better job.

3.6 Where to learn more

- Chapter 3.2 in James et al. [2021]

Chapter 4

Logistic Regression

4.1 TL;DR

What it does Models the *probability* that an observation falls into one of two
categories

When to do it When you are trying to predict a categorical variable with two
possible outcomes, like true/false, instead of something continuous like in
linear regression

How to do it With the `glm()` function, specifying `family = "binomial"`

How to assess it Look for a significant p-value for the predictor, and assess
its accuracy against training data

4.2 What it does

Logistic regression is used to make a prediction about whether an observation
falls into one of two categories. This is an alternative to linear regression, which
predicts a continuous outcome. A logistic regression results in a model that
gives the probability of Y given X.

4.3 When to do it

As the TL;DR says: When you are trying to predict a categorical variable with
two possible outcomes, like true/false, instead of something continuous.

4.4 How to do it

First, it's important to note that logistic regression will give better results when
the model is fit using *held out* or *training* data, and then tested against other
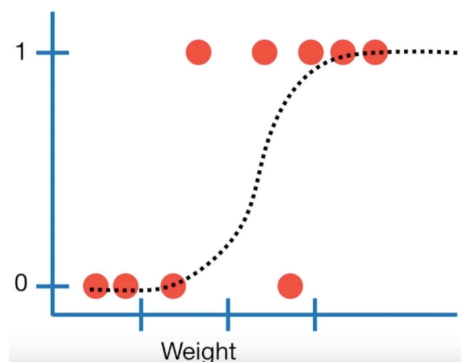
Figure 4.1: Linear regression plot (source: StatQuest)

data, to help avoid overfitting. It is not required, but it's recommended where possible.

First, before that, fit a model using glm, making sure in this case that we have a categorical value to test for:

```
data(Boston)
boston <- Boston %>%
  mutate(
    # Create the categorical crim_above_med response variable
    crim_above_med = as.factor(ifelse(crim > median(crim), "Yes", "No")),
    # Also make a numeric version of crim_above_med for the plot
    crim_above_med_num = ifelse(crim > median(crim), 1, 0)
  )
boston.fits <- glm(crim_above_med ~ nox, data = boston, family = binomial)
summary(boston.fits)
```

```
##
## Call:
## glm(formula = crim_above_med ~ nox, family = binomial, data = boston)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.27324  -0.37245  -0.06847   0.39620   2.53124
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -15.818      1.386  -11.41   <2e-16 ***
## nox           29.365      2.599   11.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.46  on 505  degrees of freedom
## Residual deviance: 320.39  on 504  degrees of freedom
## AIC: 324.39
##
## Number of Fisher Scoring iterations: 6
```

We can see the that p value for the `nox` variable is nearly 0, suggesting a strong association between `nox` and `crim_above_med`.

Now split the data into training and test sets, using, say, 80% of the data for training:

```
set.seed(1235)
boston.training <- rep(FALSE, nrow(boston))
boston.training[sample(nrow(boston), nrow(boston) * 0.8)] <- TRUE
boston.test <- !boston.training
```

Fit the above model to the training data:

```
boston.fits <-
  glm(
    crim_above_med ~ nox,
    data = boston,
    subset = boston.training,
    family = binomial
  )
```

And use the fit to predict the test data with `predict()`:

```
boston.probs <- predict(boston.fits, boston[boston.test,], type = "response")
```

4.5 How to assess it

Assess the accuracy of the predictions by building a confusion matrix, using the same values ("No" and "Yes" in this case) as contained in the outcome variable we want to predict:

```
boston.pred <- rep("No", sum(boston.test))
boston.pred[boston.probs > 0.5] <- "Yes"
table(boston.pred, boston[boston.test,]$crim_above_med)
```

```
##
## boston.pred No Yes
##         No  36  12
##         Yes  5  49
```

The above table shows that the model correctly predicted `crim_above_med` from

nox 85 times out of 102, with 5 false positives and 12 false negatives. The accuracy of the model can be computed via the `mean()` of how often the predictions match the test data, a neat trick:

```
mean(boston.pred == boston[boston.test,]$crim_above_med)
```

```
## [1] 0.8333333
```

The regression can be plotted with `ggplot2` and the `stat_smooth()` function, utilizing the `glm` method, like so:

```
ggplot(boston, aes(x = nox, y = crim_above_med_num)) +
  geom_point(alpha = .5) +
  stat_smooth(
    method = "glm",
    formula = y ~ x,
    se = FALSE,
    method.args = list(family = binomial),
    col = "red",
    lty = 2
  )
```
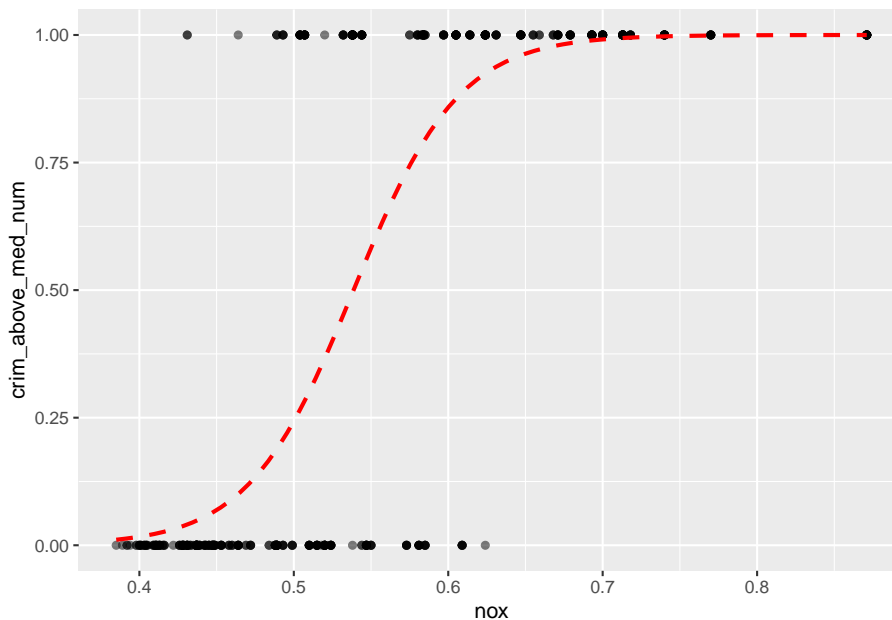

This shows that an observation with a `nox` value of 0.6 would have around a 90% probability of `crim_above_med` being true, or 1, while an observation with a `nox` value of 0.5 would only have around a 25% probability of `crim_above_med` being true.

4.6 Where to learn more

- Chapter 4 - 4.3 in James et al. [2021]
- StatQuest: Logistic Regression

Chapter 5

TODO: Multiple Logistic Regression

5.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

5.2 What it does

TODO

5.3 When to do it

TODO

5.4 How to do it

TODO

5.5 How to assess it

TODO

5.6   Where to learn more

TODO

Chapter 6

TODO: Linear Discriminant Analysis

6.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

6.2 What it does

TODO

6.3 When to do it

TODO

6.4 How to do it

TODO

6.5 How to assess it

TODO

6.6   Where to learn more

TODO

Chapter 7

TODO: Quadratic Discriminant Analysis

7.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

7.2 What it does

TODO

7.3 When to do it

TODO

7.4 How to do it

TODO

7.5 How to assess it

TODO

7.6   Where to learn more

TODO

Chapter 8

TODO: Naive Bayes

8.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

8.2 What it does

TODO

8.3 When to do it

TODO

8.4 How to do it

TODO

8.5 How to assess it

TODO

8.6 Where to learn more

TODO

Chapter 9

TODO: K-Nearest Neighbors

9.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

9.2 What it does

TODO

9.3 When to do it

TODO

9.4 How to do it

TODO

9.5 How to assess it

TODO

9.6   Where to learn more

TODO

Chapter 10

TODO: Poisson Regression

10.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

10.2 What it does

TODO

10.3 When to do it

TODO

10.4 How to do it

TODO

10.5 How to assess it

TODO

10.6 Where to learn more

TODO

Chapter 11

TODO: Cross-Validation

11.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

11.2 What it does

TODO

11.3 When to do it

TODO

11.4 How to do it

TODO

11.5 How to assess it

TODO

11.6 Where to learn more

TODO

Chapter 12

TODO: Bootstrap

12.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

12.2 What it does

TODO

12.3 When to do it

TODO

12.4 How to do it

TODO

12.5 How to assess it

TODO

12.6 Where to learn more

TODO

Chapter 13

TODO: Best Subset Selection

13.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

13.2 What it does

TODO

13.3 When to do it

TODO

13.4 How to do it

TODO

13.5 How to assess it

TODO

13.6   Where to learn more

TODO

Chapter 14

TODO: Stepwise Selection

14.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

14.2 What it does

TODO

14.3 When to do it

TODO

14.4 How to do it

TODO

14.5 How to assess it

TODO

14.6 Where to learn more

TODO

Chapter 15

TODO: Ridge Regression

15.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

15.2 What it does

TODO

15.3 When to do it

TODO

15.4 How to do it

TODO

15.5 How to assess it

TODO

15.6 Where to learn more

TODO

Chapter 16

TODO: Lasso

16.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

16.2 What it does

TODO

16.3 When to do it

TODO

16.4 How to do it

TODO

16.5 How to assess it

TODO

16.6 Where to learn more

TODO

Chapter 17

TODO: Principal Component Regression

17.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

17.2 What it does

TODO

17.3 When to do it

TODO

17.4 How to do it

TODO

17.5 How to assess it

TODO

17.6   Where to learn more

TODO

Chapter 18

TODO: Bagging

18.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

18.2 What it does

TODO

18.3 When to do it

TODO

18.4 How to do it

TODO

18.5 How to assess it

TODO

18.6 Where to learn more

TODO

Chapter 19

TODO: Random Forests

19.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

19.2 What it does

TODO

19.3 When to do it

TODO

19.4 How to do it

TODO

19.5 How to assess it

TODO

19.6 Where to learn more

TODO

Chapter 20

TODO: Boosting

20.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

20.2 What it does

TODO

20.3 When to do it

TODO

20.4 How to do it

TODO

20.5 How to assess it

TODO

20.6 Where to learn more

TODO

Chapter 21

TODO: Bayesian Additive Regression Trees

21.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

21.2 What it does

TODO

21.3 When to do it

TODO

21.4 How to do it

TODO

21.5 How to assess it

TODO

21.6   Where to learn more

TODO

Chapter 22

TODO: Support Vector Machines

22.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

22.2 What it does

TODO

22.3 When to do it

TODO

22.4 How to do it

TODO

22.5 How to assess it

TODO

22.6   Where to learn more

TODO

Chapter 23

TODO: Principal Component Analysis

23.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

23.2 What it does

TODO

23.3 When to do it

TODO

23.4 How to do it

TODO

23.5 How to assess it

TODO

23.6　Where to learn more

TODO

Chapter 24

TODO: K-Means Clustering

24.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

24.2 What it does

TODO

24.3 When to do it

TODO

24.4 How to do it

TODO

24.5 How to assess it

TODO

24.6   Where to learn more

TODO

Chapter 25

TODO: Hierarchical Clustering

25.1 TL;DR

What it does TODO
When to do it TODO
How to do it TODO
How to assess it TODO

25.2 What it does

TODO

25.3 When to do it

TODO

25.4 How to do it

TODO

25.5 How to assess it

TODO

25.6   Where to learn more

TODO

Bibliography

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2nd edition, 2021. URL https://link.springer.com/book/10.1007/978-1-0716-1418-1. ISBN 978-1-0716-1417-4.