Advanced Analytics with Tidymodels

Prof. Dr. Jan Kirenz

2020-11-05

Contents

Welcome

This book provides an introduction to advanced analytics with R using the modeling packages called tidymodels to build, evaluate, compare, and tune predictive models.

We'll cover key concepts in statistical learning and machine learning including overfitting, the holdout method, the bias-variance trade-off, ensembling, cross-validation, and feature engineering.

Part I

BUILD A MODEL

Chapter 1

Build a Model

This tutorial is based on Alisson Hill's excellent tidymodels workshop.

In this tutorial you will learn how to specify a model with the tidymodels package.

As data, we use 2,930 houses sold in Ames, IA from 2006 to 2010, collected by the Ames Assessor's Office. From this dataset, we only select our dependent variable (`Sale_Price`) and one predictor variable (`Gr_Liv_Area`).

```
library(tidyverse)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
ames <- read_csv("https://raw.githubusercontent.com/kirenz/datasets/master/ames.csv")
```

```
## Parsed with column specification:
```

```
## cols(
##   .default = col_character(),
##   Lot_Frontage = col_double(),
##   Lot_Area = col_double(),
##   Year_Built = col_double(),
##   Year_Remod_Add = col_double(),
##   Mas_Vnr_Area = col_double(),
##   BsmtFin_SF_1 = col_double(),
##   BsmtFin_SF_2 = col_double(),
##   Bsmt_Unf_SF = col_double(),
##   Total_Bsmt_SF = col_double(),
##   First_Flr_SF = col_double(),
##   Second_Flr_SF = col_double(),
##   Low_Qual_Fin_SF = col_double(),
##   Gr_Liv_Area = col_double(),
##   Bsmt_Full_Bath = col_double(),
##   Bsmt_Half_Bath = col_double(),
##   Full_Bath = col_double(),
##   Half_Bath = col_double(),
##   Bedroom_AbvGr = col_double(),
##   Kitchen_AbvGr = col_double(),
##   TotRms_AbvGrd = col_double()
##   # ... with 15 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```r
ames <-
  ames %>%
  select(Sale_Price, Gr_Liv_Area)

glimpse(ames)
```

```
## Rows: 2,930
## Columns: 2
## $ Sale_Price  <dbl> 215000, 105000, 172000, 244000, 189900, 195500, 213500,...
## $ Gr_Liv_Area <dbl> 1656, 896, 1329, 2110, 1629, 1604, 1338, 1280, 1616, 18...
```

In this first example, we don't use data splitting.

1.1 Model type

1. Pick an `model type`: choose from this list

2. Set the `engine`: choose from this list
3. Set the `mode`: regression or classification

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.0 --

## v broom     0.5.6      v rsample   0.0.7
## v dials     0.0.7      v tune      0.1.0
## v infer     0.5.2      v workflows 0.1.1
## v parsnip   0.1.2      v yardstick 0.0.6
## v recipes   0.1.13

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```r
lm_spec <- # your model specification
  linear_reg() %>%  # model type
  set_engine(engine = "lm") %>%  # model engine
  set_mode("regression") # model mode

# Show your model specification
lm_spec
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

1.2 Model training

In the training process, you run an algorithm on data and thereby produce a model. This process is also called model fitting.

```r
lm_fit <- # your fitted model
  fit(
  lm_spec, # your model specification
  Sale_Price ~ Gr_Liv_Area, # a Linear Regression formula
  data = ames # your data
  )

# Show your fitted model
lm_fit
```

```
## parsnip model object
##
## Fit time:  4ms
##
## Call:
## stats::lm(formula = Sale_Price ~ Gr_Liv_Area, data = data)
##
## Coefficients:
## (Intercept)  Gr_Liv_Area
##     13289.6        111.7
```

1.3 Model predictions

We use our fitted model to make predictions.

```
price_pred <-
  lm_fit %>%
  predict(new_data = ames) %>%
  mutate(price_truth = ames$Sale_Price)

head(price_pred)
```

```
## # A tibble: 6 x 2
##     .pred price_truth
##     <dbl>       <dbl>
## 1 198255.      215000
## 2 113367.      105000
## 3 161731.      172000
## 4 248964.      244000
## 5 195239.      189900
## 6 192447.      195500
```

If we would want to make predictions for new houses, we could proceed as follows:

```
# New values (our x variable)
new_homes <-
  tibble(Gr_Liv_Area = c(334, 1126, 1442, 1500, 1743, 5642))

# Prediction for new houses (predict y)
lm_fit %>%
 predict(new_data = new_homes)
```

```
## # A tibble: 6 x 1
##      .pred
##      <dbl>
## 1  50595.
## 2 139057.
## 3 174352.
## 4 180831.
## 5 207972.
## 6 643467.
```

1.4 Model evaluation

We use the Root Mean Squared Error (RMSE) to evaluate our regression model. Therefore, we use the function $rmse(data, truth, estimate)$.

```
rmse(data = price_pred,
     truth = price_truth,
     estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      56505.
```

Chapter 2

Process with data splitting

The best way to measure a model's performance at predicting new data is to actually predict new data.

This function "splits" data randomly into a single testing and a single training set: `initial_split(data, prop = 3/4, strata, breaks)`. We also use stratified sampling in this example.

2.1 Data splitting

```
set.seed(100)

ames_split <-  initial_split(ames,
                             strata = Sale_Price,
                             breaks = 4)

ames_train <-  training(ames_split)
ames_test <- testing(ames_split)
```

2.2 Model type

```
lm_spec_2 <-
  linear_reg() %>%
  set_engine(engine = "lm") %>%
  set_mode("regression")
```

2.3 Model training

```
lm_fit_2 <-
  lm_spec_2 %>%
  fit(Sale_Price ~ Gr_Liv_Area,
      data = ames_train) # only use training data
```

2.4 Model predictions

```
price_pred_2 <-
  lm_fit %>%
  predict(new_data = ames_test) %>%
  mutate(price_truth = ames_test$Sale_Price)
```

2.5 Model evaluation

```
rmse(price_pred_2,
     truth = price_truth,
     estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      59054.
```

Chapter 3

Import

Chapter 4

Transformation

Part II

BASICS

Chapter 5

Build a model

Chapter 6

Build models with resampling