Prostate Cancer

Lavanya N

3/2/2021

Linear regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It is used to predict values within a continuous range.

The `set.seed` function is called once to ensure that the knitting is conditionally deterministic.

```
set.seed(12345)
```

A linear regression model will be trained and tested using the `prostate` dataset. This data comes from a study that examined the correlation between the level of prostate specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. It contains 97 observations of 9 variables. The 10th variable will be used to split the data into training and testing data. The exact details of the variables can be viewed here: https://rafalab.github.io/pages/649/prostate.html.

```
prostate <- read.table("http://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data",
                       header = TRUE)
str(prostate, max.level = 1)
```

```
## 'data.frame':    97 obs. of  10 variables:
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
##  $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
##  $ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

In supervised learning, training data is used to solve some optimization problem. A corresponding matrix of predictors and outcome vector form the testing data, which are additional observations that are not used to obtain regression parameters and are instead used to evaluate the model that produced them. It is important to ensure that the trained model is tested on a different set of data to avoid biasing.

The prostate dataset already has a logical vector called `train` that can be used to separate training data from testing. The column `train` is removed from both the training and testing data as it is not relevant for the model.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
training <- select(filter(prostate, train), -train)
testing <- select(filter(prostate, !train), -train)
```

The training data consists of 67 observations and the testing data consists of 30 observations.

```
c(training = nrow(training), testing = nrow(testing))
```

```
## training  testing
##       67       30
```

The data is first prepared for modeling. In this step, variables to be included in the model should be specified. All variables are being included in this model.

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------- tidymodels 0.1.2.9000 --

## v broom     0.7.2      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v ggplot2   3.3.2      v tibble    3.0.4
## v infer     0.5.3      v tidyr     1.1.2
## v modeldata 0.1.0      v tune      0.1.2
## v parsnip   0.1.4      v workflows 0.2.1
## v purrr     0.3.4      v yardstick 0.0.7

## Warning: package 'broom' was built under R version 4.0.3

## Warning: package 'dials' was built under R version 4.0.3

## Warning: package 'infer' was built under R version 4.0.3

## Warning: package 'modeldata' was built under R version 4.0.3

## Warning: package 'parsnip' was built under R version 4.0.3

## Warning: package 'recipes' was built under R version 4.0.3

## Warning: package 'rsample' was built under R version 4.0.3

## Warning: package 'tibble' was built under R version 4.0.3
## Warning: package 'tune' was built under R version 4.0.3

## Warning: package 'workflows' was built under R version 4.0.3

## Warning: package 'yardstick' was built under R version 4.0.3

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
```

```
(simple <- recipe(lpsa ~ ., data = training))
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          8
```

```
(simple <- prep(simple, training = training))
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          8
##
## Training data contained 67 data points and no missing data.
```

Subsequently, the type of model or algorithm should be specified. In this case, it is a linear regression model.

```
lm_model <- linear_reg() %>% set_engine("lm")
lm_wflow <- workflow() %>% add_model(lm_model) %>% add_formula(lpsa ~ .)
lm_fit <- fit(lm_wflow, data = bake(simple, new_data = NULL))
```

Finally, predictions of the model can be obtained in the testing data and the root-mean squared error is calculated. In this case, it is 0.722. This value has more meaning when there are multiple linear models (possibly with different combinations of the variables) to compare for the same purpose.

```
baked <- bake(simple, new_data = testing)
select(baked, lpsa) %>% bind_cols(predict(lm_fit, new_data = baked)) %>%
  rmse(truth = lpsa, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.722
```