Predicting Apartment Prices

Lavanya N

3/3/2021

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)
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 dplyr     1.0.2      v tibble    3.0.4
## v ggplot2   3.3.2      v tidyr     1.1.2
## v infer     0.5.3      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7
## v purrr     0.3.4
```

```
## 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()
```

`training_apts.rds` and `testing_apts.rds` are 2 dataframes that each contain randomly selected observations on apartments for purchase in a Western European city in 2005. The dependent variable is `totalprice`, which is the purchase price of the apartment in Euros. The possible predictors are:

- `area` the number of square meters in the apartment
- `zone` an unordered factor indicating what neighborhood the apartment is in
- `category` an ordered factor indicating the condition of the apartment
- `age` number of years since the apartment was built
- `floor` the floor of the building where the apartment is located
- `rooms` the total number of rooms in the apartment
- `out` an ordered factor indicating what percentage of the apartment's exterior is exposed to the outside
- `conservation` an ordered factor indicating how well the apartment is conserved
- `toilets` a count
- `garage` a count, i.e. some apartments have two garages
- `elevator` a binary variable
- `streetcategory` an ordered factor that captures the quality of the street the apartment building is on
- `heating` an unordered factor indicating something about the presence or absence of (possibly central) heating for the apartment
- `storage` a count of the number of storage rooms for the apartment

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. In this case, the training and testing data are already available separately.

```r
training <- readRDS("training_apts.rds")
testing  <- readRDS("testing_apts.rds")
```

```r
training$toilets <- as.factor(training$toilets)
training$storage <- as.factor(training$storage)
testing$toilets <- as.factor(testing$toilets)
testing$storage <- as.factor(testing$storage)
training$totalprice <- training$totalprice / 1000 # expressing price in 1000s of Euros
testing$totalprice <- testing$totalprice / 1000
```

The standard linear model may perform poorly in situations with a large multivariate dataset that has a smaller number of observations. In such scenarios, a penalised regression model may be a better alternative as it allows the addition of a constraint in the equation and creation of a model that is penalised for having too many variables/fewer observations. The consequence of imposing this penalty, is to reduce (i.e. shrink) the coefficient values towards zero. This allows the less contributive variables to have a coefficient close to zero or equal zero. The shrinkage requires the selection of a tuning parameter (lambda) that determines the amount of shrinkage. Tuning parameters can be pre-specified or selected optimally.

Since variables are being compared to one another but have different units, they should be standardised using step_center and step_scale. The penalty is also pre-specified to have a value of 0.001.

```
norm_recipe <- recipe(totalprice ~ ., data = training) %>%
  step_zv(all_predictors()) %>%
  step_lincomb(all_numeric()) %>%
  step_dummy(all_nominal()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  prep(training = training)
```

Instead of specifying the tuning parameters, grid search can be employed to find the optimal parameters. This performs cross-validation to see which value of the tuning parameter yields the best predictions outside the data that are used to obtain the coefficients. In this, the optimal parameter is chosen from 50 possible options.

```
apts_rs <- bootstraps(training, times = 50)
tune_spec <- linear_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")
wf <- workflow() %>% add_recipe(norm_recipe) %>%
  remove_formula() %>% add_formula(totalprice ~ .)
```

```
## Warning: The workflow has no formula preprocessor to remove.
```

```
my_grid <- grid_regular(penalty(), mixture(), levels = 10)
doParallel::registerDoParallel()
```

The value of the tuning parameter is selected such that the resulting regression model has the lowest rmse.

```
results <- tune_grid(wf %>% add_model(tune_spec),
                     resamples = apts_rs, grid = my_grid)
lowest_rmse <- results %>% select_best("rmse")
final <- finalize_workflow(wf %>% add_model(tune_spec), lowest_rmse)
```

```
final_fit <- fit(final, data = training)
```

The penalised regression model using grid search has a rmse of 28.545.

```
baked <- bake(norm_recipe, new_data = testing)
bind_cols(select(baked, totalprice),
          predict(final_fit, new_data = testing)) %>%
  rmse(truth = totalprice, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        28.5
```