Predicting price

Rawi Baransy

2023-09-13

```r
library(neuralnet)
library(kknn)
library(data.table)
library(rpart)
library(ggcorrplot)
```

```
## Loading required package: ggplot2
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(mltools)
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 dplyr::between()    masks data.table::between()
## x dplyr::compute()    masks neuralnet::compute()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks data.table::first()
## x dplyr::lag()        masks stats::lag()
## x dplyr::last()       masks data.table::last()
## x yardstick::mcc()    masks mltools::mcc()
## x dials::prune()      masks rpart::prune()
## x tidyr::replace_na() masks mltools::replace_na()
## x yardstick::rmse()   masks mltools::rmse()
## x recipes::step()     masks stats::step()
## x purrr::transpose()  masks data.table::transpose()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v lubridate 1.9.2     v stringr   1.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x readr::col_factor()  masks scales::col_factor()
## x dplyr::compute()     masks neuralnet::compute()
## x purrr::discard()     masks scales::discard()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x stringr::fixed()     masks recipes::fixed()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x tidyr::replace_na()  masks mltools::replace_na()
## x lubridate::second()  masks data.table::second()
## x readr::spec()        masks yardstick::spec()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(pan)
library(mice)
```

```
##
## Attaching package: 'mice'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(multiUS)
```

```
##
## Attaching package: 'multiUS'
##
```

```
## The following object is masked from 'package:recipes':
##
##     discretize
```

```r
library(dplyr)
library(recipes)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(neuralnet)
library(tree)
library(stringr)
library(ggplot2)
library(rpart.plot)
library(yardstick)
library(C50)
library(tune)
library(Hmisc)
```

```
##
## Attaching package: 'Hmisc'
##
## The following object is masked from 'package:parsnip':
##
##     translate
##
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
##
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(finetune)
library(baguette)
library(ggrepel)
library(ggfortify)
```

```
## Registered S3 method overwritten by 'ggfortify':
##   method          from
##   autoplot.glmnet parsnip
```

```r
library(boot)
```

```r
r_housing <- read_csv("r-housing.csv")
```

```
## New names:
## Rows: 71 Columns: 17
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (3): Listing Date, Sale Date, Close Date dbl (14): List Price, Sale Price, Age,
## Sq Ft Total, Lot Size...5, DOM, Zip C...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘Lot Size‘ -> ‘Lot Size...5‘
## * ‘Lot Size‘ -> ‘Lot Size...13‘
```

```r
#create observation
input = c(2145000,  NA, 52, 1647,   39, 1996, 2306, 3, 3,   2,  1)

#remove unimportant variables and add new input observation to data
data_set = r_housing[,-c(5:8,10,11)]

data_set_final = rbind(data_set,input)
```

```r
standard_dev =sd(r_housing$`Sale Price`)

avg = mean(r_housing$`Sale Price`)
```

```r
#create train and predict data frames
housing.train = as.data.frame(data_set_final[1:(nrow(data_set_final) -1),])

housing.predict = as.data.frame(data_set_final[nrow(data_set_final),])

#rename column 7
colnames(housing.train)[7] ="Lot Size"
colnames(housing.predict)[7] ="Lot Size"
```

```r
  avg = 0
  std = 0

  for(j in 1:(ncol(housing.train))){
    avg[j] = mean(housing.train[,j])
    std[j] = sd(housing.train[,j])


    for(i in 1:nrow(housing.train)){
      housing.train[i,j] = (housing.train[i,j] - avg[j])/std[j]
    }
  }

  housing.predict = (housing.predict - avg)/std
```

```r
bag_tree_rpart_spec <-
  rand_forest(mtry = 10, min_n = 1) %>%
  set_engine('randomForest') %>%
  set_mode('regression')


boost_tree_xgboost_spec <-
  boost_tree(tree_depth = tune(), trees = tune(), learn_rate = tune(), min_n = tune(), loss_reduction =
  set_engine('xgboost') %>%
  set_mode('regression')

decision_tree_rpart_spec <-
  decision_tree(tree_depth = tune(), min_n = tune(), cost_complexity = tune()) %>%
  set_engine('rpart') %>%
  set_mode('regression')

linear_reg_lm_spec <-
  linear_reg() %>%
  set_engine('lm')

Lasso_reg_lm_spec <-
  linear_reg(penalty = tune(),mixture = 1) %>%
  set_engine('glmnet')

Ridge_reg_lm_spec <-
  linear_reg(penalty = tune(),mixture = 0) %>%
  set_engine('glmnet')

Elastic_reg_lm_spec <-
  linear_reg(penalty = tune(),mixture = 0.5) %>%
  set_engine('glmnet')

mlp_nnet_spec <-
  mlp(hidden_units = tune(), penalty = tune(), epochs = tune()) %>%
  set_engine('nnet') %>%
  set_mode('regression')


nearest_neighbor_kknn_spec <-
  nearest_neighbor(neighbors = tune(), weight_func = tune(), dist_power = tune()) %>%
  set_engine('kknn') %>%
  set_mode('regression')

rand_forest_randomForest_spec <-
  rand_forest(mtry = 3, min_n = tune()) %>%
  set_engine('randomForest') %>%
  set_mode('regression')
```

```
set.seed(1)
k_fold = vfold_cv(housing.train, repeats = 5)
```

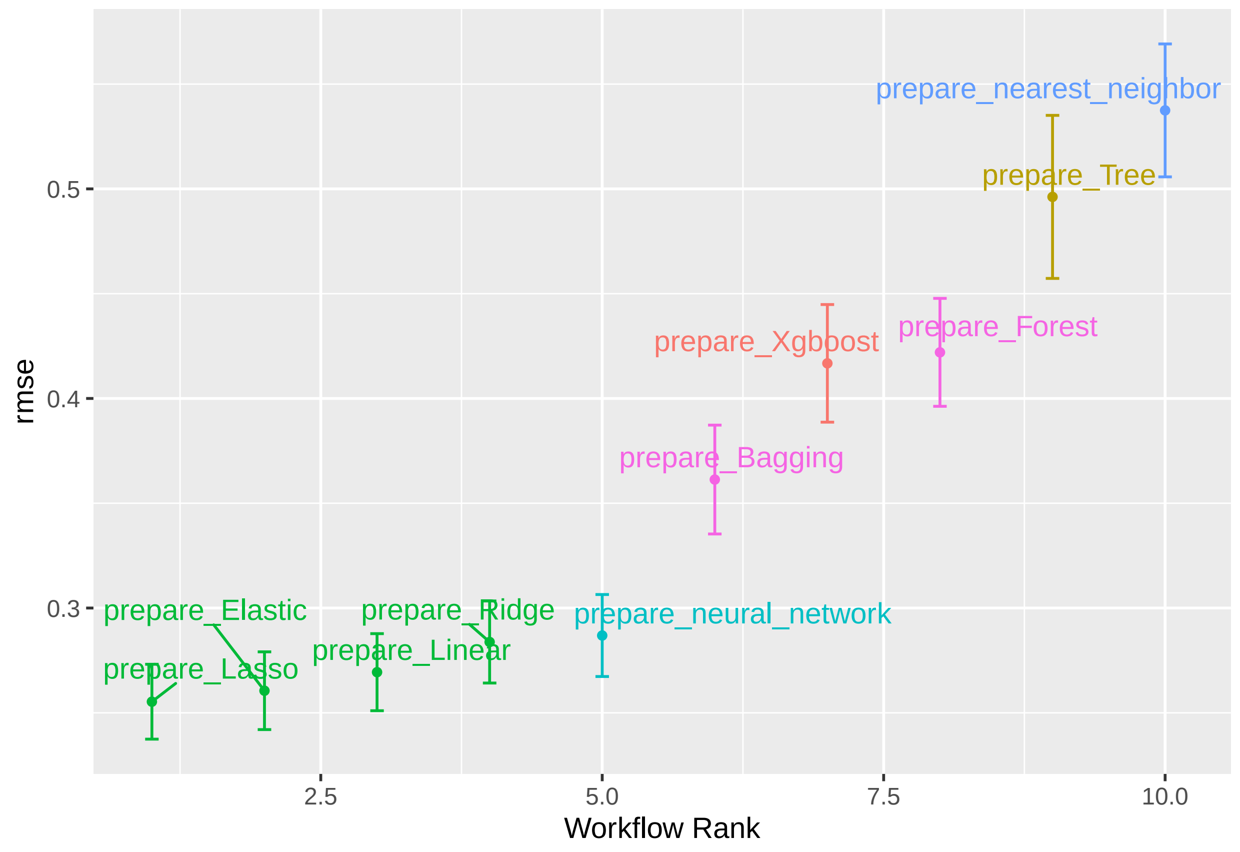
```
set.seed(1)


prep = recipe(`Sale Price` ~ ., data = housing.train)
```

```r
all_workflows <- workflow_set(preproc = list(prepare = prep),
                              models = list(Bagging = bag_tree_rpart_spec, Xgboost = boost_tree_xgboost_spe
```

```r
grid_ctrl <-
   control_sim_anneal(
      save_pred = TRUE,
      parallel_over = "everything",
      save_workflow = TRUE

   )


grid_results <-
   all_workflows %>%
   workflow_map(
      seed = 1,
      fn = "tune_sim_anneal",
      resamples = k_fold,
        iter = 15,
     metrics = metric_set(rmse),
      control = grid_ctrl
   )
```

```
## Optimizing rmse

## Initial best: 0.62798

## 1 <3 new best           rmse=0.60548 (+/-0.02207)

## 2 ( ) accept suboptimal  rmse=0.63341 (+/-0.02381)

## 3 + better suboptimal  rmse=0.62287 (+/-0.02506)

## 4 ( ) accept suboptimal  rmse=0.63515 (+/-0.02431)

## 5 ( ) accept suboptimal  rmse=0.6435 (+/-0.02592)

## 6 - discard suboptimal rmse=0.69281 (+/-0.02932)

## 7 + better suboptimal  rmse=0.60854 (+/-0.02337)

## 8 <3 new best           rmse=0.49725 (+/-0.02018)

## 9 <3 new best           rmse=0.48358 (+/-0.01931)

## 10 - discard suboptimal rmse=0.58172 (+/-0.02667)

## 11 <3 new best           rmse=0.43765 (+/-0.01745)

## 12 - discard suboptimal rmse=0.4882 (+/-0.02165)
## 13 <3 new best           rmse=0.41957 (+/-0.01622)

## 14 - discard suboptimal rmse=0.45574 (+/-0.01784)

## 15 <3 new best           rmse=0.41676 (+/-0.01706)

## Optimizing rmse

## Initial best: 0.58252

## 1 <3 new best           rmse=0.57457 (+/-0.02883)

## 2 ( ) accept suboptimal  rmse=0.57542 (+/-0.02869)

## 3 ( ) accept suboptimal  rmse=0.57542 (+/-0.02869)

## 4 ( ) accept suboptimal  rmse=0.57542 (+/-0.02869)

## 5 + better suboptimal  rmse=0.57503 (+/-0.02983)

## 6 <3 new best           rmse=0.52641 (+/-0.02638)

## 7 ( ) accept suboptimal  rmse=0.52641 (+/-0.02638)

## 8 ( ) accept suboptimal  rmse=0.57808 (+/-0.02938)

## 9 ( ) accept suboptimal  rmse=0.57808 (+/-0.02938)

## 10 + better suboptimal  rmse=0.57671 (+/-0.02939)

## 11 + better suboptimal  rmse=0.57542 (+/-0.02869)

## 12 ( ) accept suboptimal  rmse=0.57675 (+/-0.02877)

## 13 + better suboptimal  rmse=0.57457 (+/-0.02883)

## 14 x restart from best  rmse=0.5927 (+/-0.02868)

## 15 <3 new best           rmse=0.49618 (+/-0.02365)

## Optimizing rmse

## Initial best: 0.58410

## 1 - discard suboptimal rmse=0.66895 (+/-0.03041)

## 2 <3 new best           rmse=0.5775 (+/-0.02447)
## 3 ( ) accept suboptimal  rmse=0.57892 (+/-0.02743)

## 4 - discard suboptimal rmse=0.63931 (+/-0.02943)

## 5 <3 new best           rmse=0.50615 (+/-0.02773)

## 6 <3 new best           rmse=0.49884 (+/-0.0212)

## 7 <3 new best           rmse=0.36806 (+/-0.01167)

## 8 <3 new best           rmse=0.29473 (+/-0.01351)

## 9 - discard suboptimal rmse=0.52923 (+/-0.02283)

## 10 - discard suboptimal rmse=0.41652 (+/-0.01607)

## 11 - discard suboptimal rmse=0.30834 (+/-0.009667)

## 12 <3 new best           rmse=0.28689 (+/-0.01189)

## 13 - discard suboptimal rmse=0.33473 (+/-0.01006)

## 14 - discard suboptimal rmse=0.44356 (+/-0.01687)

## 15 ( ) accept suboptimal  rmse=0.2977 (+/-0.01366)

## Optimizing rmse

## Initial best: 0.56017

## 1 ( ) accept suboptimal  rmse=0.57317 (+/-0.02418)

## 2 ( ) accept suboptimal  rmse=0.57741 (+/-0.02404)

## 3 + better suboptimal  rmse=0.56583 (+/-0.02353)

## 4 <3 new best           rmse=0.55278 (+/-0.02362)

## 5 <3 new best           rmse=0.53901 (+/-0.02311)

## 6 <3 new best           rmse=0.52728 (+/-0.02222)

## 7 <3 new best           rmse=0.50506 (+/-0.02148)

## 8 <3 new best           rmse=0.48597 (+/-0.02023)

## 9 <3 new best           rmse=0.46935 (+/-0.0191)
## 10 <3 new best           rmse=0.45547 (+/-0.01832)

## 11 ( ) accept suboptimal  rmse=0.45877 (+/-0.01839)

## 12 <3 new best           rmse=0.44205 (+/-0.0174)

## 13 <3 new best           rmse=0.42693 (+/-0.01583)

## 14 <3 new best           rmse=0.422 (+/-0.01566)

## 15 ( ) accept suboptimal  rmse=0.43071 (+/-0.01644)

## Optimizing rmse

## Initial best: 0.53745

## 1 ( ) accept suboptimal  rmse=0.5499 (+/-0.02036)

## 2 ( ) accept suboptimal  rmse=0.59141 (+/-0.02429)

## 3 + better suboptimal  rmse=0.58617 (+/-0.02392)

## 4 ( ) accept suboptimal  rmse=0.59581 (+/-0.02378)

## 5 ( ) accept suboptimal  rmse=0.6371 (+/-0.02565)

## 6 - discard suboptimal rmse=0.68652 (+/-0.02483)

## 7 ( ) accept suboptimal  rmse=0.6791 (+/-0.02653)

## 8 x restart from best  rmse=0.65696 (+/-0.02639)

## 9 ( ) accept suboptimal  rmse=0.55362 (+/-0.02236)

## 10 + better suboptimal  rmse=0.55035 (+/-0.02209)

## 11 ( ) accept suboptimal  rmse=0.55867 (+/-0.02288)

## 12 ( ) accept suboptimal  rmse=0.56218 (+/-0.0228)

## 13 ( ) accept suboptimal  rmse=0.56696 (+/-0.02316)

## 14 ( ) accept suboptimal  rmse=0.58144 (+/-0.02424)

## 15 + better suboptimal  rmse=0.57779 (+/-0.0239)

## Optimizing rmse
## Initial best: 0.27519
## 1 <3 new best           rmse=0.26361 (+/-0.01142)
## 2 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 3 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 4 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 5 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 6 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 7 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 8 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## 9 x restart from best  rmse=0.26457 (+/-0.01146)
## 10 <3 new best           rmse=0.26125 (+/-0.01139)
## 11 - discard suboptimal rmse=0.26457 (+/-0.01146)
## 12 - discard suboptimal rmse=0.34795 (+/-0.01161)
## 13 <3 new best           rmse=0.25531 (+/-0.01086)
## 14 - discard suboptimal rmse=0.43056 (+/-0.01672)
## 15 ( ) accept suboptimal  rmse=0.26457 (+/-0.01146)
## Optimizing rmse
## Initial best: 0.28379
## 1 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)
## 2 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)
## 3 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)
## 4 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)
## 5 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)
## 6 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)
## 7 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)

## 8 x restart from best  rmse=0.28379 (+/-0.0119)

## 9 - discard suboptimal rmse=0.3615 (+/-0.01266)

## 10 ( ) accept suboptimal  rmse=0.29655 (+/-0.01219)

## 11 + better suboptimal  rmse=0.28379 (+/-0.0119)

## 12 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)

## 13 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)

## 14 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)

## 15 ( ) accept suboptimal  rmse=0.28379 (+/-0.0119)

## Optimizing rmse

## Initial best: 0.26218

## 1 ( ) accept suboptimal  rmse=0.26433 (+/-0.01145)

## 2 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 3 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 4 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 5 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 6 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 7 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 8 x restart from best  rmse=0.2646 (+/-0.01146)

## 9 - discard suboptimal rmse=0.40184 (+/-0.01426)

## 10 - discard suboptimal rmse=0.29276 (+/-0.01051)

## 11 <3 new best           rmse=0.26055 (+/-0.01128)

## 12 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 13 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 14 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## 15 ( ) accept suboptimal  rmse=0.2646 (+/-0.01146)

## > A | warning: prediction from a rank-deficient fit may be misleading

## There were issues with some computations   A: x1There were issues with some computations   A: x4Ther
```

```r
autoplot(
   grid_results,
   rank_metric = "rmse",  # <- how to order models
   metric = "rmse",       # <- which metric to visualize
   select_best = TRUE     # <- one point per workflow
) +geom_text_repel(aes(label = wflow_id), nudge_x = 1/8, nudge_y = 1/100) +
  theme(legend.position = "none")
```



```r
best_results <-
   grid_results %>%
   extract_workflow_set_result("prepare_Lasso") %>%
   select_best(metric = "rmse")
best_results
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
## 1  0.0151 Iter13
```

```r
lasso_spec <-
  linear_reg(penalty = best_results$penalty,mixture = 1) %>%
  set_engine('glmnet')
```

```r
wf <- workflow(preproc = prep)

 lasso_fit <- wf %>%
  add_model(lasso_spec) %>%
  fit(data = housing.train)
```

```r
#create prediction
prediction = as.numeric(predict(lasso_fit,housing.predict[,-2] ) *std[2] + avg[2])

prediction
```

```
## [1] 2123355
```