housingproject_redux

2024-05-06

```r
#housing project redux
#set directory
setwd("/Users/seanmilligan/Desktop/EC424/Homework/housingproject_redux")

#loading packages
pacman::p_load(tidyverse, tidymodels, skimr, caret, leaps, magrittr, janitor, glmnet, zoo)
```

```r
#loading data
training_df = read.csv('train.csv')
test_df = read.csv('test.csv')

#create age variable (year sold - year built)
house_df = training_df %>% transmute(
  id = Id,
  sale_price = log(SalePrice),
  age = YrSold - YearBuilt,
  remod = YrSold - YearRemodAdd,
  area = GrLivArea,
  lot_area = LotArea,
  cond = OverallCond,
  veneer = MasVnrArea,
  bsmt_sf = TotalBsmtSF,
  bath = FullBath,
  bed_abv = BedroomAbvGr,
  kit_abv = KitchenAbvGr,
  rms_abv = TotRmsAbvGrd,
  fire = Fireplaces,
  grg_age = YrSold - GarageYrBlt,
  wd_dck = WoodDeckSF,
  cl_prch = EnclosedPorch,
  pool = PoolArea
)
```

```r
#5-fold cross validation
#set seed
set.seed(1234)
#5-fold CV on training dataset
house_cv = house_df %>% vfold_cv(v = 5)
#view CV
house_cv %>% tidy()
```

```
## # A tibble: 7,300 x 3
##      Row Data     Fold
```

```
##    <int> <chr>    <chr>
##  1     1 Analysis Fold1
##  2     1 Analysis Fold2
##  3     1 Analysis Fold4
##  4     1 Analysis Fold5
##  5     2 Analysis Fold1
##  6     2 Analysis Fold2
##  7     2 Analysis Fold4
##  8     2 Analysis Fold5
##  9     3 Analysis Fold1
## 10     3 Analysis Fold2
## # i 7,290 more rows
```

```r
#define a recipe_all is
recipe_all = recipe(sale_price ~ ., data = house_df)

#putting it together
house_recipe = recipe_all %>%
  #mean imputation for numeric predictors
  step_impute_mean(all_predictors() & all_numeric()) %>%
  #KNN imputation for categorical predictors
  step_impute_knn(all_predictors() & all_nominal(), neighbors = 5 ) %>%
  #create dummies for categorical variables
  step_dummy(all_predictors() & all_nominal())

#putting it together (again for Forward selection)
house_clean = recipe_all %>%
  #mean imputation for numeric predictors
  step_impute_mean(all_predictors() & all_numeric()) %>%
  #KNN imputation for categorical predictors
  step_impute_knn(all_predictors() & all_nominal(), neighbors = 5 ) %>%
  #create dummies for categorical variables
  step_dummy(all_predictors() & all_nominal()) %>%
  #prep and juicing!
  prep() %>% juice()
```

```r
#defining model (model type and desired engine)
model_lm =
  linear_reg() %>%
  set_mode('regression') %>%
  set_engine('lm')

#estimating linear regression
#fitting simple linear regression using tidymodels
lm_workflow =
  workflow() %>%
  add_model(model_lm) %>%
  add_recipe(house_recipe)

#fit workflow to data
lm_fit =
  lm_workflow %>%
  fit(data = house_df)
```

```r
#view model summary
lm_fit %>% extract_fit_parsnip() %>% tidy()
```

```
## # A tibble: 18 x 5
##    term          estimate   std.error statistic  p.value
##    <chr>            <dbl>       <dbl>     <dbl>    <dbl>
##  1 (Intercept) 11.2        0.0435        258.    0
##  2 id          -0.00000482 0.0000108      -0.445 6.57e- 1
##  3 age         -0.00500    0.000298      -16.8   1.02e-57
##  4 remod       -0.00180    0.000335       -5.37  9.25e- 8
##  5 area         0.000272   0.0000196      13.9   2.98e-41
##  6 lot_area     0.00000191 0.000000492     3.88  1.10e- 4
##  7 cond         0.0629     0.00501        12.6   2.28e-34
##  8 veneer       0.0000635  0.0000290       2.19  2.89e- 2
##  9 bsmt_sf      0.000172   0.0000131      13.1   3.35e-37
## 10 bath         0.0339     0.0126          2.69  7.31e- 3
## 11 bed_abv     -0.0299     0.00810        -3.69  2.33e- 4
## 12 kit_abv     -0.144      0.0232         -6.23  6.00e-10
## 13 rms_abv      0.0286     0.00605         4.73  2.48e- 6
## 14 fire         0.0761     0.00852         8.92  1.36e-18
## 15 grg_age     -0.0000751  0.000331       -0.227 8.21e- 1
## 16 wd_dck       0.000118   0.0000390       3.02  2.58e- 3
## 17 cl_prch      0.000285   0.0000821       3.47  5.40e- 4
## 18 pool        -0.000474   0.000117       -4.06  5.12e- 5
```

```r
#fitting linear regression w/ 5-fold CV
fit_lm_cv =
  workflow() %>%
  add_model(model_lm) %>%
  add_recipe(house_recipe) %>%
  fit_resamples(house_cv)
#checking performance
fit_lm_cv %>% collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.179     5  0.0164 Preprocessor1_Model1
## 2 rsq     standard   0.799     5  0.0386 Preprocessor1_Model1
```

```r
#checking performance within each fold
fit_lm_cv %>% collect_metrics(summarize = F)
```

```
## # A tibble: 10 x 5
##    id    .metric .estimator .estimate .config
##    <chr> <chr>   <chr>          <dbl> <chr>
##  1 Fold1 rmse    standard       0.159 Preprocessor1_Model1
##  2 Fold1 rsq     standard       0.844 Preprocessor1_Model1
##  3 Fold2 rmse    standard       0.147 Preprocessor1_Model1
##  4 Fold2 rsq     standard       0.856 Preprocessor1_Model1
##  5 Fold3 rmse    standard       0.200 Preprocessor1_Model1
##  6 Fold3 rsq     standard       0.771 Preprocessor1_Model1
```

```
##  7 Fold4 rmse    standard       0.234 Preprocessor1_Model1
##  8 Fold4 rsq     standard       0.659 Preprocessor1_Model1
##  9 Fold5 rmse    standard       0.156 Preprocessor1_Model1
## 10 Fold5 rsq     standard       0.863 Preprocessor1_Model1
```

```
#forward selection for available variables
train_forward1 = train(
  y = house_clean[["sale_price"]],
  x = house_clean %>% dplyr::select(-sale_price),
  trControl = trainControl(method = "cv", number = 5),
  method = "leapForward",
  tuneGrid = expand.grid(nvmax = 1:18)
)
```

```
## Warning: Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
```

```
train_forward1$results
```

```
##    nvmax      RMSE  Rsquared       MAE     RMSESD RsquaredSD       MAESD
## 1      1 0.2842110 0.4940369 0.2090159 0.02306191 0.02998926 0.01160796
## 2      2 0.2175185 0.7053151 0.1556852 0.03166460 0.05487518 0.01582137
## 3      3 0.2077624 0.7322241 0.1418299 0.04111959 0.07660834 0.01195041
## 4      4 0.1875634 0.7813858 0.1233449 0.04395227 0.08041279 0.01282676
## 5      5 0.1814070 0.7947432 0.1202967 0.04273816 0.07580137 0.01250764
## 6      6 0.1783330 0.8017521 0.1193154 0.04223832 0.07343004 0.01236017
## 7      7 0.1810199 0.7962162 0.1192956 0.03965631 0.06787574 0.01000858
## 8      8 0.1804600 0.7973894 0.1182092 0.04066592 0.06951185 0.01042072
## 9      9 0.1806877 0.7970194 0.1178035 0.04146419 0.07121055 0.01033003
## 10    10 0.1802968 0.7979378 0.1179301 0.04017025 0.06864131 0.01011049
## 11    11 0.1808203 0.7971098 0.1180978 0.04017259 0.06849221 0.01013907
## 12    12 0.1795220 0.7998376 0.1175321 0.04030441 0.06890698 0.01044305
## 13    13 0.1785452 0.8017009 0.1172972 0.04046780 0.06900410 0.01105157
## 14    14 0.1778665 0.8031226 0.1166371 0.04134069 0.07068748 0.01136296
## 15    15 0.1769187 0.8053560 0.1156569 0.04136260 0.07068315 0.01114497
## 16    16 0.1773657 0.8043758 0.1158068 0.04174652 0.07159018 0.01105627
## 17    17 0.1773501 0.8044076 0.1157926 0.04174004 0.07158181 0.01107723
## 18    18 0.1773501 0.8044076 0.1157926 0.04174004 0.07158181 0.01107723
```

```
#model with all variables has lowest RMSE ^^^
```

Our model containing all variables possesses the lowest Residual Mean Squared Error.

```
#using lasso regression w/tidy models with CV
#standardizing data for use
house_recipe_lasso = house_clean %>% recipe(sale_price ~ .) %>%
  update_role(id, new_role = 'id_variable') %>%
  step_normalize(all_predictors() & all_numeric()) %>%
```

```r
  step_dummy(all_predictors() & all_nominal()) %>%
  step_rename_at(everything(), fn = str_to_lower)
#time to juice it up
house_recipe_lasso_clean = house_recipe %>% prep() %>% juice()
```

```r
#using lasso and ridge w/5-fold cross validation for penalty on lasso and regression
set.seed(12345)
ctrl_cv = trainControl(method = "cv", number = 5)

#define range of lambdas (glmnet wants decreasing range)
lambdas = 10^seq(from = 5, to = -2, length = 100)

#defining model
lasso_est = linear_reg(penalty = tune(), mixture = 1) %>% set_engine('glmnet')

#defining lasso workflow
workflow_lasso = workflow() %>%
  add_model(lasso_est) %>% add_recipe(house_recipe_lasso)
#CV w/range of lambdas
cv_lasso =
  workflow_lasso %>%
  tune_grid(
    resamples = vfold_cv(house_clean, v = 5),
    grid = data.frame(penalty = lambdas),
    metrics = metric_set(rmse)
  )
#show best models
cv_lasso %>% show_best()
```

```
## Warning in show_best(.): No value of ‘metric‘ was given; "rmse" will be used.

## # A tibble: 5 x 7
##   penalty .metric .estimator  mean     n std_err .config
##     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1  0.01   rmse    standard   0.181     5  0.0146 Preprocessor1_Model001
## 2  0.0118 rmse    standard   0.181     5  0.0143 Preprocessor1_Model002
## 3  0.0138 rmse    standard   0.182     5  0.0139 Preprocessor1_Model003
## 4  0.0163 rmse    standard   0.183     5  0.0134 Preprocessor1_Model004
## 5  0.0192 rmse    standard   0.185     5  0.0128 Preprocessor1_Model005
```

```r
#finding best lambda
cv_lasso$.metrics
```

```
## [[1]]
## # A tibble: 100 x 5
##    penalty .metric .estimator .estimate .config
##      <dbl> <chr>   <chr>          <dbl> <chr>
##  1  0.01   rmse    standard       0.238 Preprocessor1_Model001
##  2  0.0118 rmse    standard       0.237 Preprocessor1_Model002
##  3  0.0138 rmse    standard       0.237 Preprocessor1_Model003
##  4  0.0163 rmse    standard       0.236 Preprocessor1_Model004
##  5  0.0192 rmse    standard       0.235 Preprocessor1_Model005
```

```
##  6  0.0226 rmse    standard       0.234 Preprocessor1_Model006
##  7  0.0266 rmse    standard       0.233 Preprocessor1_Model007
##  8  0.0313 rmse    standard       0.233 Preprocessor1_Model008
##  9  0.0368 rmse    standard       0.233 Preprocessor1_Model009
## 10  0.0433 rmse    standard       0.234 Preprocessor1_Model010
## # i 90 more rows
##
## [[2]]
## # A tibble: 100 x 5
##    penalty .metric .estimator .estimate .config
##      <dbl> <chr>   <chr>          <dbl> <chr>
##  1  0.01   rmse    standard       0.159 Preprocessor1_Model001
##  2  0.0118 rmse    standard       0.159 Preprocessor1_Model002
##  3  0.0138 rmse    standard       0.161 Preprocessor1_Model003
##  4  0.0163 rmse    standard       0.162 Preprocessor1_Model004
##  5  0.0192 rmse    standard       0.164 Preprocessor1_Model005
##  6  0.0226 rmse    standard       0.166 Preprocessor1_Model006
##  7  0.0266 rmse    standard       0.170 Preprocessor1_Model007
##  8  0.0313 rmse    standard       0.174 Preprocessor1_Model008
##  9  0.0368 rmse    standard       0.180 Preprocessor1_Model009
## 10  0.0433 rmse    standard       0.186 Preprocessor1_Model010
## # i 90 more rows
##
## [[3]]
## # A tibble: 100 x 5
##    penalty .metric .estimator .estimate .config
##      <dbl> <chr>   <chr>          <dbl> <chr>
##  1  0.01   rmse    standard       0.174 Preprocessor1_Model001
##  2  0.0118 rmse    standard       0.175 Preprocessor1_Model002
##  3  0.0138 rmse    standard       0.176 Preprocessor1_Model003
##  4  0.0163 rmse    standard       0.177 Preprocessor1_Model004
##  5  0.0192 rmse    standard       0.179 Preprocessor1_Model005
##  6  0.0226 rmse    standard       0.181 Preprocessor1_Model006
##  7  0.0266 rmse    standard       0.184 Preprocessor1_Model007
##  8  0.0313 rmse    standard       0.188 Preprocessor1_Model008
##  9  0.0368 rmse    standard       0.193 Preprocessor1_Model009
## 10  0.0433 rmse    standard       0.197 Preprocessor1_Model010
## # i 90 more rows
##
## [[4]]
## # A tibble: 100 x 5
##    penalty .metric .estimator .estimate .config
##      <dbl> <chr>   <chr>          <dbl> <chr>
##  1  0.01   rmse    standard       0.160 Preprocessor1_Model001
##  2  0.0118 rmse    standard       0.162 Preprocessor1_Model002
##  3  0.0138 rmse    standard       0.163 Preprocessor1_Model003
##  4  0.0163 rmse    standard       0.165 Preprocessor1_Model004
##  5  0.0192 rmse    standard       0.168 Preprocessor1_Model005
##  6  0.0226 rmse    standard       0.171 Preprocessor1_Model006
##  7  0.0266 rmse    standard       0.175 Preprocessor1_Model007
##  8  0.0313 rmse    standard       0.181 Preprocessor1_Model008
##  9  0.0368 rmse    standard       0.188 Preprocessor1_Model009
## 10  0.0433 rmse    standard       0.193 Preprocessor1_Model010
## # i 90 more rows
```

```
##
## [[5]]
## # A tibble: 100 x 5
##    penalty .metric .estimator .estimate .config
##      <dbl> <chr>   <chr>          <dbl> <chr>
##  1  0.01   rmse    standard       0.173 Preprocessor1_Model001
##  2  0.0118 rmse    standard       0.174 Preprocessor1_Model002
##  3  0.0138 rmse    standard       0.176 Preprocessor1_Model003
##  4  0.0163 rmse    standard       0.177 Preprocessor1_Model004
##  5  0.0192 rmse    standard       0.180 Preprocessor1_Model005
##  6  0.0226 rmse    standard       0.182 Preprocessor1_Model006
##  7  0.0266 rmse    standard       0.186 Preprocessor1_Model007
##  8  0.0313 rmse    standard       0.191 Preprocessor1_Model008
##  9  0.0368 rmse    standard       0.196 Preprocessor1_Model009
## 10  0.0433 rmse    standard       0.201 Preprocessor1_Model010
## # i 90 more rows
```

```r
#lowest RMSE ~0.182 @ lambda = 0.0118
#fitting final model
final_lasso = glmnet(
  x = house_clean %>% dplyr::select(-sale_price, -id) %>% as.matrix(),
  y = house_clean$sale_price,
  standardize = T,
  alpha = 1,
  lambda = 0.0118
)
```

```r
#cleaning test data set
#create age variable (year sold - year built)
pred_df = test_df %>% transmute(
    id = Id,
    age = YrSold - YearBuilt,
    remod = YrSold - YearRemodAdd,
    area = GrLivArea,
    lot_area = LotArea,
    cond = OverallCond,
    veneer = MasVnrArea,
    bsmt_sf = TotalBsmtSF,
    bath = FullBath,
    bed_abv = BedroomAbvGr,
    kit_abv = KitchenAbvGr,
    rms_abv = TotRmsAbvGrd,
    fire = Fireplaces,
    grg_age = YrSold - GarageYrBlt,
    wd_dck = WoodDeckSF,
    cl_prch = EnclosedPorch,
    pool = PoolArea
  )

#create function to remove NA values from all columns
rep_NA_func = function(data) {
  for (col in names(data)) {
    data[[col]] = na.aggregate(data[[col]])
  }
```

```r
  return(data)
}

#clean prediction dataframe w/rep_NA function
pred_clean = rep_NA_func(pred_df)
```

```r
#logarithmic prediction
pred_log = predict(
  final_lasso,
  type = "response",
  #our chosen lambda
  s = 0.0118,
  #our data
  newx = pred_clean %>% dplyr::select(-id) %>% as.matrix()
)

#final prediction w/o logarithms
pred_final = exp(pred_log)

#create submission datatset
submit_df = data.frame(
  Id = test_df$Id,
  SalePrice = pred_final
)

#change name of s1 to SalePrice
colnames(submit_df)[colnames(submit_df) == 's1'] = 'SalePrice'

#view first few lines of dataset
head(submit_df)
```

```
##     Id SalePrice
## 1 1461  114853.9
## 2 1462  142464.9
## 3 1463  195685.8
## 4 1464  206797.7
## 5 1465  162418.2
## 6 1466  186738.9
```

```r
#save dataset as CSV
write_csv(x = submit_df, file = 'spm_submit_redux.csv')
```