P8106 - HW1

Tucker Morgan, tlm2152

2/10/2022

```
library(tidyverse)
library(glmnet)
library(caret)

house_trn <- read_csv(file = "./Homework 1/housing_training.csv") %>%
  janitor::clean_names() %>%
  na.omit()

house_tst <- read_csv(file = "./Homework 1/housing_test.csv") %>%
  janitor::clean_names() %>%
  na.omit()
```

In this analysis, we will predict the price of a house based on its characteristics. To begin, I have loaded in a training data set, `house_trn`, that consists of 1440 training observations and a test data set, `house_tst`, that contains 959 test observations. Each data set includes 26 variables: `sale_price` along with 25 predictors.

(a) First, we will use the training data set to fit a linear model using least squares. We will use a cross-validation technique on the training data.

```
set.seed(100)
# setting up ten-fold CV, repeated five times
ctrl1 <- trainControl(method = "repeatedcv",
                      repeats = 5,
                      number = 10)

lm_fit <- train(sale_price ~ .,
                data = house_trn,
                method = "lm",
                trControl = ctrl1,
                preProcess = "scale")

mean(lm_fit$resample$RMSE) # cross-validation RMSE
```

```
## [1] 23039.1
```

(b) Next, we will fit a lasso model on the training data.

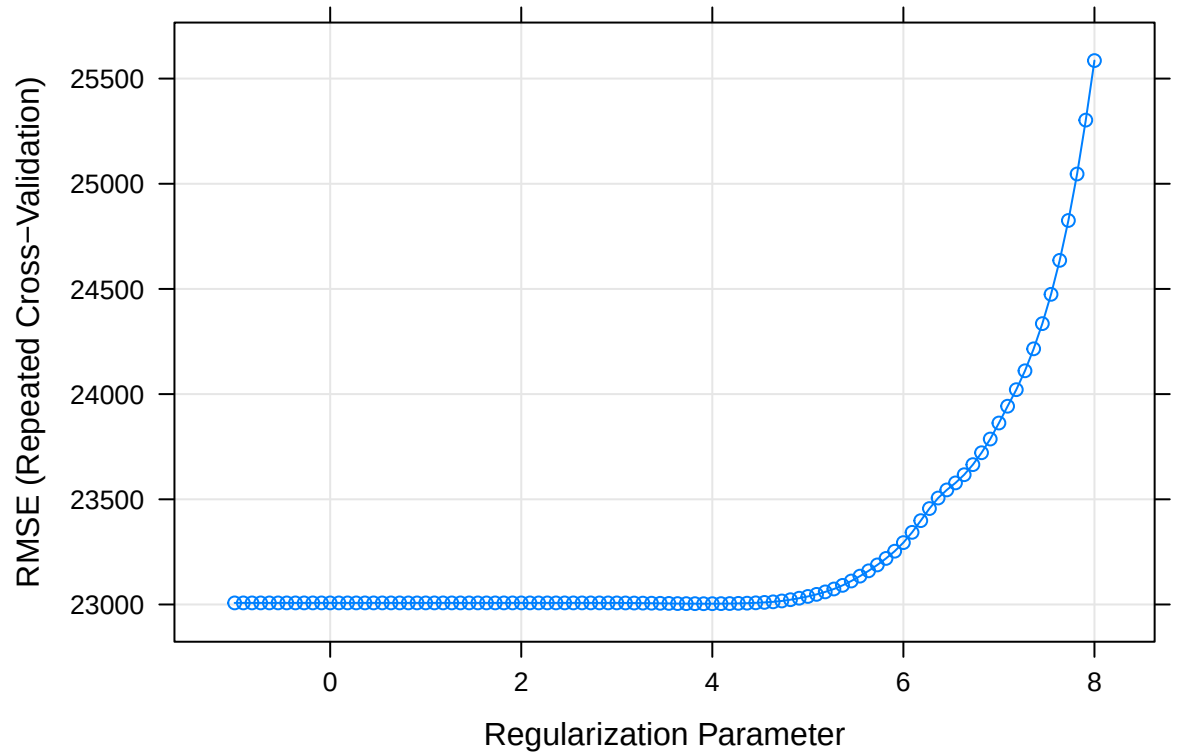
```
set.seed(100)
# creating another control for the 1se rule lasso
ctrl2 <- trainControl(method = "repeatedcv",
```

```
                      repeats = 5,
                      number = 10,
                      selectionFunction = "oneSE")

# creating model matrix of predictors
house_trn_pred <- model.matrix(sale_price ~ ., house_trn)[,-1]
# creating vector of response variable
house_trn_price <- house_trn$sale_price

lasso_fit <- train(x = house_trn_pred,
                   y = house_trn_price,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(8, -1, length = 100))),
                   trControl = ctrl2,
                   preProcess = c("center", "scale"))

plot(lasso_fit, xTrans = log)
```




```
lasso_fit$bestTune # 1se lambda value
```

```
##    alpha   lambda
## 76     1 336.3599
```

```r
# creating model matrix of predictors
house_tst_mat <- model.matrix(sale_price ~ ., house_tst)[,-1]

# creating predictions using lasso model
tst_predict <- predict(lasso_fit, newdata = house_tst_mat)

#calculating RMSE
postResample(pred = tst_predict, obs = house_tst$sale_price) %>%
  knitr::kable()
```

	x
RMSE	2.055714e+04
Rsquared	9.028134e-01
MAE	1.505168e+04

The RMSE noted above is around 20,500 with a tuning parameter (lambda) value of 1, 336.359933810117. Below are the coefficients in the model with this tuning parameter.

```r
coef(lasso_fit$finalModel, lasso_fit$finalModel$lambdaOpt)
```

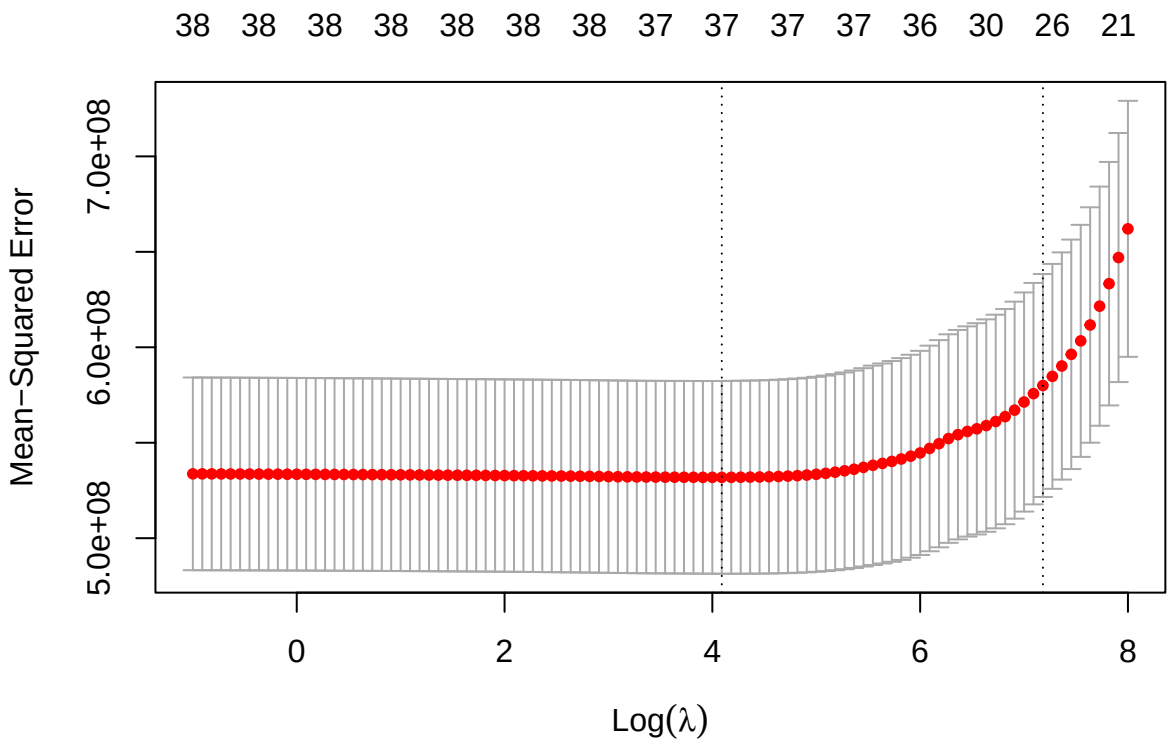
```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                 177568.50208
## gr_liv_area                  29933.71078
## first_flr_sf                   340.07337
## second_flr_sf                        .
## total_bsmt_sf                14953.52327
## low_qual_fin_sf              -1638.11571
## wood_deck_sf                  1384.95299
## open_porch_sf                  812.40144
## bsmt_unf_sf                  -8588.13614
## mas_vnr_area                  2128.83391
## garage_cars                   2600.55063
## garage_area                   1914.98269
## year_built                    9298.06465
## tot_rms_abv_grd              -4158.59861
## full_bath                    -1004.14952
## overall_qualAverage          -1911.25254
## overall_qualBelow_Average    -2888.30522
## overall_qualExcellent        14424.51828
## overall_qualFair             -1125.87374
## overall_qualGood              4605.83789
## overall_qualVery_Excellent   14572.57365
## overall_qualVery_Good        11429.00005
## kitchen_qualFair             -2161.32044
## kitchen_qualGood             -4955.23750
## kitchen_qualTypical          -9349.85198
## fireplaces                    5650.99399
## fireplace_quFair              -816.09155
## fireplace_quGood               637.50900
## fireplace_quNo_Fireplace             .
```

```
## fireplace_quPoor             -342.98853
## fireplace_quTypical         -2070.01681
## exter_qualFair              -1756.91793
## exter_qualGood                -14.90519
## exter_qualTypical           -2315.11851
## lot_frontage                 2900.05089
## lot_area                     4941.05083
## longitude                    -650.69049
## latitude                      747.72297
## misc_val                      252.56840
## year_sold                    -297.12827
```

Below is a similar process using `glmnet` instead of `caret`.

```
# creating model matrix of predictors for glmnet
house_trn_pred <- model.matrix(sale_price ~ ., house_trn)[,-1]
# creating vector of response variable
house_trn_price <- house_trn$sale_price

cv_lasso <- cv.glmnet(x = house_trn_pred,
                      y = house_trn_price,
                      alpha = 1,
                      lambda = exp(seq(8, -1, length = 100)))
plot(cv_lasso)
```



```
cv_lasso$lambda.1se
```

```
## [1] 1315.298
```

```
# creating model matrix of test predictors
house_tst_pred <- model.matrix(sale_price ~ ., house_tst)[,-1]
# creating a vector of test set observations
house_tst_price <- house_tst$sale_price

test_perf <- assess.glmnet(cv_lasso,
                           newx = house_tst_pred,
                           newy = house_tst_price,
                           s = "lambda.1se")
```

The test RMSE for the lasso model is approximately 2.079e+04 with the "1se" lambda value of 1315.3.

```
predict(cv_lasso, s = "lambda.1se", type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                lambda.1se
## (Intercept)                -5.999732e+05
## gr_liv_area                 5.347386e+01
## first_flr_sf                1.416800e+00
## second_flr_sf               .
## total_bsmt_sf               3.656018e+01
## low_qual_fin_sf            -1.639094e+01
## wood_deck_sf                7.324484e+00
## open_porch_sf               4.820064e+00
## bsmt_unf_sf                -1.745468e+01
## mas_vnr_area                1.463122e+01
## garage_cars                 3.306303e+03
## garage_area                 1.200709e+01
## year_built                  3.262336e+02
## tot_rms_abv_grd             .
## full_bath                   .
## overall_qualAverage        -2.297241e+03
## overall_qualBelow_Average  -7.361731e+03
## overall_qualExcellent       8.641791e+04
## overall_qualFair           -3.028855e+03
## overall_qualGood            8.101184e+03
## overall_qualVery_Excellent  1.541066e+05
## overall_qualVery_Good       3.374993e+04
## kitchen_qualFair           -2.424008e+03
## kitchen_qualGood            .
## kitchen_qualTypical        -9.965604e+03
## fireplaces                  7.152757e+03
## fireplace_quFair            .
## fireplace_quGood            3.785450e+03
## fireplace_quNo_Fireplace    .
## fireplace_quPoor            .
## fireplace_quTypical         .
## exter_qualFair             -1.202492e+04
```

```
## exter_qualGood              .
## exter_qualTypical          -5.792050e+03
## lot_frontage                5.082190e+01
## lot_area                    5.054348e-01
## longitude                   .
## latitude                    .
## misc_val                    .
## year_sold                   .
```

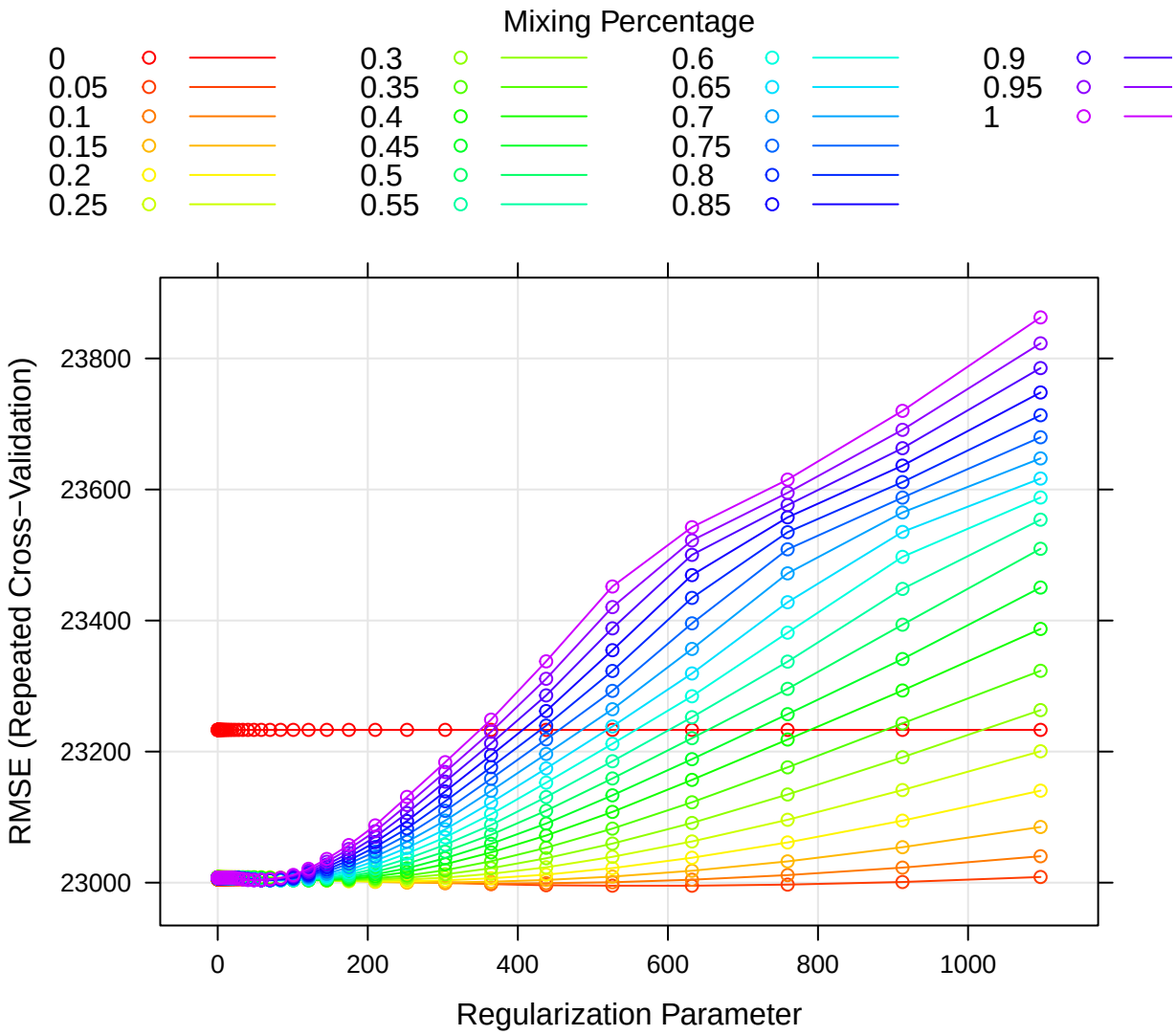
There are 31 coefficients and the intercept in the lasso model when the "1se" rule is used. This lambda value can be seen in the plot above, shown by the right-most dashed line.

(c) Now, we will fit an elastic net model on the training data set.

```
set.seed(100)
enet_fit <- train(x = house_trn_pred,
                  y = house_trn_price,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(7, -2, length = 50))),
                  trControl = ctrl1)
```

```
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(enet_fit, par.settings = myPar)
```



The selected tuning parameters can be seen in the table below, corresponding to the minimum value in the plot above.

	alpha	lambda
97	0.05	632.057