HW 1

Shivalika Chavan

2026-02-15

```
housing_train = read.csv("./data/housing_training.csv") |>
  janitor::clean_names() |>
  mutate(
    overall_qual = factor(overall_qual,
                      levels = c("Very_Excellent",
                                 "Excellent",
                                 "Very_Good",
                                 "Good",
                                 "Above_Average",
                                 "Average",
                                 "Below_Average",
                                 "Fair"),
                      ordered = TRUE),
    kitchen_qual = factor(kitchen_qual,
                      levels = c("Excellent",
                                 "Good",
                                 "Typical",
                                 "Fair"),
                      ordered = TRUE),
    fireplace_qu = factor(fireplace_qu,
                      levels = c("Excellent",
                                 "Good",
                                 "Fair",
                                 "Poor",
                                 "No_Fireplace"
                                 ),
                      ordered = TRUE),
    fireplace_qu = replace_na(fireplace_qu, "No_Fireplace"),
    exter_qual = factor(exter_qual,
                      levels = c("Excellent",
                                 "Good",
                                 "Typical",
                                 "Fair"),
                      ordered = TRUE)
  )

housing_test = read.csv("./data/housing_test.csv") |>
  janitor::clean_names() |>
  mutate(
    overall_qual = factor(overall_qual,
                      levels = c("Very_Excellent",
                                 "Excellent",
```

```
                                "Very_Good",
                                "Good",
                                "Above_Average",
                                "Average",
                                "Below_Average",
                                "Fair"),
                      ordered = TRUE),
    kitchen_qual = factor(kitchen_qual,
                      levels = c("Excellent",
                                "Good",
                                "Typical",
                                "Fair"),
                      ordered = TRUE),
    fireplace_qu = factor(fireplace_qu,
                      levels = c("Excellent",
                                "Good",
                                "Fair",
                                "Poor",
                                "No_Fireplace"
                                ),
                      ordered = TRUE),
    fireplace_qu = replace_na(fireplace_qu, "No_Fireplace"),
    exter_qual = factor(exter_qual,
                      levels = c("Excellent",
                                "Good",
                                "Typical",
                                "Fair"),
                      ordered = TRUE)
  )

y = housing_train |> pull(sale_price)
x = model.matrix(sale_price ~., housing_train)[,-1]

y_test = housing_test$sale_price
x_test = model.matrix(sale_price ~ ., housing_test)[,-1]
```

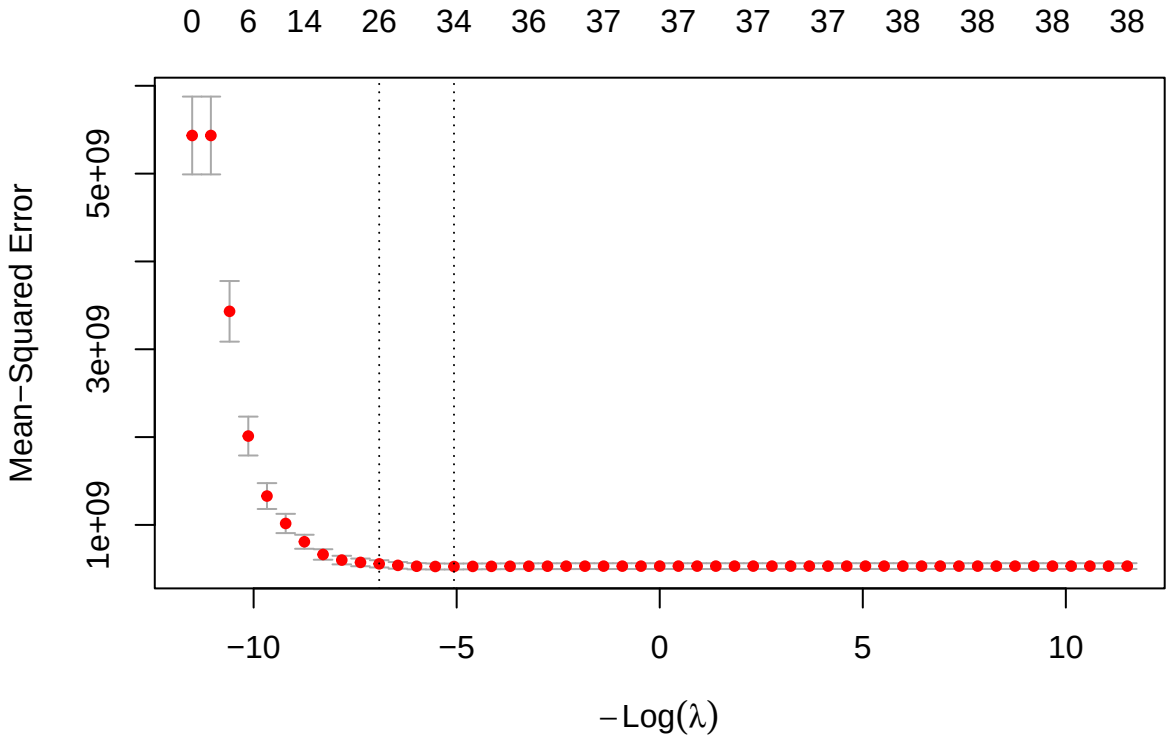
a - LASSO Model

```
set.seed(1234)

lambda = 10^(seq(-5, 5, 0.2)) # lambda grid of lambda values for penalty tuning

# k-fold cross-validation for Lasso (alpha = 1) over the lambda values
lasso_cv = cv.glmnet(x, y,
                     alpha = 1,
                     lambda = lambda)

lambda_min_glmnet = lasso_cv[["lambda.min"]] # minimum mean cross-validated error
lambda_1se_glmnet = lasso_cv[["lambda.1se"]] # largest value of lambda such that error is within 1 stan
CVM_min = lasso_cv |> broom::tidy() |> filter(lambda == lambda_min_glmnet) |> pull(estimate)
CVM_1se = lasso_cv |> broom::tidy() |> filter(lambda == lambda_1se_glmnet) |> pull(estimate)
```

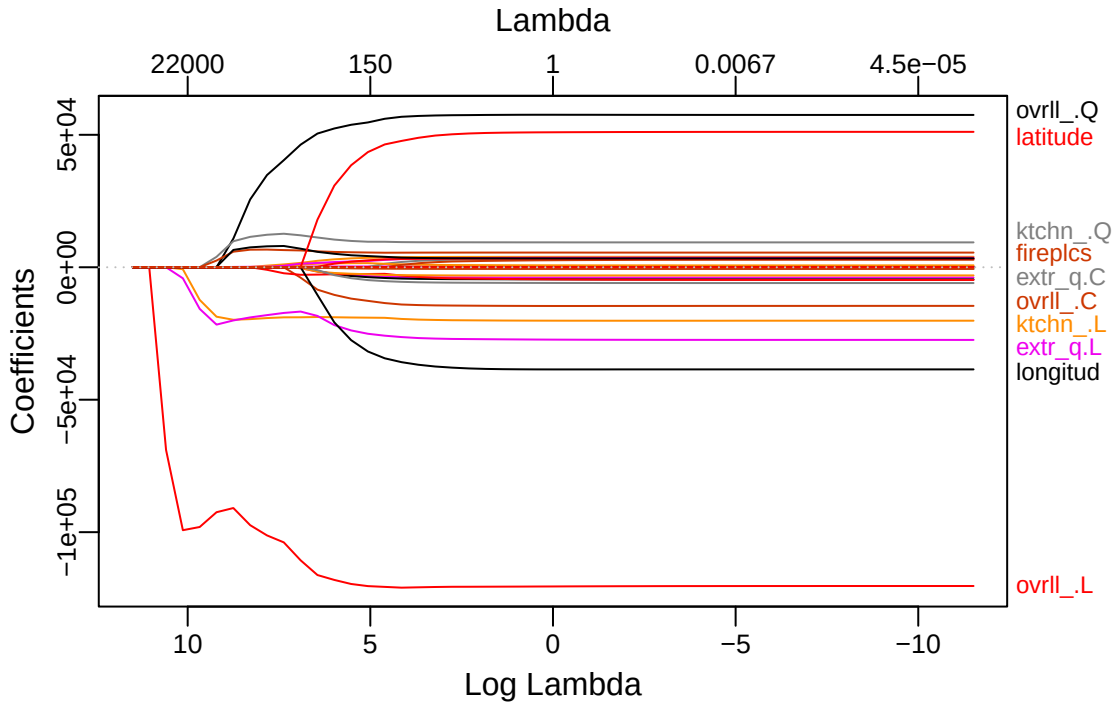
The λ value with the smallest CVM (5.2645935×10^8) is 158.4893192. The λ value with the 1SE CVM

(5.5664618×10^8) is 1000.

```
plot(lasso_cv)
```




```
plot_glmnet(lasso_cv$glmnet.fit)
```



Test Error with $\lambda = 158.49$:

```
y_pred = predict(lasso_cv, newx = x_test, s = lambda_min_glmnet, type = "response")
mspe_lasso <- mean((y_test - y_pred)^2)
```

The test error is 4.3060946×10^8.

```
coef_1se = predict(lasso_cv, type = "coefficients", s = lambda_1se_glmnet)
# Count non-zero coefficients to determine the number of predictors (excluding intercept)
num_predictors_1se = sum(coef_1se != 0) - 1
```

When using λ_{1SE}, there are 26 predictors.

b - Elastic Net Model

```
set.seed(1234)

alpha = seq(0, 1, length = 21) # alpha grid ranging from 0 (Ridge) to 1 (Lasso)
lambda = 10^(seq(-5, 5, 0.2)) # lambda grid of lambda values for penalty tuning

# Iterate through each alpha to perform cross-validation and store results in a tibble
enet_cv_results = tibble(alpha = alpha) |>
  mutate(
    cv_fit = map(alpha, ~cv.glmnet(x, y, alpha = .x, lambda = lambda)), # runs glmnet for each alpha
    min_cvm = map_dbl(cv_fit, ~min(.x$cvm)) # finds min CVM for each model at each alpha
  )

# pull the alpha value with the lowest overall CVM
enet_alpha_min_cvm = enet_cv_results |>
  filter(min_cvm == min(min_cvm)) |>
  pull(alpha)

# pull the corresponding model with that optimal alpha
best_enet_cv_fit = enet_cv_results |>
  filter(alpha == enet_alpha_min_cvm) |>
  pull(cv_fit) |>
  pluck(1)

# pull 1SE lambda for optimal alpha
lambda_1se_enet = best_enet_cv_fit$lambda.1se

# make predictions using optimal alpha and 1SE lambda at that alpha
y_pred = predict(best_enet_cv_fit, newx = x_test, s = "lambda.1se")
mspe_elastic_net = mean((y_test - y_pred)^2)
```

The selected tuning parameters are $\alpha = 0.75$ and $\lambda_{1SE} = 1000$. The model with these parameters has a test error of 4.2461127×10^8.

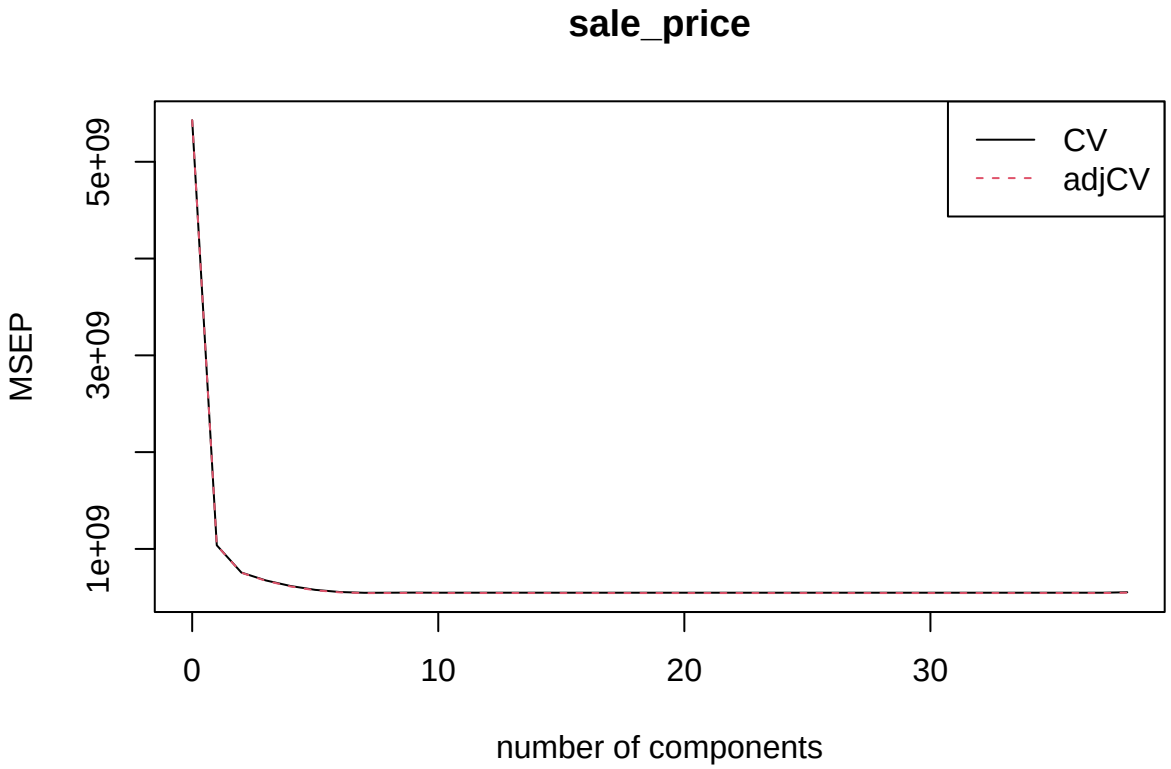
Applying the 1SE rule is a bit more complicated with elastic net, because it has two parameters, α and λ. The 1SE rule, when used in the Lasso model, is easy to implement because a larger value for λ means fewer non-zero coefficients and a more parsimonious model. In elastic net models, this will work when α is fixed at a single value.

c - Partial Least Squares (PLS) Model

```r
set.seed(1234)

pls_mod <- plsr(sale_price ~ .,
                data = housing_train,
                scale = TRUE, # similar scaling importance as PCR
                validation = "CV")
summary(pls_mod)
```

```
## Data:    X dimension: 1440 38
##  Y dimension: 1440 1
## Fit method: kernelpls
## Number of components considered: 38
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           73685    32214    27493    25962    24834    24019    23548
## adjCV        73685    32199    27454    25886    24730    23928    23476
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23381    23400    23419     23395     23401     23403     23404
## adjCV    23311    23326    23342     23319     23324     23325     23326
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        23406     23398     23399     23399     23400     23399     23399
## adjCV     23327     23320     23321     23321     23322     23321     23321
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        23399     23399     23399     23399     23399     23399     23399
## adjCV     23321     23321     23321     23321     23321     23321     23321
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV        23399     23399     23399     23399     23399     23399     23399
## adjCV     23321     23321     23321     23321     23321     23321     23321
##        35 comps  36 comps  37 comps  38 comps
## CV        23399     23399     23399     23499
## adjCV     23321     23321     23321     23400
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             22.75    28.96    34.05    38.43    42.42    46.49    49.60
## sale_price    81.33    86.89    88.79    90.02    90.65    90.90    91.02
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X             52.36    55.49     57.47     59.89     62.33     64.09      65.6
## sale_price    91.05    91.07     91.09     91.09     91.10     91.10      91.1
##             15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## X              66.94      68.8     70.45     71.54     73.23     74.72
## sale_price     91.10      91.1     91.10     91.10     91.10     91.10
##             21 comps  22 comps  23 comps  24 comps  25 comps  26 comps
## X              76.57      77.8     79.91      81.9     83.09     84.17
## sale_price     91.10      91.1     91.10      91.1     91.10     91.10
##             27 comps  28 comps  29 comps  30 comps  31 comps  32 comps
## X              85.82     87.25     88.32     89.51     90.68     91.96
## sale_price     91.10     91.10     91.10     91.10     91.10     91.10
##             33 comps  34 comps  35 comps  36 comps  37 comps  38 comps
## X              93.66     95.02     97.15     97.97     100.0    100.73
## sale_price     91.10     91.10     91.10     91.10      91.1     91.08
```

```r
# plot cross-validated MSEP for PLS
validationplot(pls_mod, val.type = "MSEP", legendpos = "topright")
```




```r
# determine the optimal number of components
cv_mse <- RMSEP(pls_mod)
ncomp_cv <- which.min(cv_mse$val[1,,]) - 1
ncomp_cv
```

```
## 7 comps
##       7
```

```r
# calculate test MSE
predy2_pls <- predict(pls_mod, newdata = housing_test,
                      ncomp = ncomp_cv)

mspe_pls = mean((y_test - predy2_pls)^2)
```

There are 7 components in the partial least squares model, with an MSPE of 4.5063981×10^8.

d - Comparing Models

```r
summary = tibble(
  model = c("LASSO", "Elastic Net", "PLS"),
  mspe = c(mspe_lasso, mspe_elastic_net, mspe_pls)
)

knitr::kable(summary)
```

model	mspe
LASSO	430609458
Elastic Net	424611266
PLS	450639809

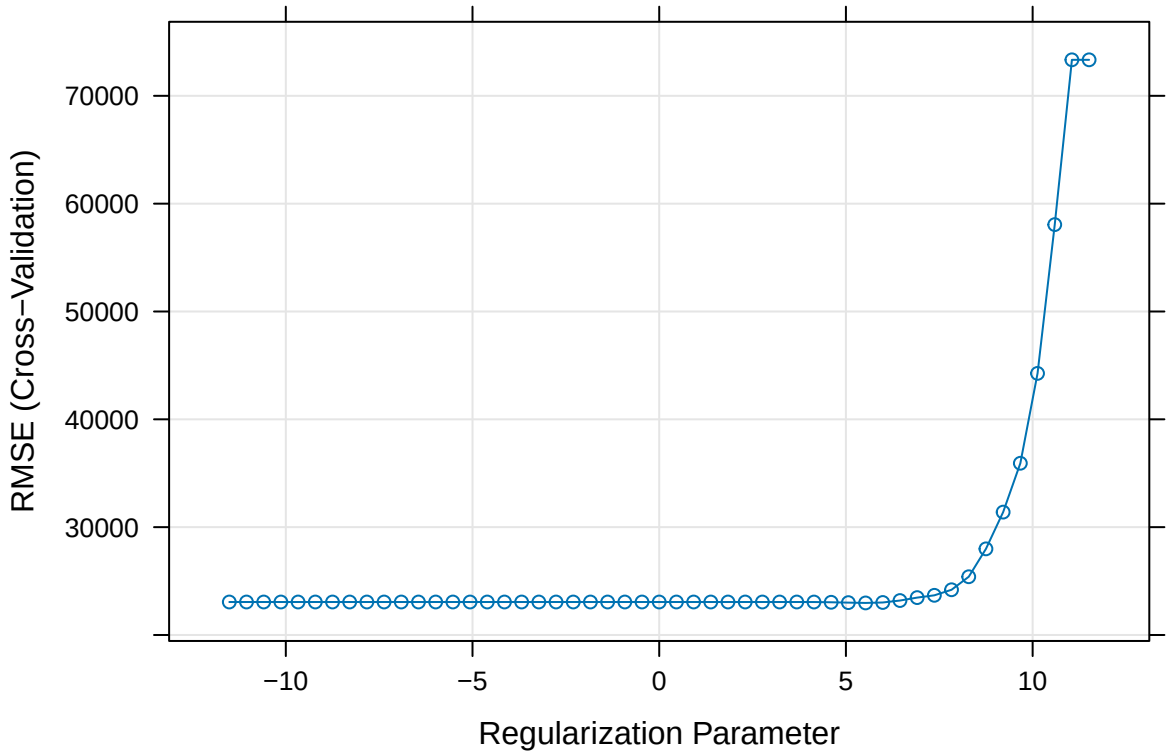
Comparing the mean squared predicted error across the three models, the elastic net model is the best model for making predictions. This means that the model likely benefits from the balance of LASSO and Ridge (λ, α) penalties, as compared to the LASSO model, which only has the λ penalty.

e - Retraining LASSO model using caret instead of glmnet

```
set.seed(1234)

ctrl1 = trainControl(method = "cv", number = 10) #cross validation, in this case 10-fold

lasso_caret = train(sale_price ~ .,
                    data = housing_train,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                                           lambda = lambda),
                    trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
plot(lasso_caret, xTrans = log)
```




```
lambda_min_caret = lasso_caret$bestTune$lambda
```

```r
tibble(
  package = c("caret (min)", "glmnet (min)", "glmnet (1SE)"),
  lambda = c(lambda_min_caret, lambda_min_glmnet, lambda_1se_glmnet)
) |> knitr::kable(digits = 2)
```

package	lambda
caret (min)	251.19
glmnet (min)	158.49
glmnet (1SE)	1000.00

The "best" values for λ differ slightly between `caret` and `cv.glmnet`. However, these two values are much closer to each other than they are to the 1SE value found using `cv.glmnet`. While they will result in slightly different models, their cross-validation errors will be within 1SE of each other. The reason for this discrepancy is the difference in internal functions; specifically, how each package assigns observations to k cross-validation folds, even when the random seed `set.seed(1234)` is kept consistent.