

HW 1

Shivalika Chavan

2026-02-15

```
housing_train = read.csv("./data/housing_training.csv") |> janitor::clean_names()
housing_test = read.csv("./data/housing_test.csv") |> janitor::clean_names()

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

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

a

```
set.seed(1234)

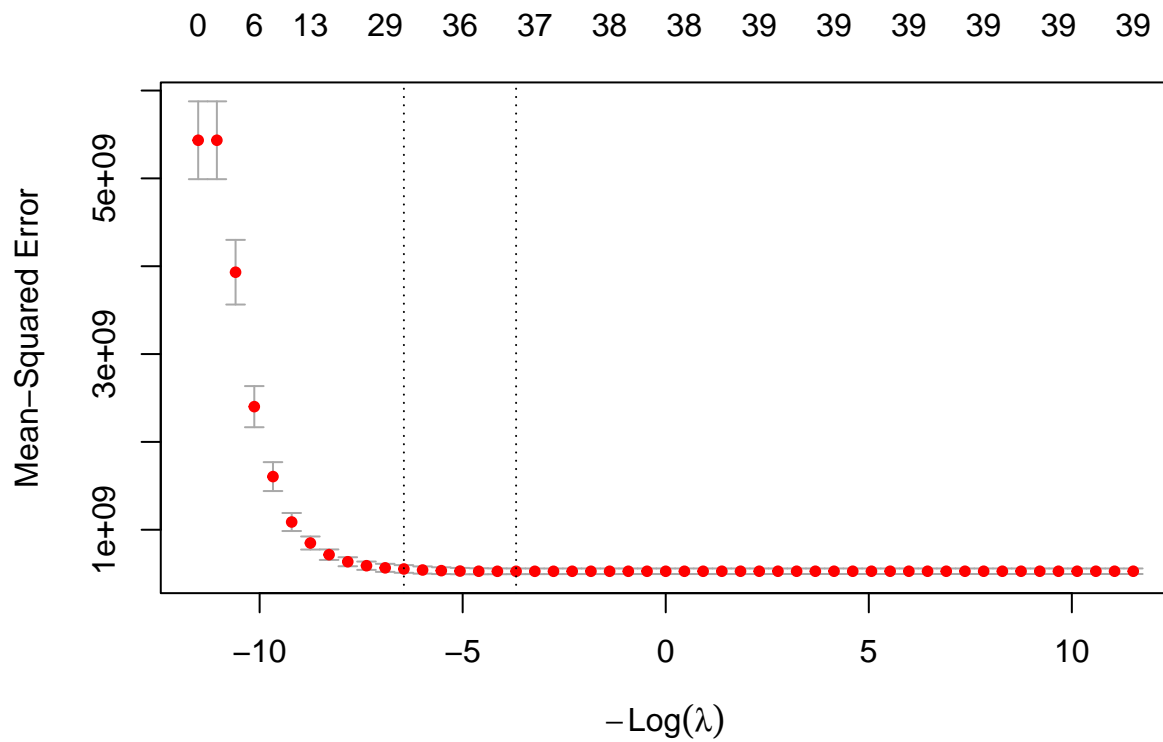
lambda = 10^(seq(-5, 5, 0.2)) # lambda grid of lambda values for penalty tuning

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

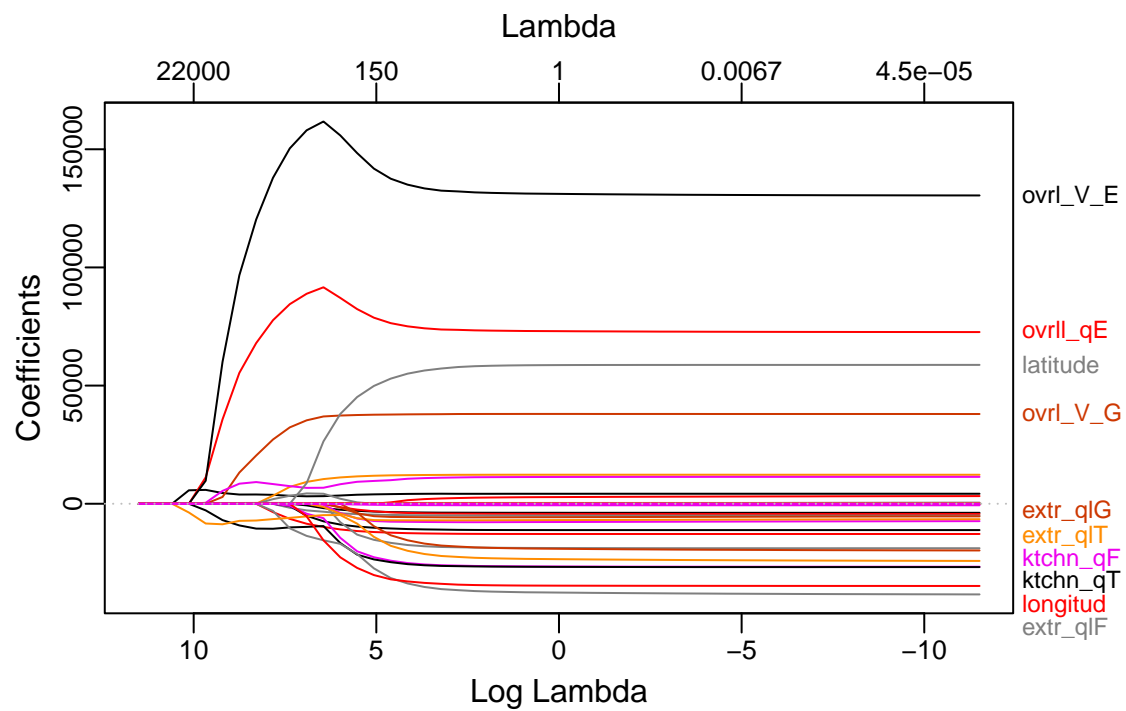
lambda_min = lasso_cv[["lambda.min"]] # minimum mean cross-validated error
lambda_1se = lasso_cv[["lambda.1se"]] # largest value of lambda such that error is within 1 standard error
CVM_min = lasso_cv |> broom::tidy() |> filter(lambda == lambda_min) |> pull(estimate)
CVM_1se = lasso_cv |> broom::tidy() |> filter(lambda == lambda_1se) |> pull(estimate)
```

The λ value with the smallest CVM (5.2673109×10^8) is 39.8107171.

```
plot(lasso_cv)
```



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



Test Error with $\lambda = 39.81$:

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

The test error is 4.4304636×10^8 .

```
coef_1se = predict(lasso_cv, type = "coefficients", s = lambda_1se)
# 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 31 predictors.

b

```
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")
test_error = mean((y_test - y_pred)^2)
```

The selected tuning parameters are $\alpha = 0.45$ and $\lambda_{1SE} = 1584.89$. The model with these parameters has a test error of 4.2029165×10^8 .

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

```
set.seed(1234)

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

```

validation = "CV")
summary(pls_mod)

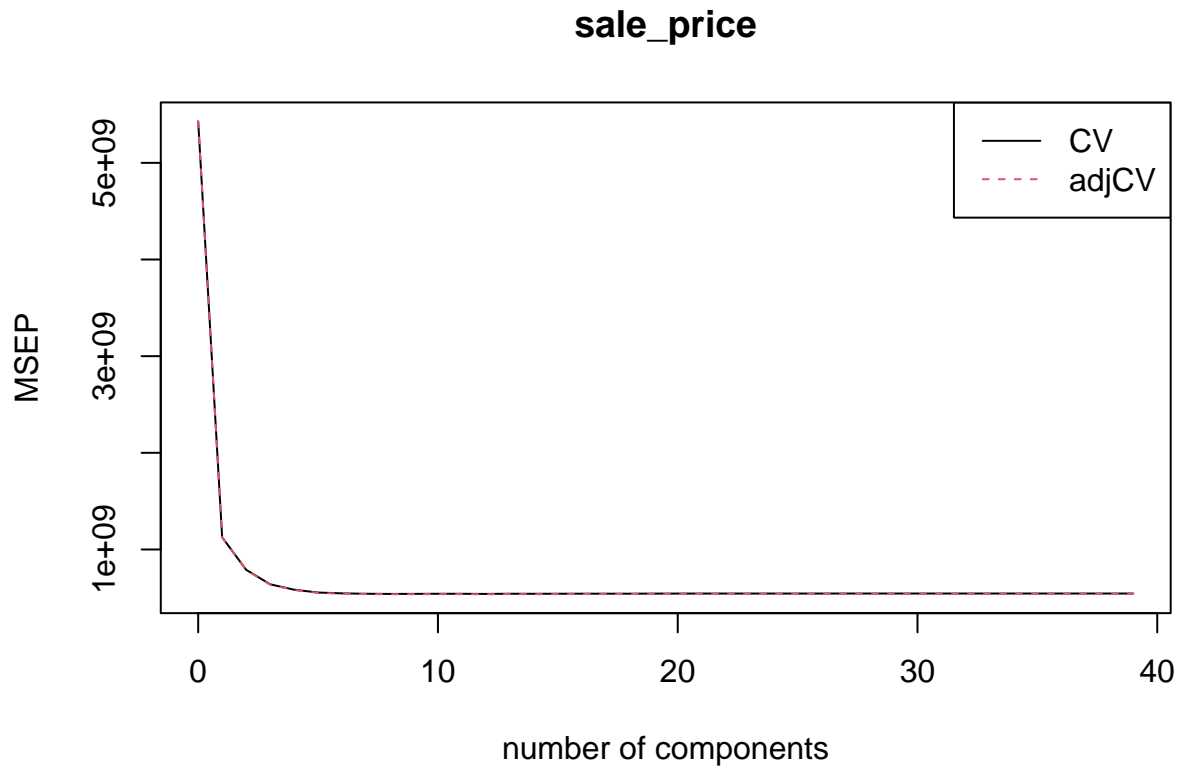
## Data:      X dimension: 1440 39
## Y dimension: 1440 1
## Fit method: kernelppls
## Number of components considered: 39
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##      (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV              73685   33553   28106   25289   24162   23546   23362
## adjCV           73685   33537   28060   25207   24086   23471   23295
##      7 comps  8 comps  9 comps 10 comps 11 comps 12 comps 13 comps
## CV       23277   23238   23250   23272   23269   23240   23282
## adjCV     23210   23173   23182   23200   23196   23170   23207
##      14 comps 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps
## CV       23266   23279   23295   23290   23294   23312   23315
## adjCV     23193   23205   23219   23215   23219   23235   23238
##      21 comps 22 comps 23 comps 24 comps 25 comps 26 comps 27 comps
## CV       23323   23322   23322   23322   23323   23324   23326
## adjCV     23245   23245   23244   23244   23245   23246   23248
##      28 comps 29 comps 30 comps 31 comps 32 comps 33 comps 34 comps
## CV       23326   23326   23326   23327   23327   23327   23327
## adjCV     23248   23248   23248   23248   23248   23248   23248
##      35 comps 36 comps 37 comps 38 comps 39 comps
## CV       23327   23327   23327   23327   23326
## adjCV     23248   23248   23248   23248   23266
##
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X          20.02   25.93   29.67   33.59   37.01   40.03   42.49
## sale_price  79.73   86.35   89.36   90.37   90.87   90.99   91.06
##      8 comps  9 comps 10 comps 11 comps 12 comps 13 comps 14 comps
## X          45.53   47.97   50.15   52.01   53.69   55.35   56.86
## sale_price  91.08   91.10   91.13   91.15   91.15   91.16   91.16
##      15 comps 16 comps 17 comps 18 comps 19 comps 20 comps
## X          58.64   60.01   62.18   63.87   65.26   67.10
## sale_price  91.16   91.16   91.16   91.16   91.16   91.16
##      21 comps 22 comps 23 comps 24 comps 25 comps 26 comps
## X          68.44   70.12   71.72   73.35   75.20   77.27
## sale_price  91.16   91.16   91.16   91.16   91.16   91.16
##      27 comps 28 comps 29 comps 30 comps 31 comps 32 comps
## X          78.97   80.10   81.83   83.55   84.39   86.34
## sale_price  91.16   91.16   91.16   91.16   91.16   91.16
##      33 comps 34 comps 35 comps 36 comps 37 comps 38 comps
## X          88.63   90.79   92.79   95.45   97.49   100.00
## sale_price  91.16   91.16   91.16   91.16   91.16   91.16
##      39 comps
## X          100.24
## sale_price  91.14

```

```

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

```



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

## 8 comps
##      8

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

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

There are 8 components in the partial least squares model, with an MSPE of 4.4021794×10^8 .