hw1

Johnstone Tcheou

2025-02-21

Contents


```
library(ISLR)
library(glmnet)
library(caret)
library(tidymodels)
library(corrplot)
library(ggplot2)
library(plotmo)
library(ggrepel)
library(pls)
```

```
training <- read.csv("housing_training.csv")
testing <- read.csv("housing_test.csv")
```

Question a

We will be using `glmnet` to fit a lasso model on the training data. Since `glmnet` functions require the predictors to be passed as a matrix and the response variable to be passed as a vector.

```
predictors <- model.matrix(Sale_Price ~ ., training)[, -1]
response <- training[, "Sale_Price"]
test <- model.matrix(Sale_Price ~ ., testing)[, -1]
```

Before fitting a model, we should also check for correlations between predictors, which may cause problems with lasso regression.

```r
# corrplot(cor(predictors),
#          method = "circle",
#          type = "full")
```

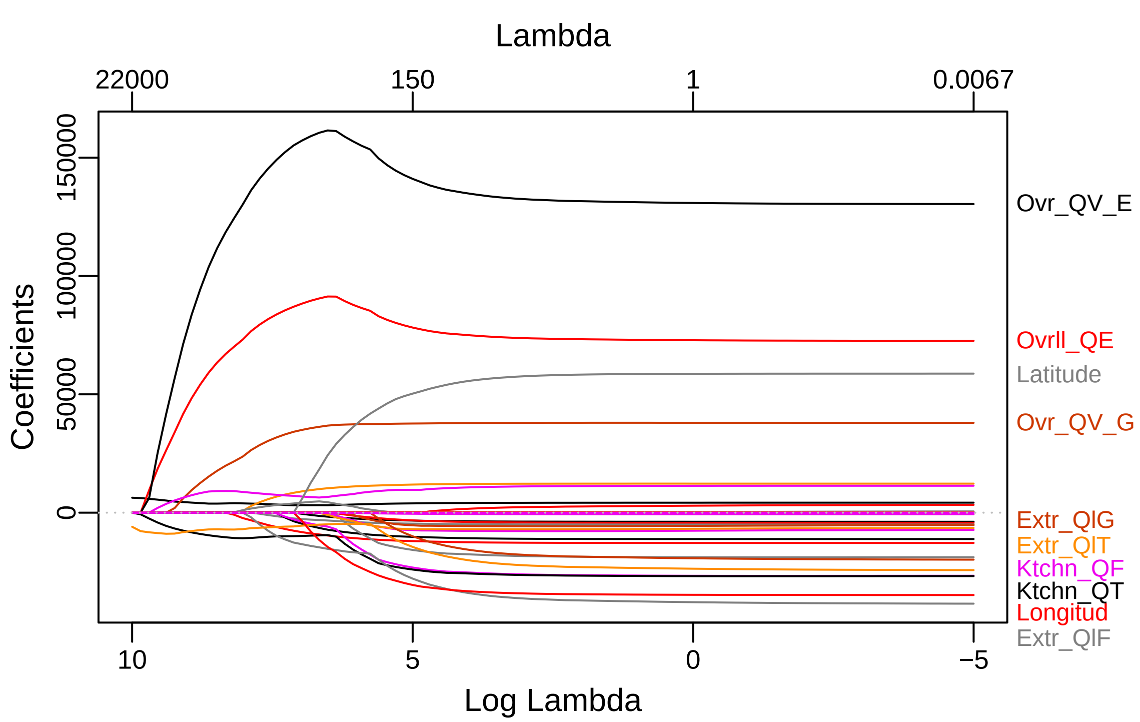
There are some predictors which are correlated with each other, such as `Total_Bsmt_SF` and `First_Flr_SF`, `Second_Flr_SF` and `Gr_Liv_Area`, `Kitchen_QualTypical` and `Kitchen_QualGood`, `Fireplaces` and `Fireplace_QuNo_FirePlace`.

Fitting the lasso model with `glmnet` requires `alpha` to be 1 for lasso regression, and passing a sequence of `lambda` values to hopefully capture the optimal lambda. The lambda values must be in descending order, hence the initial sequence value of 10 and the terminal sequence value of -20.

```r
lasso_glmnet <-
  glmnet(
    predictors,
    response,
    alpha = 1,
    lambda = exp(seq(10, -5, length = 100))
)

mat.coef <- coef(lasso_glmnet)
dim(mat.coef)
```

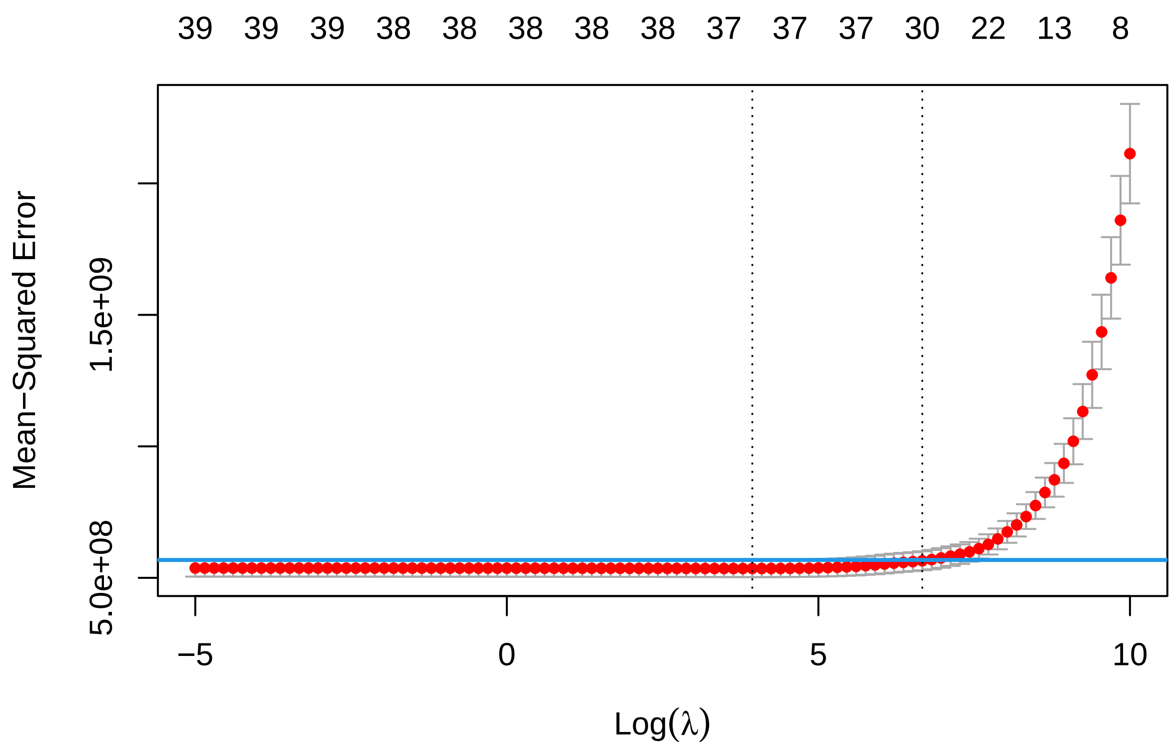
```
## [1]  40 100
```

```r
plot_glmnet(lasso_glmnet)
```



Use 10-fold cross validation to determine optimal lambda and regression parameters.

```
lasso_glmnet_cv <-
  cv.glmnet(
    predictors,
    response,
    alpha = 1,
    lambda = exp(seq(10, -5, length = 100))
  )

plot(lasso_glmnet_cv)
abline(h = (lasso_glmnet_cv$cvm + lasso_glmnet_cv$cvsd)[which.min(lasso_glmnet_cv$cvm)], col = 4, lwd =
```




```
lasso_glmnet_cv$lambda.1se
```

```
## [1] 785.772
```

```
lasso_glmnet_cv$cvm[which.min(lasso_glmnet_cv$cvm)]
```

```
## [1] 535383715
```

The lambda value, our tuning parameter, that minimizes MSE is given by 51.387448, and the smallest lambda value within 1 SE of the MSE is given by 785.7719942. With the lambda that minimizes MSE, the test error is 2.1130781×10^9, 1.8596203×10^9, 1.6406674×10^9, 1.4350298×10^9, 1.2719402×10^9, 1.1323557×10^9,

1.019273×10^9, 9.3529018×10^8, 8.7262273×10^8, 8.245815×10^8, 7.7503756×10^8, 7.32891×10^8, 7.0133707×10^8, 6.7473148×10^8, 6.4820695×10^8, 6.2716134×10^8, 6.1111748×10^8, 5.988317×10^8, 5.8987205×10^8, 5.8258281×10^8, 5.755101×10^8, 5.6938832×10^8, 5.6493395×10^8, 5.6157441×10^8, 5.5882602×10^8, 5.5585979×10^8, 5.525671×10^8, 5.4887811×10^8, 5.462704×10^8, 5.436162×10^8, 5.4161045×10^8, 5.3994082×10^8, 5.3880495×10^8, 5.3765419×10^8, 5.3685887×10^8, 5.3629444×10^8, 5.3591888×10^8, 5.3567669×10^8, 5.3551136×10^8, 5.3542918×10^8, 5.3538371×10^8, 5.3540238×10^8, 5.3543637×10^8, 5.3549198×10^8, 5.3557351×10^8, 5.3564031×10^8, 5.3570666×10^8, 5.3578233×10^8, 5.3585939×10^8, 5.3594259×10^8, 5.36015×10^8, 5.3611373×10^8, 5.3615623×10^8, 5.362255×10^8, 5.3626957×10^8, 5.3631241×10^8, 5.3635227×10^8, 5.363937×10^8, 5.3643508×10^8, 5.3647601×10^8, 5.3651301×10^8, 5.3654917×10^8, 5.3658419×10^8, 5.3661855×10^8, 5.366518×10^8, 5.3668214×10^8, 5.3671144×10^8, 5.3673967×10^8, 5.3676687×10^8, 5.3679305×10^8, 5.3681823×10^8, 5.3684244×10^8, 5.3686569×10^8, 5.3688799×10^8, 5.3690936×10^8, 5.3692979×10^8, 5.3694937×10^8, 5.3696811×10^8, 5.3698603×10^8, 5.3700315×10^8, 5.370196×10^8, 5.3703519×10^8, 5.3705007×10^8, 5.3706438×10^8, 5.3707793×10^8, 5.3709082×10^8, 5.3710309×10^8, 5.3711477×10^8, 5.371259×10^8, 5.3713649×10^8, 5.3714657×10^8, 5.3715617×10^8, 5.3716531×10^8, 5.37174×10^8, 5.3718227×10^8, 5.3719013×10^8, 5.3719762×10^8, 5.3720473×10^8, 5.372115×10^8, 5.3721794×10^8.

To get the parameters of the model which minimizes MSE, we need to pass the corresponding lambda, `lasso_glmnet_cv$lambda_min`, to the `s` argument of `predict()`.

```r
predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.min, type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                -4.857850e+06
## Gr_Liv_Area                 6.558437e+01
## First_Flr_SF                7.941717e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.537096e+01
## Low_Qual_Fin_SF            -4.116996e+01
## Wood_Deck_SF                1.171376e+01
## Open_Porch_SF               1.559648e+01
## Bsmt_Unf_SF                -2.088377e+01
## Mas_Vnr_Area                1.079970e+01
## Garage_Cars                 4.108515e+03
## Garage_Area                 8.108598e+00
## Year_Built                  3.238589e+02
## TotRms_AbvGrd              -3.662544e+03
## Full_Bath                  -3.945695e+03
## Overall_QualAverage        -4.891940e+03
## Overall_QualBelow_Average  -1.253114e+04
## Overall_QualExcellent       7.487820e+04
## Overall_QualFair           -1.083609e+04
## Overall_QualGood            1.214813e+04
## Overall_QualVery_Excellent  1.346299e+05
## Overall_QualVery_Good       3.789959e+04
## Kitchen_QualFair           -2.535278e+04
## Kitchen_QualGood           -1.766776e+04
## Kitchen_QualTypical        -2.575091e+04
## Fireplaces                  1.072137e+04
## Fireplace_QuFair           -7.701625e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.723065e+03
## Fireplace_QuPoor           -5.676385e+03
## Fireplace_QuTypical        -7.012112e+03
```

```
## Exter_QualFair             -3.419413e+04
## Exter_QualGood             -1.588680e+04
## Exter_QualTypical          -2.031623e+04
## Lot_Frontage                1.002166e+02
## Lot_Area                    6.043544e-01
## Longitude                  -3.329580e+04
## Latitude                    5.583816e+04
## Misc_Val                    8.484191e-01
## Year_Sold                  -5.778504e+02
```

```r
dim(predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.min, type = "coefficients"))
```

```
## [1] 40  1
```

```r
predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.1se, type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -2.416529e+06
## Gr_Liv_Area                 5.702348e+01
## First_Flr_SF                1.095166e+00
## Second_Flr_SF               .
## Total_Bsmt_SF               3.680178e+01
## Low_Qual_Fin_SF            -2.684114e+01
## Wood_Deck_SF                8.426736e+00
## Open_Porch_SF               8.230199e+00
## Bsmt_Unf_SF                -1.964099e+01
## Mas_Vnr_Area                1.420422e+01
## Garage_Cars                 3.140917e+03
## Garage_Area                 1.102330e+01
## Year_Built                  3.125272e+02
## TotRms_AbvGrd              -1.367640e+03
## Full_Bath                   .
## Overall_QualAverage        -3.111372e+03
## Overall_QualBelow_Average  -9.209300e+03
## Overall_QualExcellent       9.051030e+04
## Overall_QualFair           -6.458144e+03
## Overall_QualGood            9.962347e+03
## Overall_QualVery_Excellent  1.605120e+05
## Overall_QualVery_Good       3.627260e+04
## Kitchen_QualFair           -5.366575e+03
## Kitchen_QualGood            .
## Kitchen_QualTypical        -9.606954e+03
## Fireplaces                  6.406063e+03
## Fireplace_QuFair            .
## Fireplace_QuGood            4.763404e+03
## Fireplace_QuNo_Fireplace    .
## Fireplace_QuPoor            .
## Fireplace_QuTypical        -1.085364e+02
## Exter_QualFair             -1.465468e+04
## Exter_QualGood              .
## Exter_QualTypical          -5.052160e+03
## Lot_Frontage                7.168312e+01
```

```
## Lot_Area                       5.626606e-01
## Longitude                     -1.151561e+04
## Latitude                       1.827149e+04
## Misc_Val                       .
## Year_Sold                      .
```

```r
dim(predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.1se, type = "coefficients"))
```

```
## [1] 40  1
```

Both models, with and without the 1SE rule, have 40 parameters included in the model.

```r
predy2_lasso <- predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.min, newx = test, type = "response")

test_error_lasso <- mean((testing$Sale_Price - predy2_lasso)^2)
```

After fitting the lasso regression model to the testing dataset, the test error is 4.4144947×10^{8}.

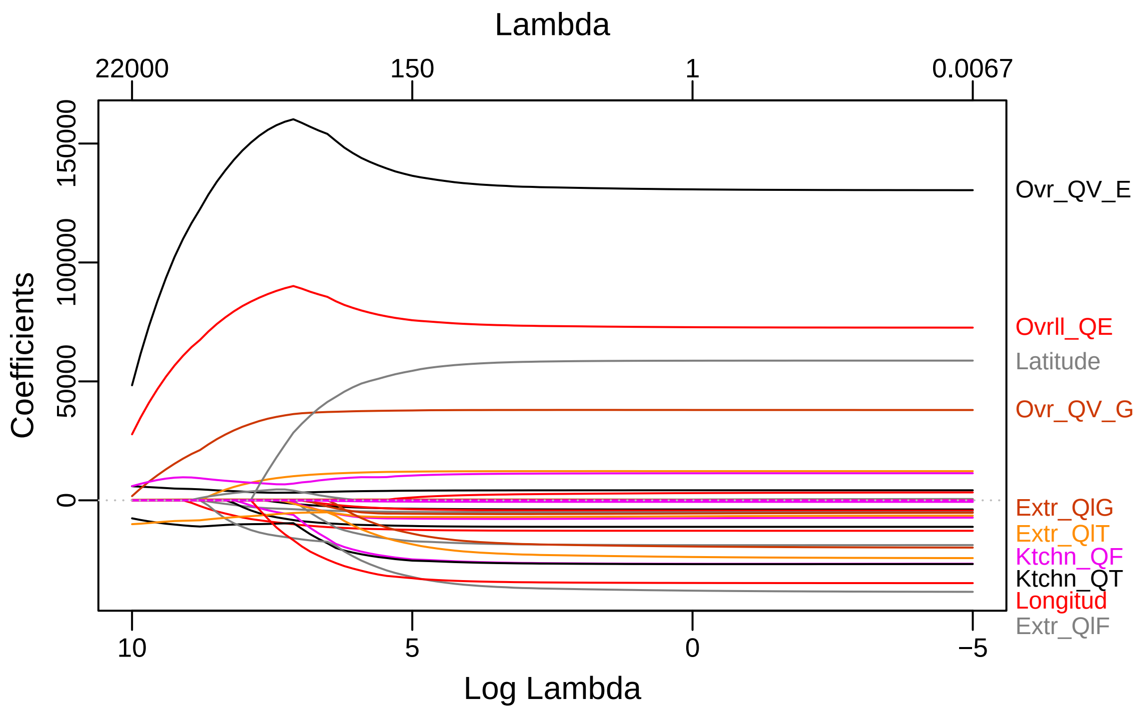
Question b

With elastic net, the `alpha` argument should now be 0.5, as it is in between lasso and ridge regression, incorporating the penalties from both models.

```r
elastic_glmnet <-
  glmnet(
    predictors,
    response,
    alpha = 0.5,
    lambda = exp(seq(10, -5, length = 100))
)

mat.coef <- coef(elastic_glmnet)
dim(mat.coef)
```

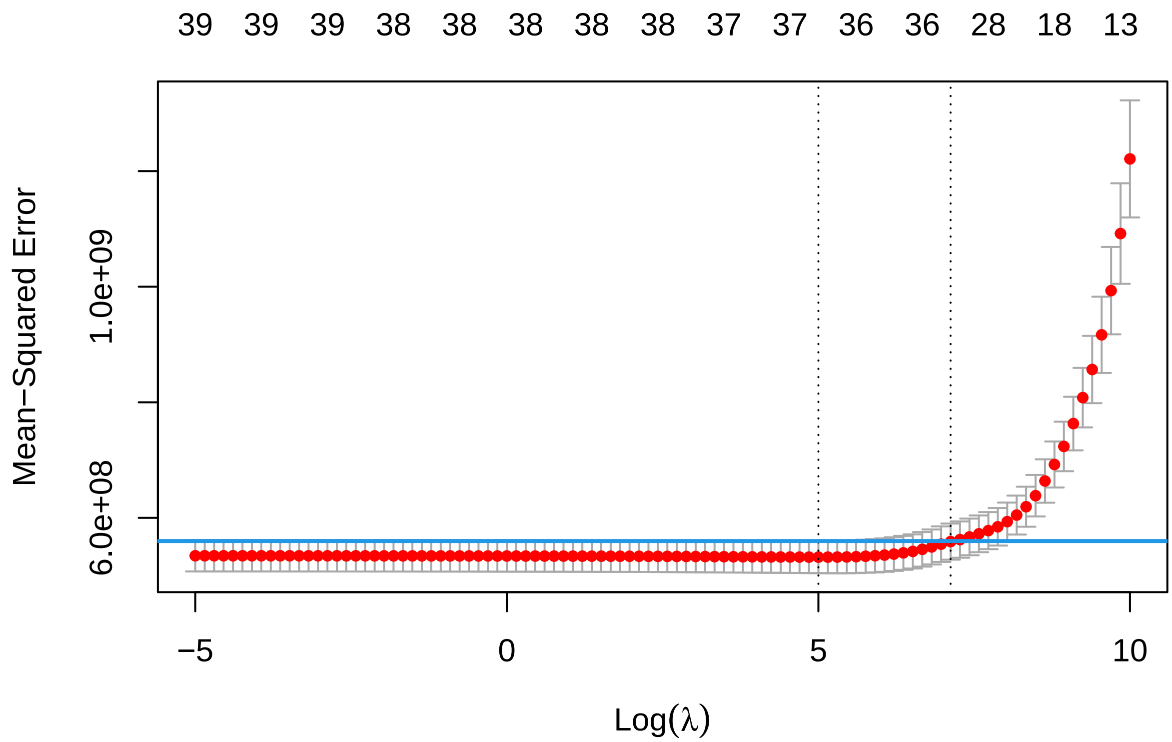
```
## [1]  40 100
```

```r
plot_glmnet(elastic_glmnet)
```



Likewise, use 10-fold cross validation to determine optimal lambda and number of parameters to have in model.

```
elastic_glmnet_cv <-
  cv.glmnet(
    predictors,
    response,
    alpha = 0.5,
    lambda = exp(seq(10, -5, length = 100))
  )

plot(elastic_glmnet_cv)
abline(h = (elastic_glmnet_cv$cvm + elastic_glmnet_cv$cvsd)[which.min(elastic_glmnet_cv$cvm)], col = 4,
```




```
elastic_glmnet_cv$lambda.1se
```

```
## [1] 1237.95
```

```
elastic_glmnet_cv$cvm[which.min(elastic_glmnet_cv$cvm)]
```

```
## [1] 532072382
```

The lambda which minimizes the cross validation MSE is 148.4131591. The corresponding test error is 5.3207238×10^8

Unlike lasso, the 1SE rule is not applicable to elastic net regression. This is because elastic net has 2 different lambdas, so the ideal alpha to balance the two lambdas is unclear.

```
predy2_elastic <- predict(elastic_glmnet_cv, s = elastic_glmnet_cv$lambda.min, newx = test, type = "res

test_error_elastic <- mean((testing$Sale_Price - predy2_elastic)^2)
```

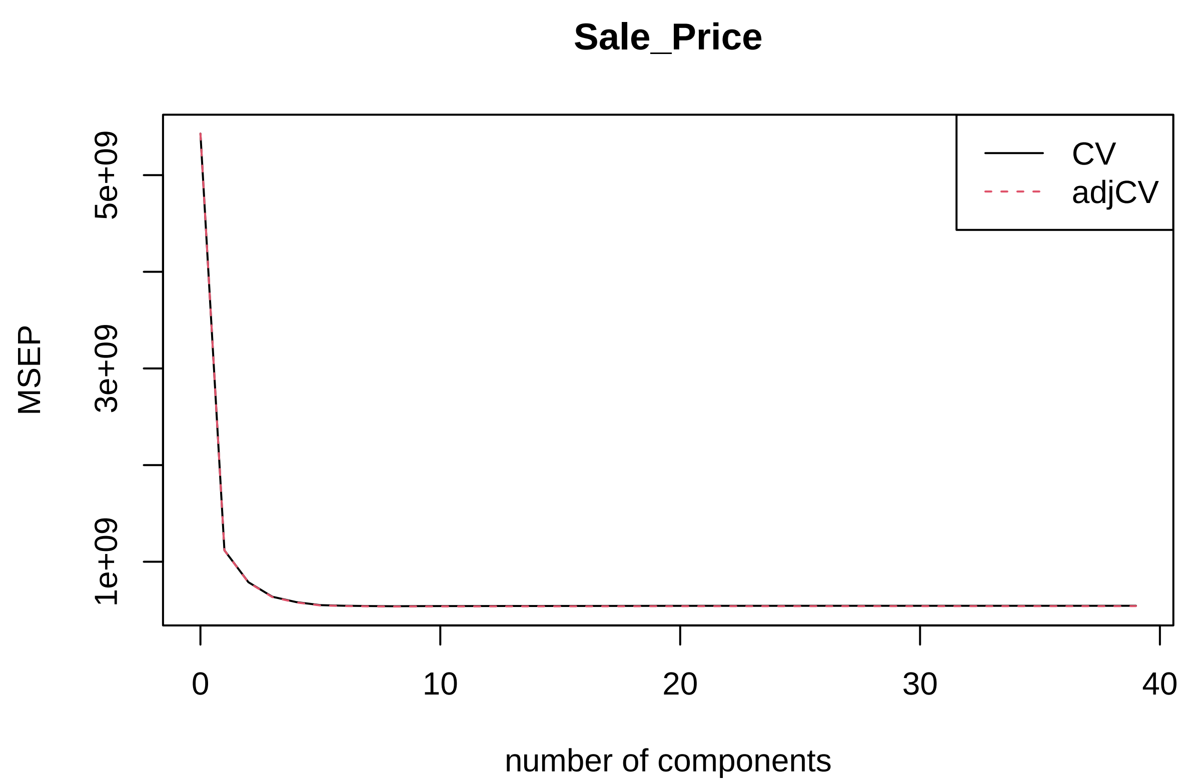
After fitting the elastic net model on the testing dataset, the test error is 4.3866617×10^8. # Question c

```
plsr_glmnet <- plsr(
  Sale_Price ~.,
  data = training,
  scale = T,
```

```r
  validation = "CV"
)

summary(plsr_glmnet)
```

```
## Data:    X dimension: 1440 39
##  Y dimension: 1440 1
## Fit method: kernelpls
## 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    33442    28081    25256    24132    23479    23343
## adjCV        73685    33436    28034    25178    24057    23410    23278
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23276    23235    23259     23270     23269     23276     23286
## adjCV    23210    23172    23192     23200     23197     23203     23212
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        23284     23293     23294     23299     23304     23315     23317
## adjCV     23210     23218     23219     23224     23228     23239     23240
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        23323     23324     23325     23326     23327     23329     23330
## adjCV     23246     23247     23248     23248     23249     23251     23252
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV        23331     23330     23330     23331     23330     23331     23331
## adjCV     23253     23252     23252     23253     23252     23253     23253
##        35 comps  36 comps  37 comps  38 comps  39 comps
## CV        23331     23331     23331     23331     23335
## adjCV     23253     23253     23253     23253     23262
##
## 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             100.67
## Sale_Price     91.16
```

```r
validationplot(plsr_glmnet, val.type = "MSEP", legendpos = "topright")
```




```r
cv_mse <- RMSEP(plsr_glmnet)
ncomp_cv <- which.min(cv_mse$val[1,,]) - 1
ncomp_cv
```

```
## 8 comps
##       8
```

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

test_error_pls <- mean((testing$Sale_Price - predy2_pls)^2)
```

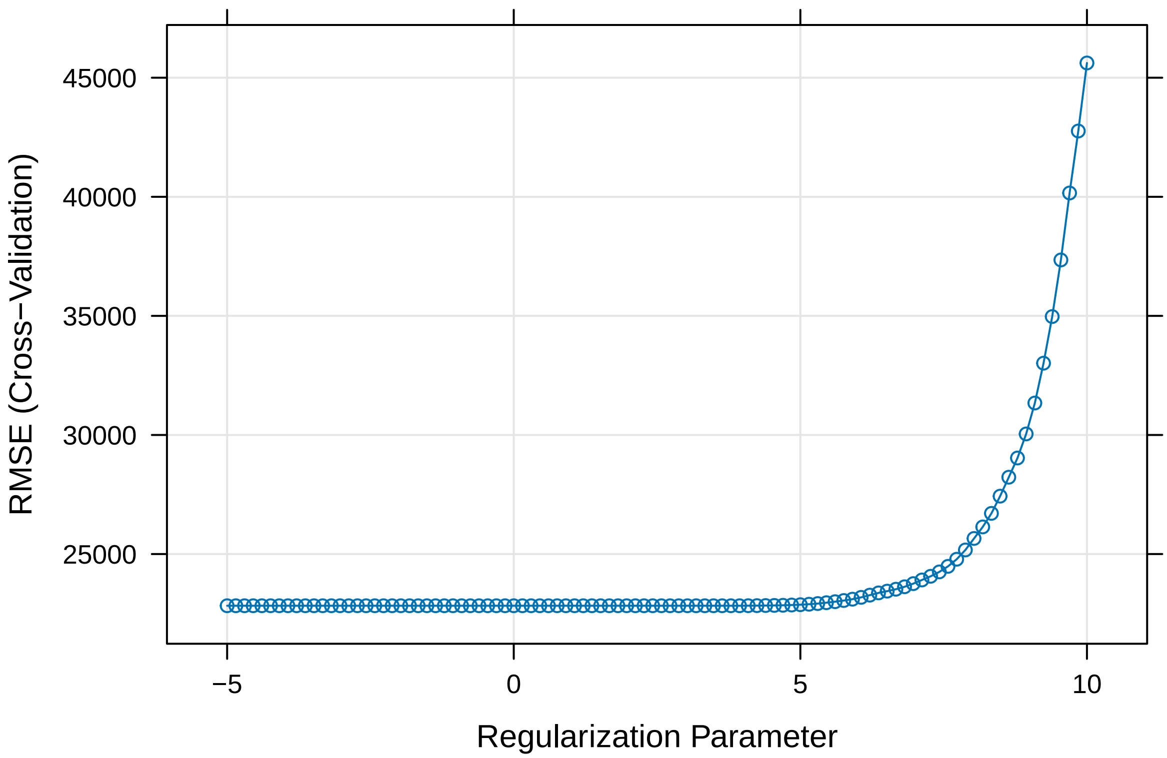
The partial least squares model with the smallest MSE has 8. The test error is 4.4021794×10^8.

Question d

The best model to predict the response is the model with the lowest test error - which is elastic net regression. Elastic net regression has a test MSE of 4.3866617×10^8, which is lower than the elastic net regression test MSE of 4.4144947×10^8 and lower than the partial least squares regression test MSE of 4.4021794×10^8

Question e

```
ctrl1 <- trainControl(method = "cv", number = 10)

lasso_caret <- train(Sale_Price ~ . ,
                     data = training,
                     method = "glmnet",
                     tuneGrid = expand.grid(alpha = 1,
                                            lambda = exp(seq(10, -5, length=100))),
                     trControl = ctrl1)

plot(lasso_caret, xTrans = log)
```

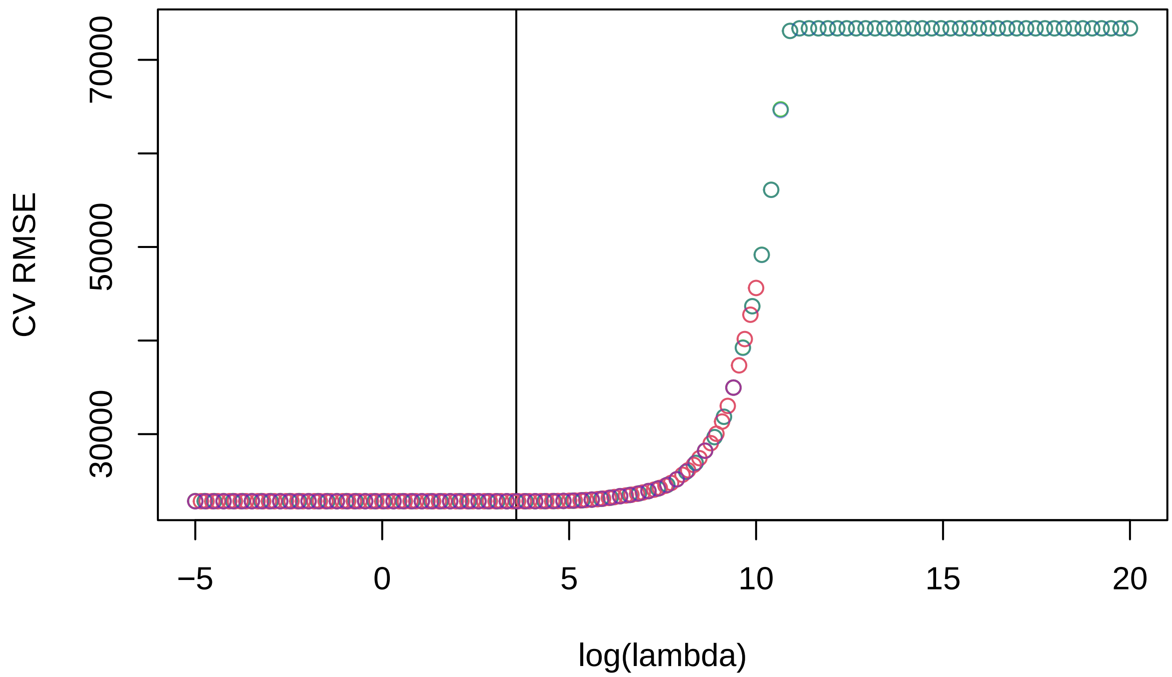



```
lasso_caret$bestTune
```

```
##    alpha   lambda
## 58     1 37.95357
```

```
train_id_list <- lasso_caret$control$index

dat_dummy <- data.frame(Sale_Price = response, predictors)
M <- 10
lambda.grid <- exp(seq(20, -5, length = 100))
```

```r
rmse <- rmse_caret <- matrix(NA, ncol = 100, nrow = M)

for (m in 1:M){
  tsdata <- dat_dummy[train_id_list[[m]],]
  vsdata <- dat_dummy[-train_id_list[[m]],]

  x1 <- as.matrix(tsdata[,-1])
  y1 <- tsdata[,1]
  x2 <- as.matrix(vsdata[,-1])
  y2 <- vsdata[,1]

  fit <- glmnet(x1, y1, alpha = 1,
                lambda = lambda.grid)

  # caret implementation did not specify lambda
  # the default grid of lambda is different from lambda.grid
  fit_caret <- glmnet(x1, y1, alpha = 1)

  pred <- predict(fit, newx = x2, s = lambda.grid)
  pred_caret <- predict(fit_caret, newx = x2, s = lambda.grid)

  rmse[m,] <- sqrt(colMeans((y2 - pred)^2))
  rmse_caret[m,] <- sqrt(colMeans((y2 - pred_caret)^2))
}

# curve from glmnet (correct)
plot(log(lambda.grid), colMeans(rmse), col = 3, xlab = "log(lambda)", ylab = "CV RMSE")
abline(v = log(lambda.grid[which.min(colMeans(rmse))]))

# caret results
points(log(lasso_caret$results$lambda), lasso_caret$results$RMSE, col = 2)

# try to reproduce caret results from scratch
points(log(lambda.grid), colMeans(rmse_caret), col = rgb(0,0,1,alpha = 0.3))
```



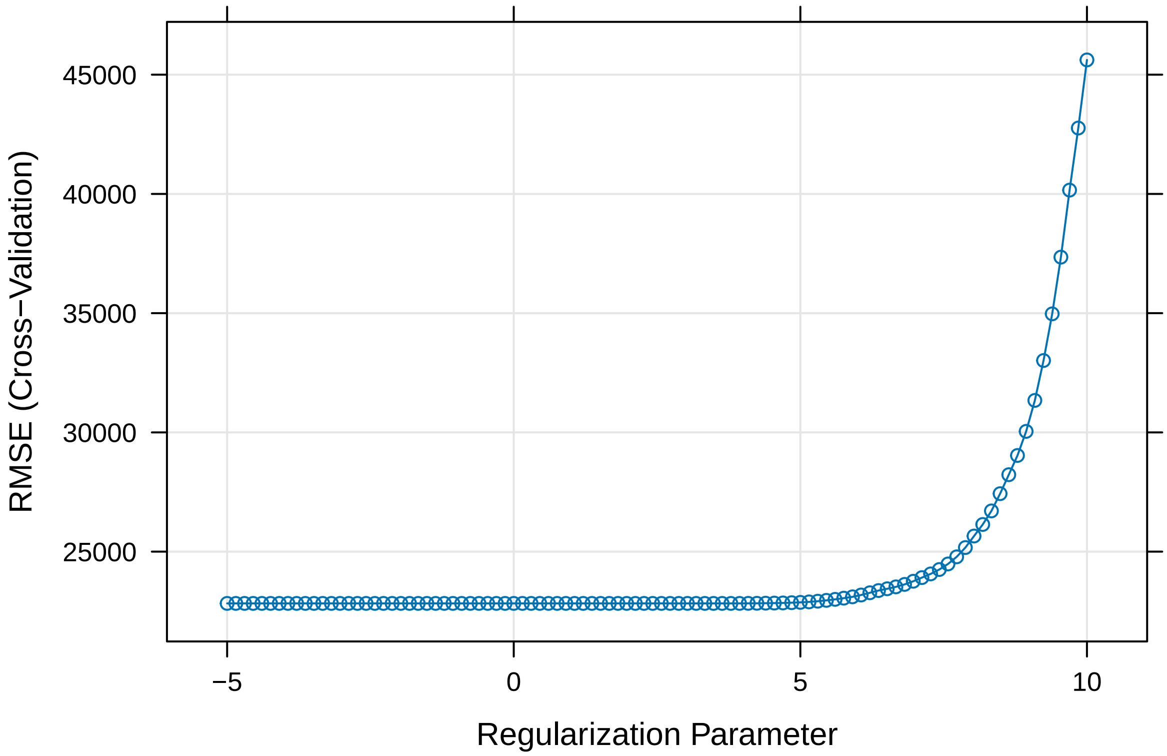

```r
# selected lambda
lambda.grid[which.min(colMeans(rmse))]
```

```
## [1] 36.08433
```

```r
# the corresponding CV RMSE
min(colMeans(rmse))
```

```
## [1] 22826.95
```

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




```r
lasso_caret$bestTune
```

```
##    alpha   lambda
## 58     1 37.95357
```

```r
# coefficients in the final model
coef(lasso_caret$finalModel, lasso_caret$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)          -4.891889e+06
## Gr_Liv_Area           6.576019e+01
## First_Flr_SF          7.854992e-01
## Second_Flr_SF         .
## Total_Bsmt_SF         3.533373e+01
## Low_Qual_Fin_SF      -4.132375e+01
## Wood_Deck_SF          1.179064e+01
## Open_Porch_SF         1.574960e+01
## Bsmt_Unf_SF          -2.089000e+01
## Mas_Vnr_Area          1.072139e+01
## Garage_Cars           4.139513e+03
## Garage_Area           8.020923e+00
## Year_Built            3.241105e+02
## TotRms_AbvGrd        -3.705645e+03
## Full_Bath            -4.036604e+03
```

```
## Overall_QualAverage         -4.924548e+03
## Overall_QualBelow_Average  -1.260070e+04
## Overall_QualExcellent       7.446458e+04
## Overall_QualFair           -1.091887e+04
## Overall_QualGood            1.218755e+04
## Overall_QualVery_Excellent  1.337787e+05
## Overall_QualVery_Good       3.793984e+04
## Kitchen_QualFair           -2.556466e+04
## Kitchen_QualGood           -1.785058e+04
## Kitchen_QualTypical        -2.590666e+04
## Fireplaces                  1.087859e+04
## Fireplace_QuFair           -7.738271e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.978771e+03
## Fireplace_QuPoor           -5.709901e+03
## Fireplace_QuTypical        -7.015803e+03
## Exter_QualFair             -3.511041e+04
## Exter_QualGood             -1.674606e+04
## Exter_QualTypical          -2.116559e+04
## Lot_Frontage                1.007469e+02
## Lot_Area                    6.044410e-01
## Longitude                  -3.366154e+04
## Latitude                    5.661433e+04
## Misc_Val                    8.658075e-01
## Year_Sold                  -5.939770e+02
```