p8106_hw1_lr3257

Leyang Rui

2025-02-15

p8106 Homework 1

Data import & clearning

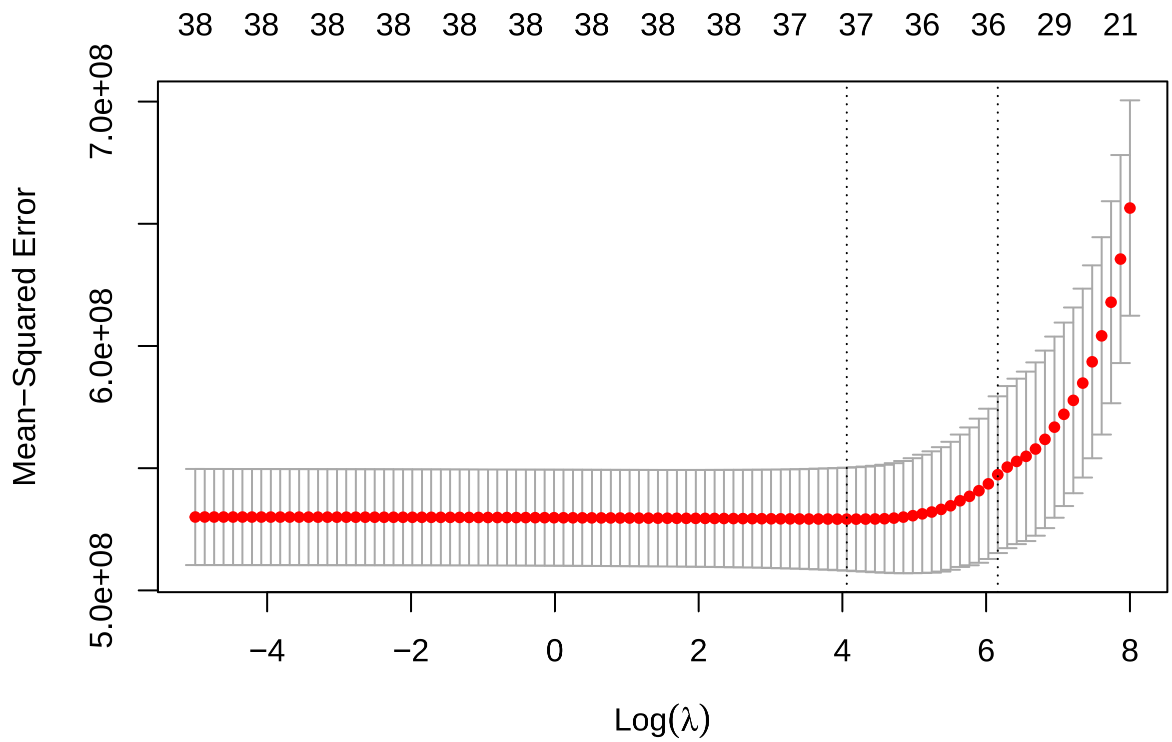
For easier model interpretation, I changed variables year_built and year_sold from categorical to numerical variables.

```
housing_test = read_csv("data/housing_test.csv") |>
  janitor::clean_names() |>
  mutate(year_built = as.numeric(year_built),
         year_sold = as.numeric(year_sold))

housing_train = read_csv("data/housing_training.csv") |>
  janitor::clean_names() |>
  mutate(year_built = as.numeric(year_built),
         year_sold = as.numeric(year_sold))
```

(a) Fit a lasso model on the training data. Report the selected tuning parameter and the test error. When the 1SE rule is applied, how many predictors are included in the model?

```
x = model.matrix(sale_price ~ ., housing_train)[, -1]
y = housing_train[["sale_price"]]
```

```
lasso.model = glmnet(x, y, alpha = 1,
                        lambda = exp(seq(8, -5, length = 100)))

cv.lasso = cv.glmnet(x, y, alpha = 1,
                        lambda = exp(seq(8, -5, length = 100)))

plot(cv.lasso)
```




```
cv_lambda_min = cv.lasso$lambda.min

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

y_pred_lasso = predict(lasso.model, newx = x_test, s = cv_lambda_min)

test_mse_lasso = mean((y_test - y_pred_lasso)^2)
```

For a smallest test MSE of 4.407688×10^8, the lambda is about 58.0094577

```
cv.lasso$lambda.1se
```

```
## [1] 474.1938
```

```
coef_1se = coef(cv.lasso, s = cv.lasso$lambda.1se)
num_coef_1se = sum(coef_1se[-1] != 0)
```

When the 1SE rule is applied, around 36 predictors are included in the model.

(b) Fit an elastic net model on the training data. Report the selected tuning parameters and the test error. Is it possible to apply the 1SE rule to select the tuning parameters for elastic net? If the 1SE rule is applicable, implement it to select the tuning parameters. If not, explain why.

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

enet.model = train(sale_price ~ .,
                  data = housing_train,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(-10, 10, length = 100))),
                  trControl = ctrl1)
enet.model$bestTune
```

```
##     alpha   lambda
## 181  0.05 474.1938
```

```r
x_test = housing_test
y_test = housing_test$sale_price

y_pred_enet = predict(enet.model, newdata = x_test)

test_mse_enet = mean((y_test - y_pred_enet)^2)
```

For a smallest test MSE of 4.3987576×10^8, the lambda is about 474.1938297

Yes, the 1SE rule is applicable:

```r
min_rmse = min(enet.model$results$RMSE)

se_rmse = sd(enet.model$results$RMSE) / sqrt(nrow(enet.model$resample))
lambda_1se_threshold = min_rmse + se_rmse

enet_1se = enet.model$results |>
  filter(RMSE <= lambda_1se_threshold) |>
  arrange(desc(lambda)) |>
  slice(1)
```

The tuning parameter lambda using 1SE rule is 9817.4745282

(c) Fit a partial least squares model on the training data and report the test error. How many components are included in your model?

```r
pls.model = plsr(sale_price ~ ., data = housing_train,
                 scale = TRUE, validation = "CV")
summary(pls.model)
```

```
## 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    33410    27986    25121    23916    23276    23140
```

```
## adjCV        73685    33405    27949    25049    23857    23217    23084
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23027    22997    23013     23009     22985     22974     22979
## adjCV    22974    22945    22957     22952     22928     22917     22922
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        22978     22982     22976     22981     22992     22993     22996
## adjCV     22921     22925     22919     22924     22934     22935     22938
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        22999     23001     23003     23003     23004     23004     23006
## adjCV     22940     22942     22944     22944     22945     22945     22947
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV        23007     23008     23008     23008     23008     23008     23008
## adjCV     22948     22948     22948     22948     22948     22948     22948
##        35 comps  36 comps  37 comps  38 comps  39 comps
## CV        23008     23008     23008     23008     23019
## adjCV     22948     22948     22948     22948     22933
##
## 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
cv_mse = RMSEP(pls.model)
ncomp_cv = which.min(cv_mse$val[1,,]) - 1

y_pred_pls = predict(pls.model, newdata = housing_test,
                   ncomp = ncomp_cv)
test_mse_pls = mean((y_test - y_pred_pls)^2)
```

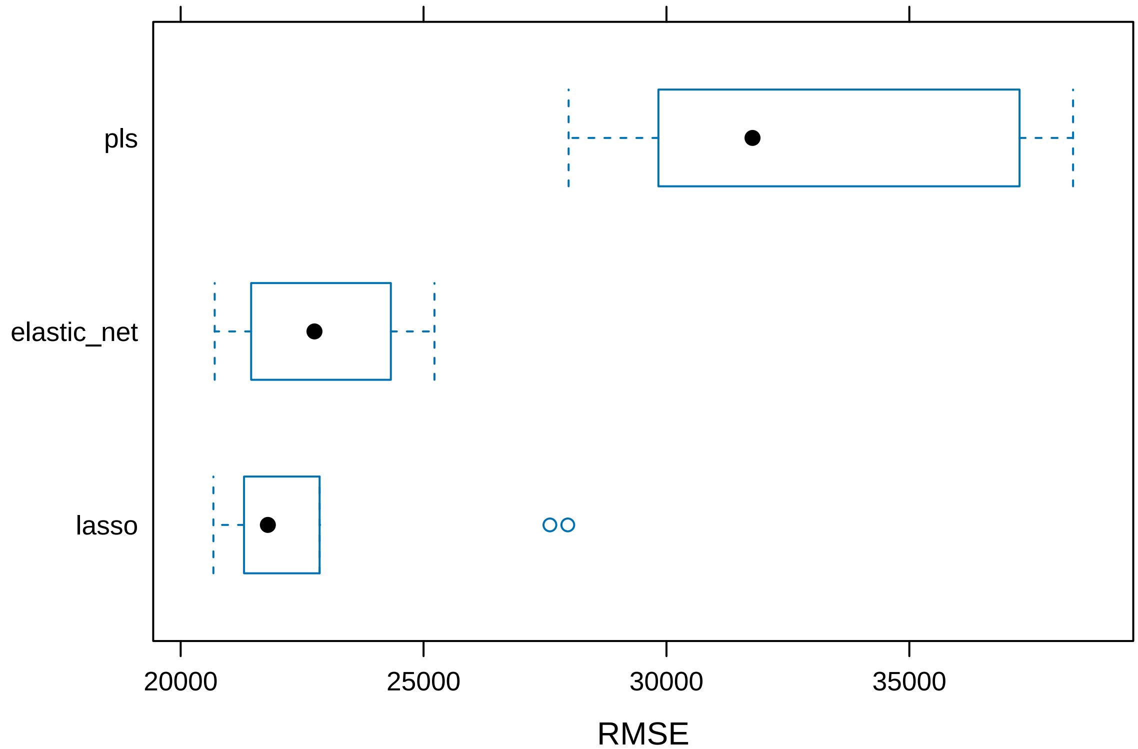
The test MSE of the partial least squares model is 4.4962272×10^8, while the model has 12 components

(d) Choose the best model for predicting the response and explain your choice.

```r
lasso.model = train(sale_price ~ ., data = housing_train,
                     method = "glmnet",
                     tuneGrid = expand.grid(alpha = 1,
                                            lambda = exp(seq(-10, 10, length = 100))),
                     trControl = ctrl1)

pls.model = train(sale_price ~ ., data = housing_train,
                   method = "pls",
                   tuneLength = 10,
                   trControl = ctrl1)

resamp = resamples(list(lasso = lasso.model,
                        elastic_net = enet.model,
                        pls = pls.model))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, elastic_net, pls
## Number of resamples: 10
##
## MAE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso       15459.28 16041.42 16478.26 16794.05 17178.92 18984.35    0
## elastic_net 15227.38 15931.63 16462.11 16608.21 17430.55 18344.76    0
## pls         20346.63 22287.51 22621.64 23344.97 25150.73 27392.32    0
##
## RMSE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso       20673.19 21336.58 21795.02 22930.58 22813.03 27968.86    0
## elastic_net 20700.39 21464.36 22754.18 22886.18 24283.02 25224.07    0
## pls         27985.84 29938.37 31771.50 32693.58 36357.34 38371.00    0
##
## Rsquared
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso       0.8872972 0.8945839 0.8995071 0.9023868 0.9062259 0.9244404    0
## elastic_net 0.8672210 0.8895533 0.9077499 0.9013775 0.9162796 0.9233194    0
## pls         0.7339267 0.7872989 0.8063534 0.8031579 0.8241052 0.8543557    0
```

```r
bwplot(resamp, metric = "RMSE")
```



I will choose the lasso model as the best model. Since the lasso model has both a smallest median RMSE and its distribution of RMSE is the most concentrated (despite the outliers), although it has a few outlier cases, it will still be the best and the most stable model in general.

(e) If R package "caret" was used for the lasso in (a), retrain this model using R package "glmnet", and vice versa. Compare the selected tuning parameters between the two software approaches. Should there be discrepancies in the chosen parameters, discuss potential reasons for these differences.

```
### Using caret here:
lasso.model = train(sale_price ~ ., data = housing_train,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                                           lambda = exp(seq(10, -10, length = 100))),
                    trControl = ctrl1)
lasso.model$bestTune
```

```
##    alpha   lambda
## 72     1 76.97143
```

```
cv_lambda_min
```

```
## [1] 58.00946
```

The results above show that the lambda from using the "caret" package is larger than that from using the "glmnet" package. The difference may come from how the "glmnet" method does cross validation together with building the model, while the "caret" method has an extra step of standardizing tuning process.