P8106 HW1

Shihui Zhu

Contents

(a) Least Square

Fit a linear model using least squares on the training data. Is there any potential disadvantage of this model?

```r
house_training <- read_csv("housing_training.csv") %>%
  janitor::clean_names()
```

```
## Rows: 1440 Columns: 26

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): Overall_Qual, Kitchen_Qual, Fireplace_Qu, Exter_Qual
## dbl (22): Gr_Liv_Area, First_Flr_SF, Second_Flr_SF, Total_Bsmt_SF, Low_Qual_...

##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

Perform 10-folds cross-validation on the full model

There are 25 predictors in total, we want to perform CV to find the best fitted parameters.

```
# Reproducibility
set.seed(1)
fit.lm <- train(sale_price ~ .,
                data = house_training,
                method = "lm",
                trControl = trainControl(method = "cv", number = 10))
# Print the coefficients of the final model
fit.lm$finalModel$coefficients
```

```
##                (Intercept)                 gr_liv_area
##              -4.984628e+06                2.458001e+01
##               first_flr_sf               second_flr_sf
##               4.252481e+01                4.176763e+01
##              total_bsmt_sf             low_qual_fin_sf
##               3.519195e+01                          NA
##               wood_deck_sf               open_porch_sf
##               1.202429e+01                1.617780e+01
##                bsmt_unf_sf                mas_vnr_area
##              -2.086972e+01                1.045801e+01
##                garage_cars                 garage_area
##               4.228630e+03                7.769075e+00
##                 year_built             tot_rms_abv_grd
##               3.251188e+02               -3.838101e+03
##                  full_bath       overall_qualAverage
##              -4.340544e+03               -5.013000e+03
##  overall_qualBelow_Average     overall_qualExcellent
##              -1.279921e+04                7.260609e+04
##           overall_qualFair            overall_qualGood
##              -1.114686e+04                1.226308e+04
## overall_qualVery_Excellent     overall_qualVery_Good
##               1.303770e+05                3.797518e+04
##           kitchen_qualFair            kitchen_qualGood
##              -2.662852e+04               -1.878675e+04
##        kitchen_qualTypical                  fireplaces
##              -2.676512e+04                1.138083e+04
##           fireplace_quFair            fireplace_quGood
##              -7.206510e+03                6.069919e+02
##    fireplace_quNo_Fireplace            fireplace_quPoor
##               3.394267e+03               -5.184597e+03
##        fireplace_quTypical             exter_qualFair
##              -6.397856e+03               -3.854088e+04
##             exter_qualGood          exter_qualTypical
##              -1.993620e+04               -2.436104e+04
##               lot_frontage                    lot_area
##               1.024022e+02                6.041670e-01
##                  longitude                    latitude
##              -3.481322e+04                5.874429e+04
##                   misc_val                   year_sold
##               9.171290e-01               -6.454655e+02
```

MSE, R-squared values

Check the mean squared error (MSE) and the R-squared value of the model:

```
# The Mean Squared Error is
mean((fit.lm$resample$RMSE)^2)
```

```
## [1] 531169999
```

```
# The R^2 value is
mean(fit.lm$resample$Rsquared)
```

```
## [1] 0.9029694
```

So the training error for the LS model is 531169999, with R-squared value of 0.9029694.

Potential disadvantage: 1. The model has the smallest variance among all the unbiased models, but it has larger variance compared to biased models, but it is not necessarily good enough. It may be better to have a biased estimator with smaller variance/mean squared error. 2. There are many predictors used in this model, and this can cause problem (large variance, colinearity). 3. The model is too complex, a simple model might be better. 4. This model has overfitting problem, meaning that it may perform badly on testing set.

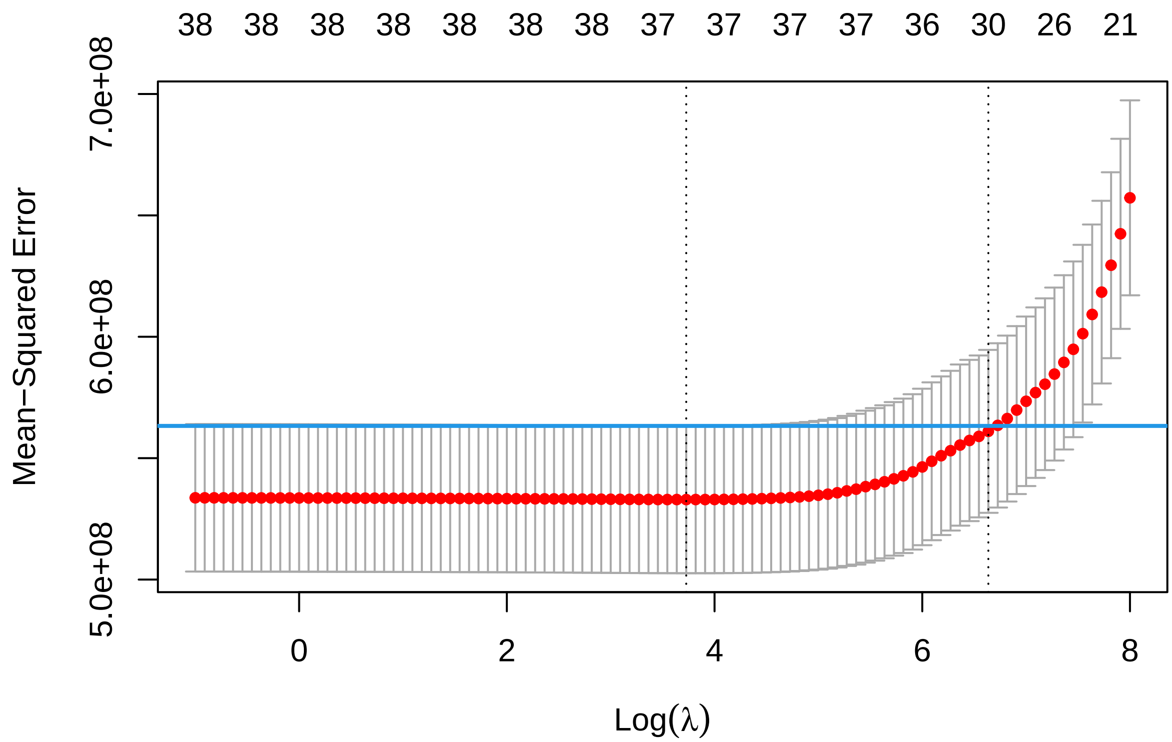
(b) Lasso

```
x_train <- model.matrix(sale_price ~ ., house_training)[ ,-1]
y_train <- house_training$sale_price
```

Fit the model using Lasso Regression

```
set.seed(1)
# fit the laso regression (alpha = 1) with a sequence of lambdas
cv.lasso <- cv.glmnet(x = x_train, y = y_train,
                      standardize = TRUE,
                      alpha = 1,
                      lambda = exp(seq(8, -1, length = 100)))
```

```
plot(cv.lasso)
abline(h = (cv.lasso$cvm + cv.lasso$cvsd)[which.min(cv.lasso$cvm)], col = 4, lwd = 2)
```



When 1se rule is applied, there are total 30 predictors in the model.

The coefficient of LASSO regression model is:

```
predict(cv.lasso, s = cv.lasso$lambda.1se, type = "coefficient")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                        s1
## (Intercept)                 -2.535481e+06
## gr_liv_area                  5.723258e+01
## first_flr_sf                 1.066538e+00
## second_flr_sf                .
## total_bsmt_sf                3.678078e+01
## low_qual_fin_sf             -2.724627e+01
## wood_deck_sf                 8.467731e+00
## open_porch_sf                8.403562e+00
## bsmt_unf_sf                 -1.971379e+01
## mas_vnr_area                 1.414590e+01
## garage_cars                  3.078143e+03
## garage_area                  1.115981e+01
## year_built                   3.123467e+02
## tot_rms_abv_grd             -1.439453e+03
## full_bath                    .
## overall_qualAverage         -3.158907e+03
## overall_qualBelow_Average   -9.303478e+03
## overall_qualExcellent        9.066941e+04
```

```
## overall_qualFair           -6.610047e+03
## overall_qualGood            1.002368e+04
## overall_qualVery_Excellent  1.606478e+05
## overall_qualVery_Good       3.636198e+04
## kitchen_qualFair           -5.480630e+03
## kitchen_qualGood            .
## kitchen_qualTypical        -9.580041e+03
## fireplaces                  6.514237e+03
## fireplace_quFair            .
## fireplace_quGood            4.630463e+03
## fireplace_quNo_Fireplace    .
## fireplace_quPoor            .
## fireplace_quTypical        -3.377484e+02
## exter_qualFair             -1.481214e+04
## exter_qualGood              .
## exter_qualTypical          -5.036385e+03
## lot_frontage                7.261206e+01
## lot_area                    5.648507e-01
## longitude                  -1.213699e+04
## latitude                    1.973031e+04
## misc_val                    .
## year_sold                   .
```

Training and Testing Error

Calculate the MSE of the test set

```r
housing_testing <- read_csv("housing_test.csv") %>%
  janitor::clean_names()
x_test <- model.matrix(sale_price ~ ., housing_testing)[ ,-1]
y_test <- housing_testing$sale_price

# Training Error
y_pred_t <- predict(cv.lasso, newx = x_train, s = "lambda.1se", type = "response")
mean(RMSE(y_pred_t, y_train)^2)
```

```
## [1] 515636300
```

The training error (MSE) is 515636300 for lasso model.

```r
y_pred <- predict(cv.lasso, newx = x_test, s = "lambda.1se", type = "response")
lasso_te <- mean(RMSE(y_pred, y_test)^2)
lasso_te
```

```
## [1] 420354616
```

The test error (MSE) is 420354616 for lasso regression model when the 1SE rule is applied.

c) Fit an elastic net model on the training data

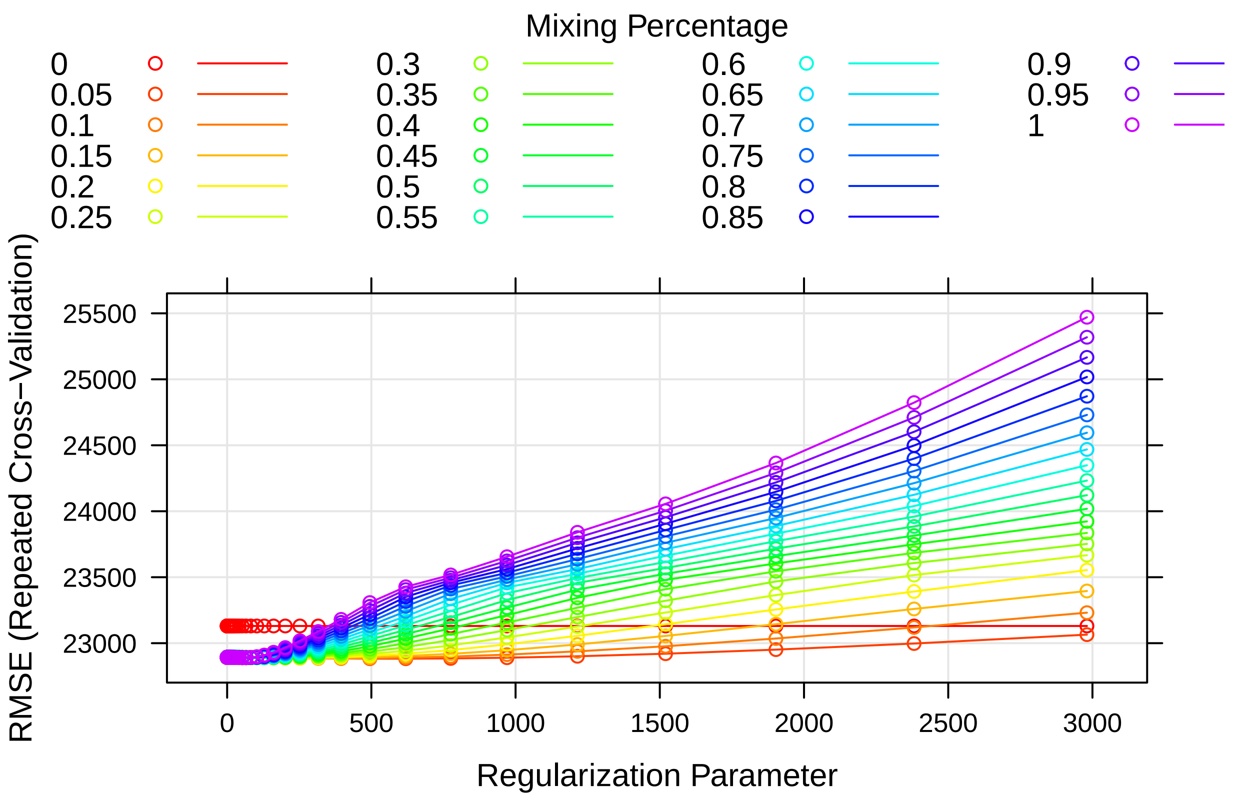
Report the selected tuning parameters

```
set.seed(2)
enet.fit <- train(x_train, y_train,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(8, -3, length = 50))),
                  trControl = trainControl(method = "repeatedcv", number = 10, repeats = 5))
enet.fit$bestTune
```

```
##    alpha   lambda
## 93  0.05 619.2886
```

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

plot(enet.fit, par.settings = myPar)
```



```
coef(enet.fit$finalModel, enet.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                -5.119099e+06
## gr_liv_area                 3.871803e+01
## first_flr_sf                2.667284e+01
## second_flr_sf               2.539870e+01
## total_bsmt_sf               3.492629e+01
## low_qual_fin_sf            -1.586669e+01
## wood_deck_sf                1.234466e+01
## open_porch_sf               1.692220e+01
## bsmt_unf_sf                -2.071319e+01
## mas_vnr_area                1.172260e+01
## garage_cars                 4.041408e+03
## garage_area                 8.948237e+00
## year_built                  3.188394e+02
## tot_rms_abv_grd            -3.419303e+03
## full_bath                  -3.659651e+03
## overall_qualAverage        -5.119003e+03
## overall_qualBelow_Average  -1.269888e+04
## overall_qualExcellent       7.592629e+04
## overall_qualFair           -1.148475e+04
## overall_qualGood            1.195840e+04
## overall_qualVery_Excellent  1.365876e+05
## overall_qualVery_Good       3.762260e+04
## kitchen_qualFair           -2.354051e+04
## kitchen_qualGood           -1.597423e+04
## kitchen_qualTypical        -2.402780e+04
## fireplaces                  1.079788e+04
## fireplace_quFair           -7.863526e+03
## fireplace_quGood            1.474300e+02
## fireplace_quNo_Fireplace    1.757694e+03
## fireplace_quPoor           -5.809405e+03
## fireplace_quTypical        -6.964071e+03
## exter_qualFair             -3.276755e+04
## exter_qualGood             -1.436584e+04
## exter_qualTypical          -1.897822e+04
## lot_frontage                1.000357e+02
## lot_area                    6.030183e-01
## longitude                  -3.517964e+04
## latitude                    5.769405e+04
## misc_val                    8.652364e-01
## year_sold                  -5.712351e+02
```

The tunning parameter λ is 619.2886. The parameter α is 0.05.

Test error

```
enet.pred <- predict(enet.fit, newdata = x_test)
# test error
```

```r
enet_te <- mean(RMSE(enet.pred, y_test)^2)
enet_te
```

```
## [1] 438209306
```

The test error (MSE) is 438209306 for elastic net model.

(d) Fit a partial least squares model

Fit model using plsr

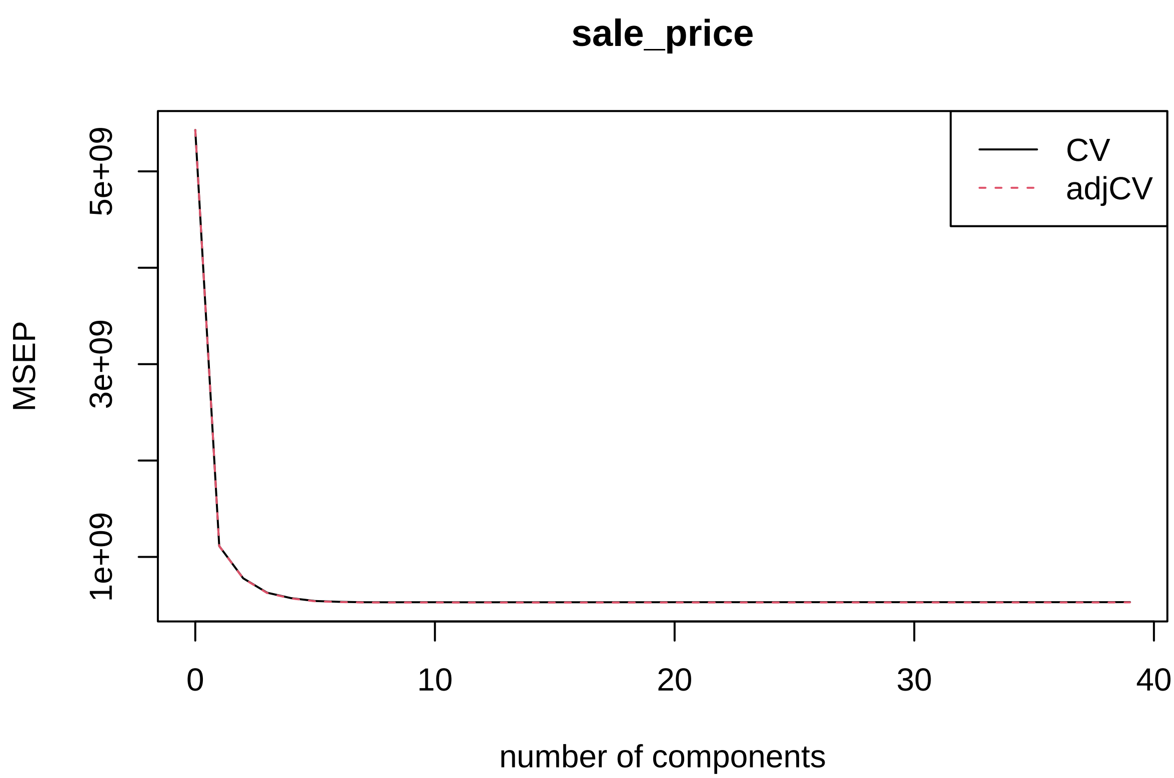
```r
set.seed(2)
pls.mod <- plsr(sale_price~.,
                data = house_training,
                scale = TRUE,
                validation = "CV")

summary(pls.mod)
```

```
## 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    33357    27895    25075    23920    23294    23114
## adjCV        73685    33354    27863    25006    23858    23233    23061
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23020    23007    23021     23022     23011     23011     23013
## adjCV    22968    22954    22965     22964     22953     22952     22953
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        23010     23016     23014     23014     23019     23022     23024
## adjCV     22951     22956     22954     22954     22959     22961     22964
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        23027     23031     23029     23030     23030     23032     23033
## adjCV     22967     22970     22968     22969     22969     22971     22972
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV        23034     23034     23034     23034     23034     23034     23034
## adjCV     22973     22973     22972     22973     22973     22973     22973
##        35 comps  36 comps  37 comps  38 comps  39 comps
## CV        23034     23034     23034     23034     23052
## adjCV     22973     22973     22973     22973     22990
##
## 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(pls.mod, val.type="MSEP", legendpos = "topright")
```



Training Error and Testing Error

```r
# training error
cv.mse <- RMSEP(pls.mod)
mean(min(cv.mse$val[1,,])^2)
```

```
## [1] 529300397
```

```r
ncomp.cv <- which.min(cv.mse$val[1,,])-1 # extract the response and delete the 0th component
# num of components
ncomp.cv
```

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

```r
# Prediction
pls_pred <- predict(pls.mod, newdata = x_test,
                    ncomp = ncomp.cv)
# test MSE
pls_te <- mean(RMSE(y_test, pls_pred)^2)
pls_te
```

```
## [1] 440217938
```

The model with 8 components yields the minimum training error (MSE) 529300397. The testing error (MSE) is 440217938.

e) Select Model

```r
#compute the testing error of LS regression first
y_pred <- predict(fit.lm, newdata = housing_testing)
ls_te <- mean(RMSE(y_pred, y_test)^2)
ls_te
```

```
## [1] 447287652
```

The test error of LS model is 447287652.

For response predicting, we want to choose the model with the smallest test error. The test error (MSE) is 420354616 for lasso regression model, 438209306 for elastic net model, and 440217938 for the PLS model. Therefore, we should choose the lasso regression model for predicting the response.