p8106_hw1

Hao Zheng

2/20/2022

```
# Data import
train = read.csv("./data/housing_training.csv") %>% janitor::clean_names()
test = read.csv("./data/housing_test.csv") %>% janitor::clean_names()

train = na.omit(train)
test = na.omit(test)

x_train = model.matrix(sale_price~., train)[,-1]
y_train = train$sale_price

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

Now, let's fit different models based on the dataset.

Linear model

```
set.seed(2022)

lm.fit = lm(sale_price ~ .,
            data = train,
            method = "lm",
            trControl = trainControl(method = "repeatedcv", number = 10))
```

```
## Warning in lm(sale_price ~ ., data = train, method = "lm", trControl =
## trainControl(method = "repeatedcv", : method = 'lm' is not supported. Using 'qr'

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trControl' will be disregarded
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = sale_price ~ ., data = train, method = "lm", trControl = trainControl(method = "repeate
##     number = 10))
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -89864 -12424    416  12143 140205
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -4.985e+06  3.035e+06  -1.642  0.10076
## gr_liv_area                 2.458e+01  1.393e+01   1.765  0.07778 .
## first_flr_sf                4.252e+01  1.409e+01   3.017  0.00260 **
## second_flr_sf               4.177e+01  1.379e+01   3.029  0.00250 **
## total_bsmt_sf               3.519e+01  2.744e+00  12.827  < 2e-16 ***
## low_qual_fin_sf                    NA         NA      NA       NA
## wood_deck_sf                1.202e+01  4.861e+00   2.474  0.01350 *
## open_porch_sf               1.618e+01  1.004e+01   1.611  0.10736
## bsmt_unf_sf                -2.087e+01  1.723e+00 -12.116  < 2e-16 ***
## mas_vnr_area                1.046e+01  4.229e+00   2.473  0.01353 *
## garage_cars                 4.229e+03  1.893e+03   2.234  0.02563 *
## garage_area                 7.769e+00  6.497e+00   1.196  0.23195
## year_built                  3.251e+02  3.130e+01  10.388  < 2e-16 ***
## tot_rms_abv_grd            -3.838e+03  6.922e+02  -5.545 3.51e-08 ***
## full_bath                  -4.341e+03  1.655e+03  -2.622  0.00883 **
## overall_qualAverage        -5.013e+03  1.735e+03  -2.890  0.00391 **
## overall_qualBelow_Average  -1.280e+04  2.677e+03  -4.782 1.92e-06 ***
## overall_qualExcellent       7.261e+04  5.381e+03  13.494  < 2e-16 ***
## overall_qualFair           -1.115e+04  5.240e+03  -2.127  0.03356 *
## overall_qualGood            1.226e+04  1.950e+03   6.287 4.30e-10 ***
## overall_qualVery_Excellent  1.304e+05  8.803e+03  14.810  < 2e-16 ***
## overall_qualVery_Good       3.798e+04  2.741e+03  13.852  < 2e-16 ***
## kitchen_qualFair           -2.663e+04  6.325e+03  -4.210 2.71e-05 ***
## kitchen_qualGood           -1.879e+04  4.100e+03  -4.582 5.01e-06 ***
## kitchen_qualTypical        -2.677e+04  4.281e+03  -6.252 5.37e-10 ***
## fireplaces                  1.138e+04  2.257e+03   5.043 5.18e-07 ***
## fireplace_quFair           -7.207e+03  6.823e+03  -1.056  0.29106
## fireplace_quGood            6.070e+02  5.833e+03   0.104  0.91713
## fireplace_quNo_Fireplace    3.394e+03  6.298e+03   0.539  0.59002
## fireplace_quPoor           -5.185e+03  7.399e+03  -0.701  0.48362
## fireplace_quTypical        -6.398e+03  5.897e+03  -1.085  0.27814
## exter_qualFair             -3.854e+04  8.383e+03  -4.598 4.66e-06 ***
## exter_qualGood             -1.994e+04  5.585e+03  -3.569  0.00037 ***
## exter_qualTypical          -2.436e+04  5.874e+03  -4.147 3.57e-05 ***
## lot_frontage                1.024e+02  1.905e+01   5.376 8.90e-08 ***
## lot_area                    6.042e-01  7.864e-02   7.683 2.91e-14 ***
## longitude                  -3.481e+04  2.537e+04  -1.372  0.17016
## latitude                    5.874e+04  3.483e+04   1.686  0.09193 .
## misc_val                    9.171e-01  1.003e+00   0.914  0.36071
## year_sold                  -6.455e+02  4.606e+02  -1.401  0.16132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22190 on 1401 degrees of freedom
## Multiple R-squared:  0.9116, Adjusted R-squared:  0.9092
## F-statistic: 380.3 on 38 and 1401 DF,  p-value: < 2.2e-16
```

```r
pred.lm <- predict(lm.fit, newdata = test)
```

```
## Warning in predict.lm(lm.fit, newdata = test): prediction from a rank-deficient
## fit may be misleading
```

```r
lm_mse = RMSE(pred.lm, test$sale_price); lm_mse
```

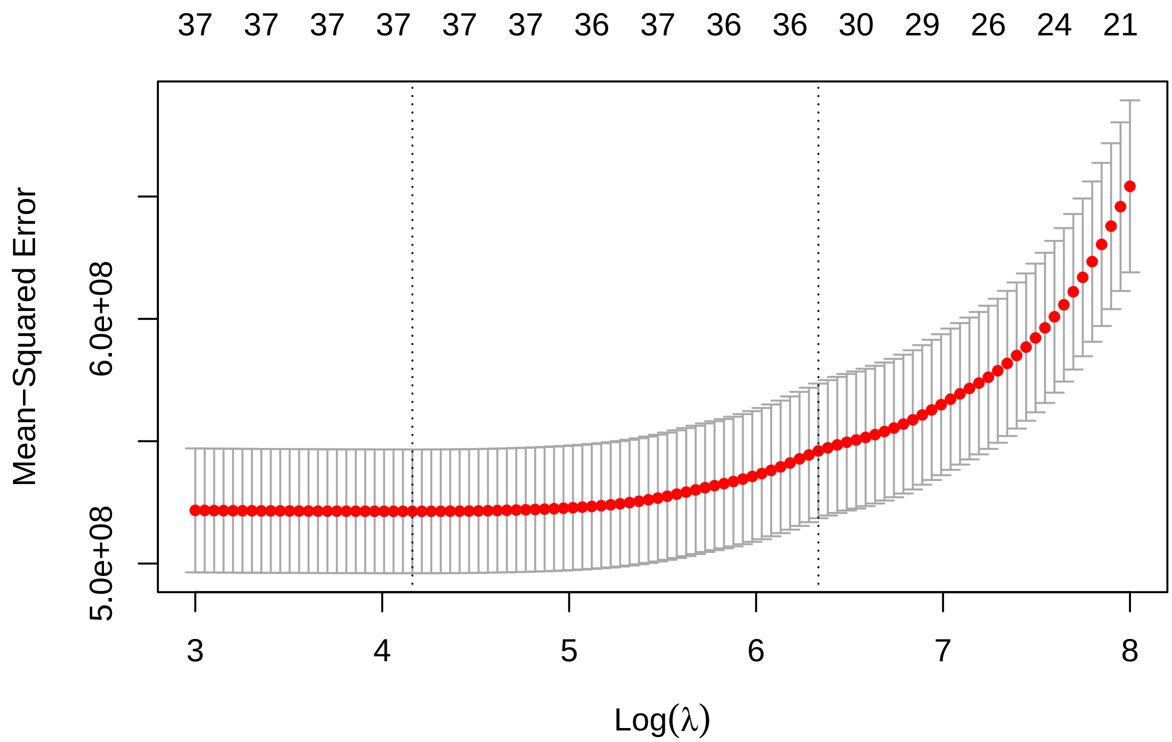
```
## [1] 21149.18
```

The test MSE for the least square method is 2.1149176×10^4. Potential disadvantage of the linear model: 1. There may be too many predictors, which could cause problems such as corlinearity among predictors, large variance; 2. The model is too complex and there exist over-fitting problems.

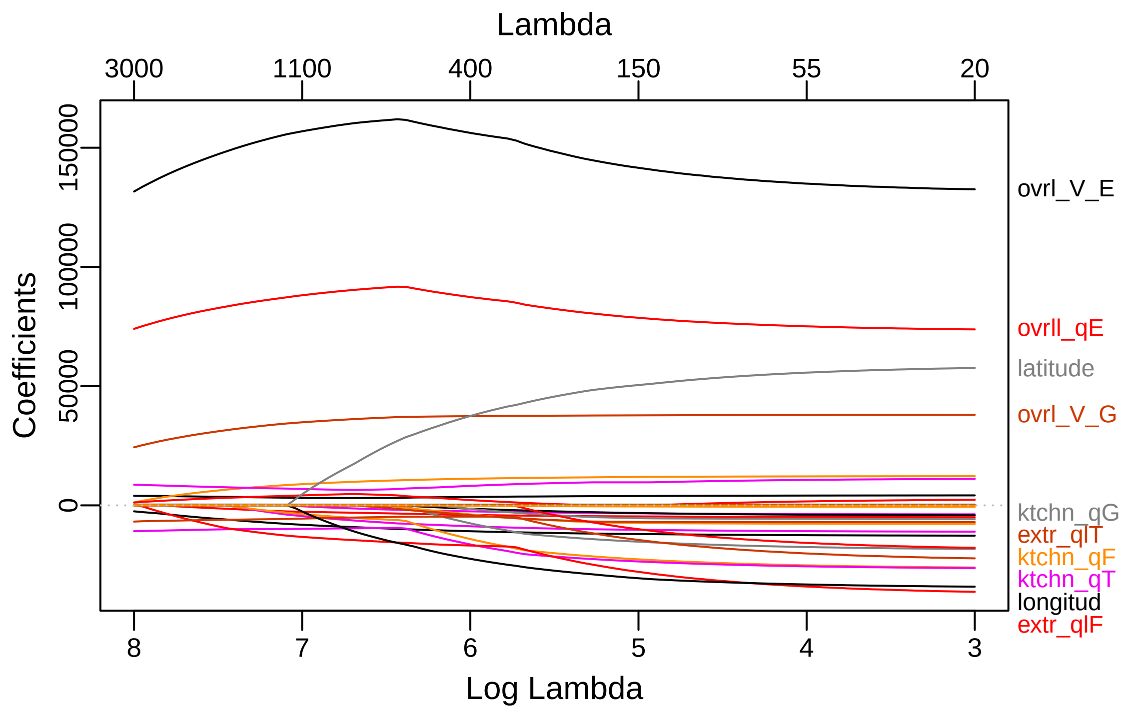
Lasso model

using glmnet

```r
set.seed(2022)

lasso.fit <- cv.glmnet(
  x = x_train,
  y = y_train,
  alpha = 1,
  lambda = exp(seq(8, 3, length = 100))
)

plot(lasso.fit)
```




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




```r
# Look at the 1SE coefficient for lasso
predict(lasso.fit, s = "lambda.1se",type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                lambda.1se
## (Intercept)                 -3.411186e+06
## gr_liv_area                  5.916147e+01
## first_flr_sf                 9.908879e-01
## second_flr_sf                .
## total_bsmt_sf                3.660286e+01
## low_qual_fin_sf             -3.110661e+01
## wood_deck_sf                 9.211416e+00
## open_porch_sf                1.040501e+01
## bsmt_unf_sf                 -2.036685e+01
## mas_vnr_area                 1.368374e+01
## garage_cars                  3.296289e+03
## garage_area                  1.014693e+01
## year_built                   3.116010e+02
## tot_rms_abv_grd             -2.051953e+03
## full_bath                   -3.535970e+02
## overall_qualAverage         -3.577405e+03
## overall_qualBelow_Average   -1.014004e+04
## overall_qualExcellent        9.096805e+04
## overall_qualFair            -7.842522e+03
## overall_qualGood             1.067898e+04
## overall_qualVery_Excellent   1.608334e+05
```

```
## overall_qualVery_Good       3.716056e+04
## kitchen_qualFair           -7.690510e+03
## kitchen_qualGood           -1.468331e+03
## kitchen_qualTypical        -1.076409e+04
## fireplaces                  7.195489e+03
## fireplace_quFair           -1.473565e+03
## fireplace_quGood            3.625170e+03
## fireplace_quNo_Fireplace    .
## fireplace_quPoor            .
## fireplace_quTypical        -2.188237e+03
## exter_qualFair             -1.590797e+04
## exter_qualGood              .
## exter_qualTypical          -4.741677e+03
## lot_frontage                8.043093e+01
## lot_area                    5.822890e-01
## longitude                  -1.729013e+04
## latitude                    2.980870e+04
## misc_val                    2.214766e-02
## year_sold                  -1.300149e+01
```

When the 1SE rule is applied, we can see there are 35 predictors included in the model.

```
y_pred <- predict(lasso.fit, newx = x_test, s = "lambda.min", type = "response")
lasso_mse <- mean(RMSE(y_pred, y_test)^2); lasso_mse
```

```
## [1] 440064267
```

The test MSE for Lasso model (1SE) is 4.4006427×10^8.

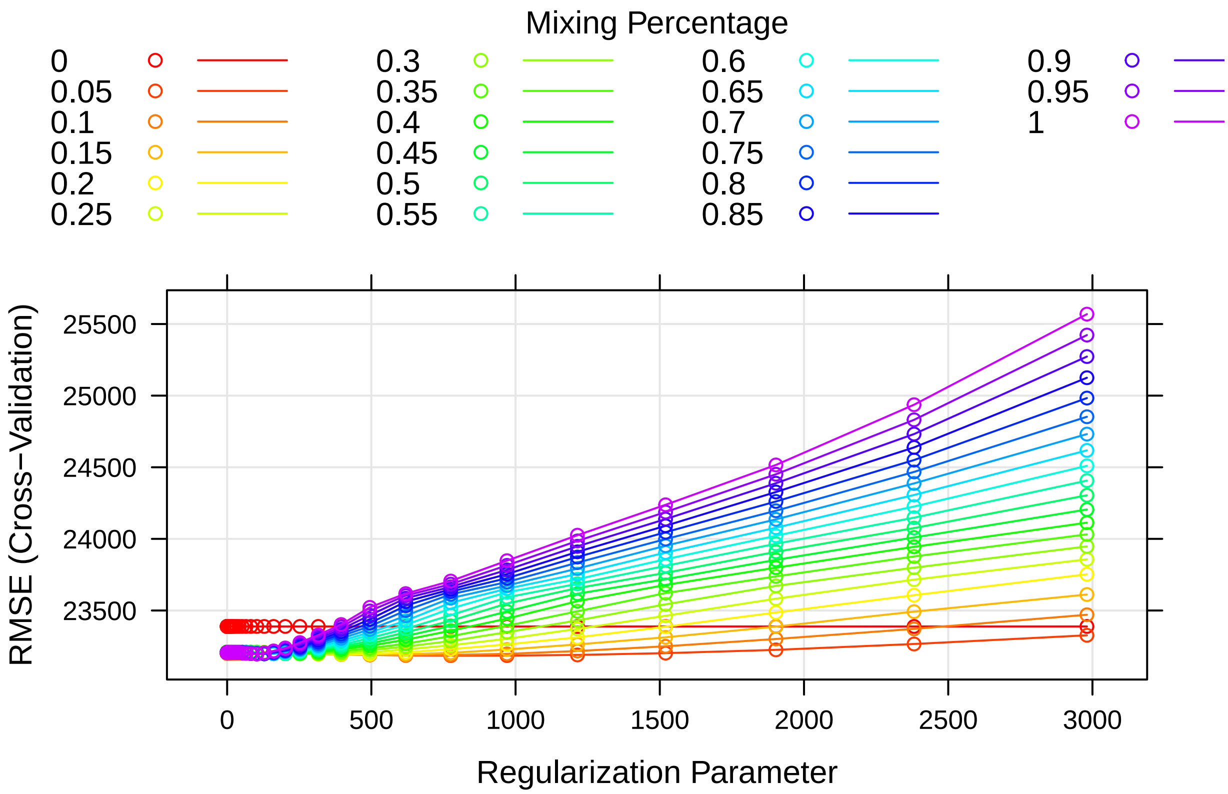
Elastic Net model

```
set.seed(2022)

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 = "cv", number = 10))

enet.fit$bestTune
```

```
##    alpha   lambda
## 95  0.05 970.2473
```

The selected tuning parameter lambda = 970.247332, alpha = 0.05. Then we visualize the elastic net result.

```
# Visualization
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)                -5.172951e+06
## gr_liv_area                 3.812434e+01
## first_flr_sf                2.637357e+01
## second_flr_sf               2.480921e+01
## total_bsmt_sf               3.473579e+01
## low_qual_fin_sf            -1.604315e+01
## wood_deck_sf                1.252592e+01
## open_porch_sf               1.727662e+01
## bsmt_unf_sf                -2.057836e+01
## mas_vnr_area                1.235856e+01
## garage_cars                 3.958861e+03
## garage_area                 9.528576e+00
## year_built                  3.152748e+02
## tot_rms_abv_grd            -3.197583e+03
## full_bath                  -3.313259e+03
## overall_qualAverage        -5.160203e+03
## overall_qualBelow_Average  -1.263204e+04
## overall_qualExcellent       7.695123e+04
## overall_qualFair           -1.165139e+04
## overall_qualGood            1.174884e+04
## overall_qualVery_Excellent  1.386447e+05
```

```
## overall_qualVery_Good       3.733755e+04
## kitchen_qualFair           -2.220863e+04
## kitchen_qualGood           -1.475407e+04
## kitchen_qualTypical        -2.285583e+04
## fireplaces                  1.053606e+04
## fireplace_quFair           -8.000449e+03
## fireplace_quGood            6.186069e+01
## fireplace_quNo_Fireplace    1.117703e+03
## fireplace_quPoor           -5.936463e+03
## fireplace_quTypical        -7.046712e+03
## exter_qualFair             -3.077995e+04
## exter_qualGood             -1.239462e+04
## exter_qualTypical          -1.717937e+04
## lot_frontage                9.877631e+01
## lot_area                    6.017072e-01
## longitude                  -3.534682e+04
## latitude                    5.708440e+04
## misc_val                    8.336667e-01
## year_sold                  -5.374119e+02
```

Next, we calculate the test MSE on our test dataset.

```
# Elastic net test MSE
enet_pred <- predict(enet.fit, newdata = x_test)
enet_mse <- mean(RMSE(enet_pred, y_test)^2); enet_mse
```

```
## [1] 434592441
```

The test error for Elastics net model is 4.3459244×10^8.

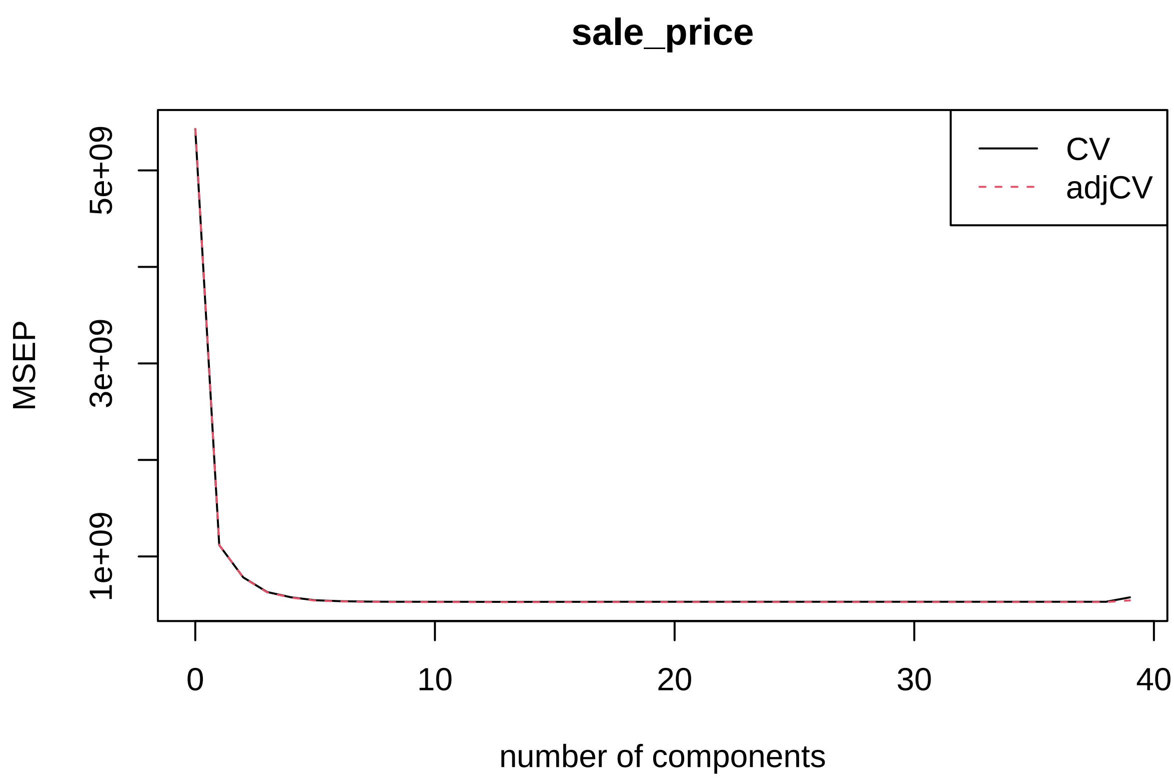
Partial Least Square model

```
set.seed(2022)

pls.fit <- plsr(sale_price~.,
                data = train,
                scale = TRUE,
                validation = "CV")

summary(pls.fit)
```

```
## 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    33432    27986    25125    24011    23369    23171
```

```
## adjCV         73685    33426    27949    25054    23942    23303    23113
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23078    23036    23033     23027     23027     23014     23009
## adjCV    23022    22982    22977     22969     22967     22955     22950
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        23013     23020     23021     23025     23030     23027     23032
## adjCV     22954     22960     22961     22965     22969     22966     22971
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        23032     23032     23032     23032     23032     23032     23034
## adjCV     22971     22971     22971     22971     22971     22971     22973
##        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     22973     22973     22973     22973     22973
##        35 comps  36 comps  37 comps  38 comps  39 comps
## CV        23034     23034     23034     23034     24005
## adjCV     22973     22973     22973     22973     23330
##
## 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.fit, val.type="MSEP", legendpos = "topright")
```




```r
# Calculate the number of component in the model
cv.mse <- RMSEP(pls.fit)
ncomp.cv <- which.min(cv.mse$val[1,,]) - 1; ncomp.cv
```

```
## 13 comps
##       13
```

```r
pls_pred <- predict(pls.fit, newdata = x_test, ncomp = ncomp.cv)
pls_mse <- mean(RMSE(y_test, pls_pred)^2); pls_mse
```

```
## [1] 448737340
```

There are 13 component in pls model, and the test error (MSE) is 4.4873734×10^8.

Comparing different models

```r
name <- c("lm", "lasso 1se", "elastic net", "pls")
MSE <- c(lm_mse, lasso_mse, enet_mse, pls_mse)
comparison <- cbind(name, MSE)
comparison <- as.data.frame(comparison)
```

Now, let's compare the test MSE of the above 4 models, as we have mentioned before, linear model may have many disadvantages, so here though he test MSE for linear model is the lowest, we may tend to choose other models. Therefore, we may choose the model with the lowest MSE besides the linear model: elastic net model.