P8106_HW1_yh3554

Yi Huang

2023-02-22

Data Science II Homework 1

In this exercise, we predict the sale price of a house using its other characteristics. The training data are in "housing train.csv", and the test data are in "housing test.csv". The response is in the column "Sale price". Among the 25 feature variables, some are numeric features, such as living area square feet or first floor square feet, and some are categorical features, such as the overall material and finish of the house or kitchen quality. A detailed description of the variables is in "dictionary.txt".

load data

```r
train_dat <- read.csv("housing_training.csv")
train_dat <- na.omit(train_dat)
test_dat <- read.csv("housing_test.csv")
test_dat <- na.omit(test_dat)

train_x <- model.matrix(Sale_Price~., data = train_dat)[,-1]
train_y <- train_dat$Sale_Price

test_x <- model.matrix(Sale_Price~., data = test_dat)[,-1]
test_y <- test_dat$Sale_Price
```

(a) Fit a linear model using least squares on the training data.

```r
##seed
set.seed(123)

##resampling  method
ctrl5 <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

##fit linear model on training data
linear_model <- train(Sale_Price~.,
               data = train_dat,
               method = "lm",
               trControl = ctrl5)

##evaluate the model on test data
pred_lm <- predict(linear_model, newdata = test_dat)
##calculate MSE
mean((pred_lm - test_y)^2)
```

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

```
##calculate RMSE
sqrt(mean((pred_lm - test_y)^2))
```

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

```
RMSE(pred_lm, test_dat$Sale_Price)
```

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

(b) 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?

```

set.seed(123)
cv.lasso <- cv.glmnet(train_x, train_y,
                      alpha = 1,
                      lambda = exp(seq(-1, 5, length = 100)))
cv.lasso$lambda.min
```

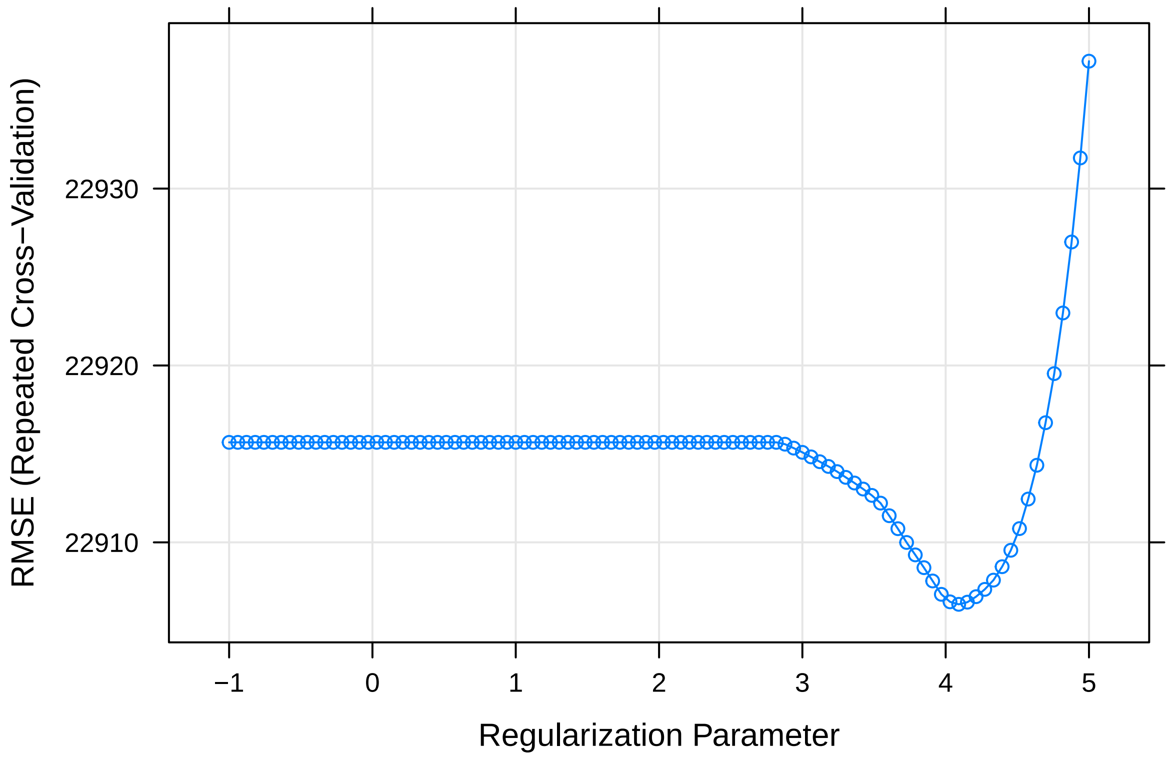
fit a lasso model (use tune parameter that minimize mean cross-validated error)

```
## [1] 59.79423
```

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

```
## [1] 148.4132
```

```
lasso_fit <- train(train_x, train_y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(-1, 5, length = 100))),
                   trControl = ctrl5)
plot(lasso_fit, xTrans = log)
```



```
lasso_fit$bestTune
```

best tune

```
##    alpha   lambda
## 85     1 59.79423
```

```
lasso_pred <- predict(lasso_fit, newdata = test_x)

##calculate MSE
mean((lasso_pred - test_y)^2)
```

test error

```
## [1] 440520057
```

```
##calculate RMSE
sqrt(mean((lasso_pred - test_y)^2))
```

```
## [1] 20988.57
```

```
RMSE(lasso_pred, test_dat$Sale_Price)
```

```
## [1] 20988.57
```

The tune parameter is 59.7942, and the mean square test error (MSE) is 440520057, test RMSE is 20988.57.

```
set.seed(123)
##resampling method
ctrl6 <- trainControl(method = "repeatedcv", selectionFunction = "oneSE", number = 10, repeats = 10)
lasso_fit_1se <- train(train_x, train_y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(-1, 5, length = 100))),
                   trControl = ctrl6)
lasso_fit_1se$bestTune
```

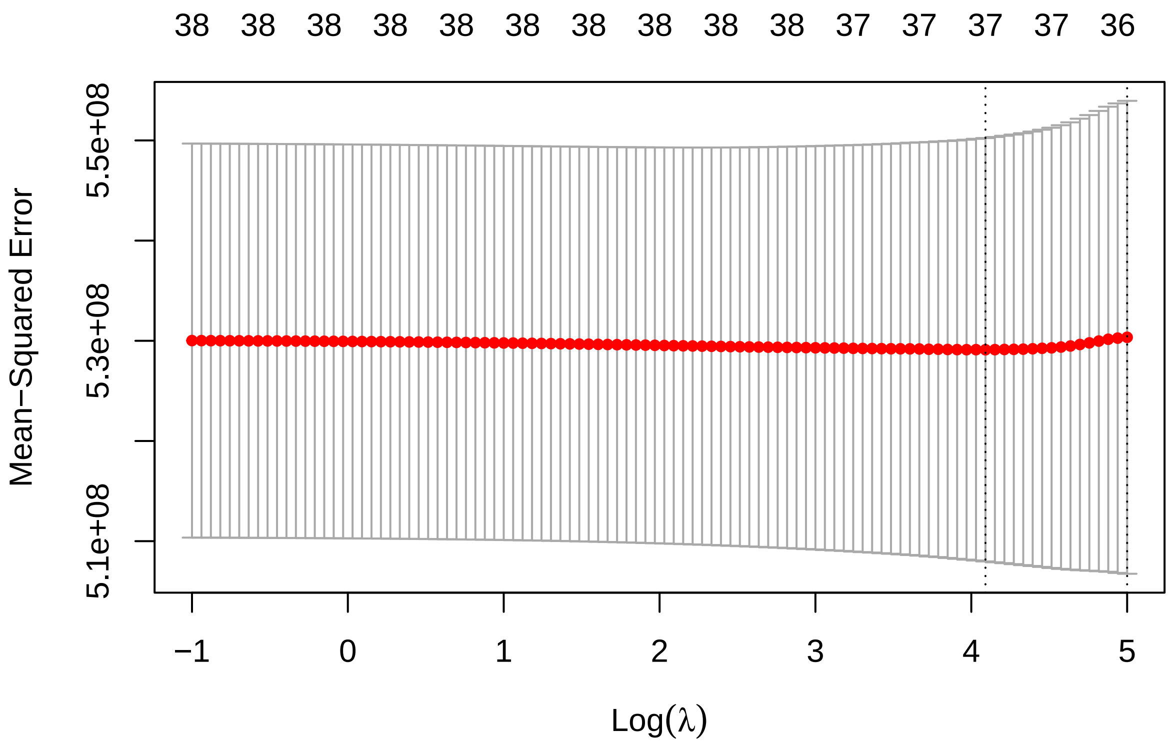
apply 1se rule to lasso

```
##     alpha   lambda
## 100     1 148.4132
```

```
coef(lasso_fit_1se$finalModel, lasso_fit_1se$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -4.622105e+06
## Gr_Liv_Area                 6.430183e+01
## First_Flr_SF                8.502170e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.569211e+01
## Low_Qual_Fin_SF            -3.974325e+01
## Wood_Deck_SF                1.118852e+01
## Open_Porch_SF               1.455212e+01
## Bsmt_Unf_SF                -2.086635e+01
## Mas_Vnr_Area                1.142289e+01
## Garage_Cars                 3.990550e+03
## Garage_Area                 8.403058e+00
## Year_Built                  3.210443e+02
## TotRms_AbvGrd              -3.351443e+03
## Full_Bath                  -3.257826e+03
## Overall_QualAverage        -4.626664e+03
## Overall_QualBelow_Average  -1.200988e+04
## Overall_QualExcellent       7.852120e+04
## Overall_QualFair           -1.020984e+04
## Overall_QualGood            1.188611e+04
## Overall_QualVery_Excellent  1.413225e+05
## Overall_QualVery_Good       3.769571e+04
## Kitchen_QualFair           -2.262179e+04
## Kitchen_QualGood           -1.524818e+04
## Kitchen_QualTypical        -2.350509e+04
## Fireplaces                  9.642085e+03
## Fireplace_QuFair           -7.387701e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    .
## Fireplace_QuPoor           -5.351245e+03
## Fireplace_QuTypical        -6.957962e+03
## Exter_QualFair             -2.806119e+04
## Exter_QualGood             -1.016167e+04
## Exter_QualTypical          -1.467782e+04
## Lot_Frontage                9.629951e+01
## Lot_Area                    6.025532e-01
## Longitude                  -3.061524e+04
## Latitude                    5.043366e+04
## Misc_Val                    7.111446e-01
## Year_Sold                  -4.581742e+02
```

```
plot(cv.lasso)
```



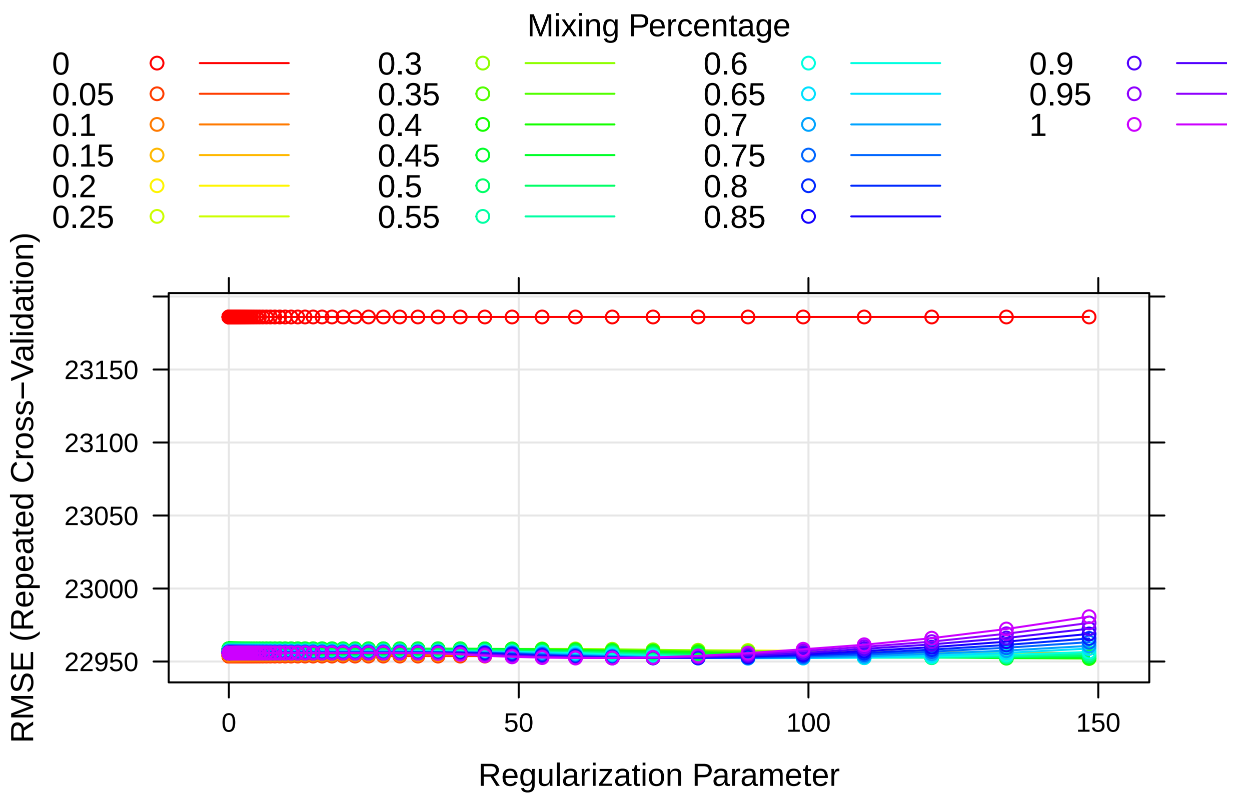
While appying 1se rule, there are 36 predictors in the model. The coefficient of Second_Flr_SF, Fireplace_QuGood, Fireplace_QuNo_Fireplace variables were shrink to zero.

(c) 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?

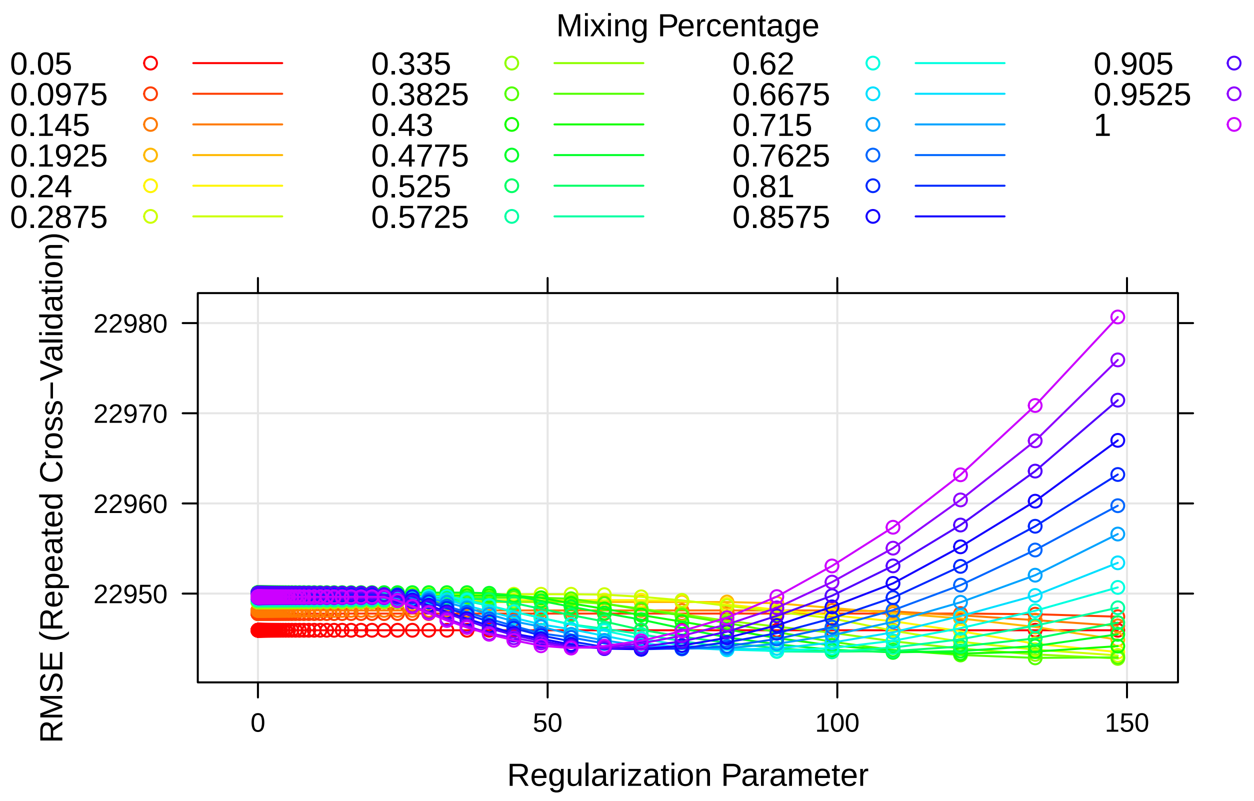
```
set.seed(123)
enet_fit <- train(train_x, train_y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(5,-5, length = 100))),
                  trControl = ctrl5)

myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
plot(enet_fit, par.settings = myPar)
```

fit elastic new model



```r
## what if chose alpha between 0.05 and 1
enet_fit_2 <- train(train_x, train_y,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = seq(0.05, 1, length = 21),
                                           lambda = exp(seq(5,-5, length = 100))),
                    trControl = ctrl5)
plot(enet_fit_2, par.settings = myPar)
```



```
##best tune
enet_fit$bestTune
```

```
##     alpha   lambda
## 800  0.35 148.4132
```

```
##test error
enet_pred <- predict(enet_fit, newdata = test_x)
mean((enet_pred - test_y)^2)
```

```
## [1] 440311482
```

```
##calculate RMSE
sqrt(mean((enet_pred - test_y)^2))
```

```
## [1] 20983.6
```

```
RMSE(enet_pred, test_dat$Sale_Price)
```

```
## [1] 20983.6
```

The selected tune parameter is 148.4132 with alpha 0.35, and the test MSE is 440311482, test RMSE is 20983.6.

```
set.seed(123)
enet_fit_1se <- train(train_x, train_y,
                      method = "glmnet",
                      tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                             lambda = exp(seq(5,-5, length = 100))),
                      trControl = ctrl6)
```

```
enet_fit_1se$bestTune
```

apply 1se rule

```
##     alpha   lambda
## 200  0.05 148.4132
```

```
coef(enet_fit_1se$finalModel, enet_fit_1se$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -5.065857e+06
## Gr_Liv_Area                 3.926323e+01
## First_Flr_SF                2.693840e+01
## Second_Flr_SF               2.591474e+01
## Total_Bsmt_SF               3.507783e+01
## Low_Qual_Fin_SF            -1.574329e+01
## Wood_Deck_SF                1.218650e+01
## Open_Porch_SF               1.657921e+01
## Bsmt_Unf_SF                -2.082250e+01
## Mas_Vnr_Area                1.112016e+01
## Garage_Cars                 4.116584e+03
## Garage_Area                 8.407979e+00
## Year_Built                  3.219256e+02
## TotRms_AbvGrd              -3.619519e+03
## Full_Bath                  -3.984801e+03
## Overall_QualAverage        -5.086126e+03
## Overall_QualBelow_Average  -1.276292e+04
## Overall_QualExcellent       7.476953e+04
## Overall_QualFair           -1.133964e+04
## Overall_QualGood            1.214139e+04
## Overall_QualVery_Excellent  1.343627e+05
## Overall_QualVery_Good       3.786471e+04
## Kitchen_QualFair           -2.491371e+04
## Kitchen_QualGood           -1.722968e+04
## Kitchen_QualTypical        -2.524691e+04
## Fireplaces                  1.105718e+04
## Fireplace_QuFair           -7.710198e+03
## Fireplace_QuGood            2.394432e+02
## Fireplace_QuNo_Fireplace    2.384611e+03
## Fireplace_QuPoor           -5.663616e+03
## Fireplace_QuTypical        -6.854064e+03
## Exter_QualFair             -3.483776e+04
## Exter_QualGood             -1.639609e+04
## Exter_QualTypical          -2.087543e+04
## Lot_Frontage                1.011915e+02
## Lot_Area                    6.041502e-01
## Longitude                  -3.502344e+04
## Latitude                    5.826334e+04
## Misc_Val                    8.926991e-01
## Year_Sold                  -6.038879e+02
```

```
enet_pred_1se <- predict(enet_fit_1se, newdata = test_x)
RMSE(enet_pred_1se, test_dat$Sale_Price)
```

```
## [1] 21024.29
```

The model with 1SE rule has a smaller alpha value 0.05, but it returns to the same lambda value 148.4143 as the default method. Apply 1se rule cannot reduce the model complexity in elastic net model. Thus it is not necessary to use 1se in elastic net model.

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

```
set.seed(123)
pls_fit <- train(train_x, train_y,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:39),
                 trControl = ctrl5,
                 preProcess = c("center", "scale"))
```

fit partial least squares model

```
pls_pred <- predict(pls_fit, newdata = test_x)
mean((pls_pred - test_y)^2)
```

test error

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

```
##calculate RMSE
sqrt(mean((pls_pred - test_y)^2))
```

```
## [1] 20981.37
```

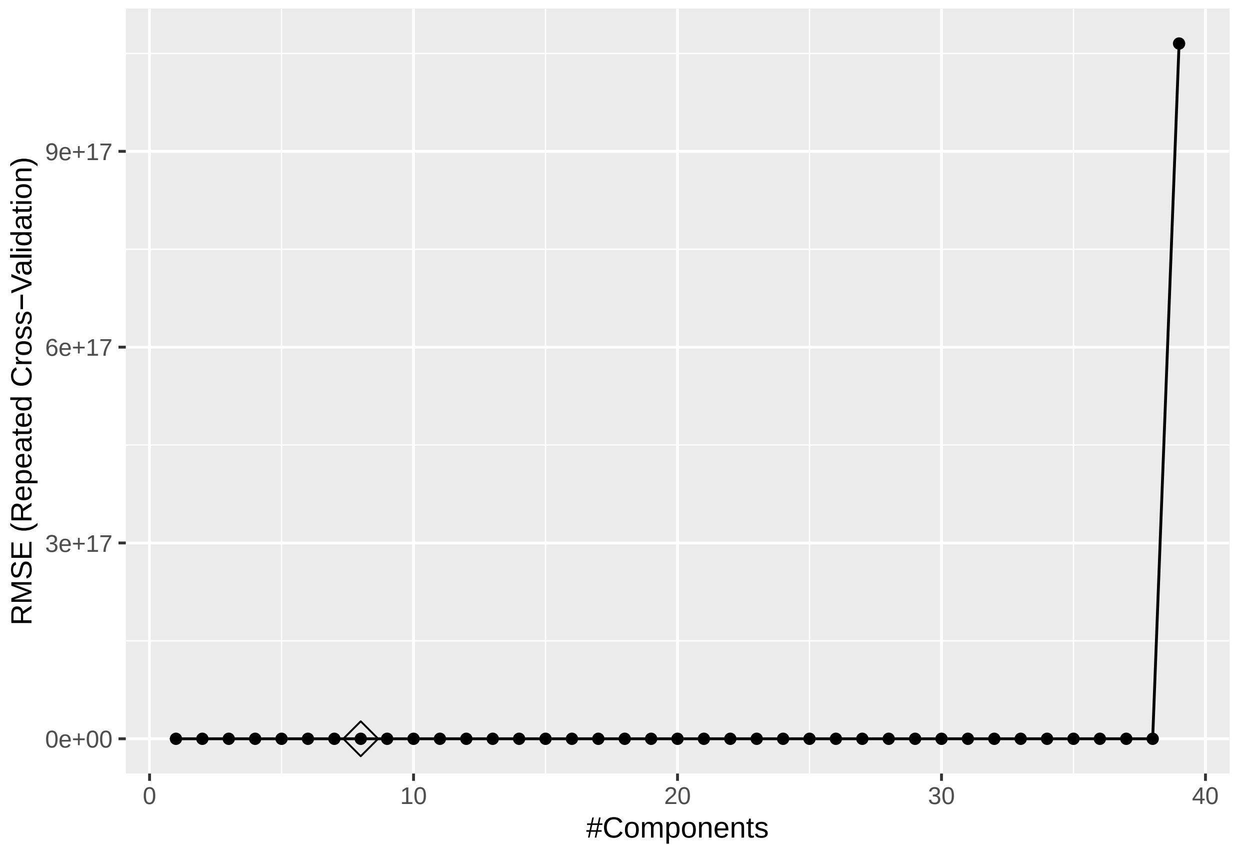
```
RMSE(pls_pred, test_dat$Sale_Price)
```

```
## [1] 20981.37
```

The test MSE is 4.4021794×10^8, test RMSE is 2.098137×10^4.

```
ggplot(pls_fit, highlight = TRUE)
```

plot the number of component in model



The plot shows there are 8 components in partial least squares model.

(e) Which model will you choose for predicting the response? Why?

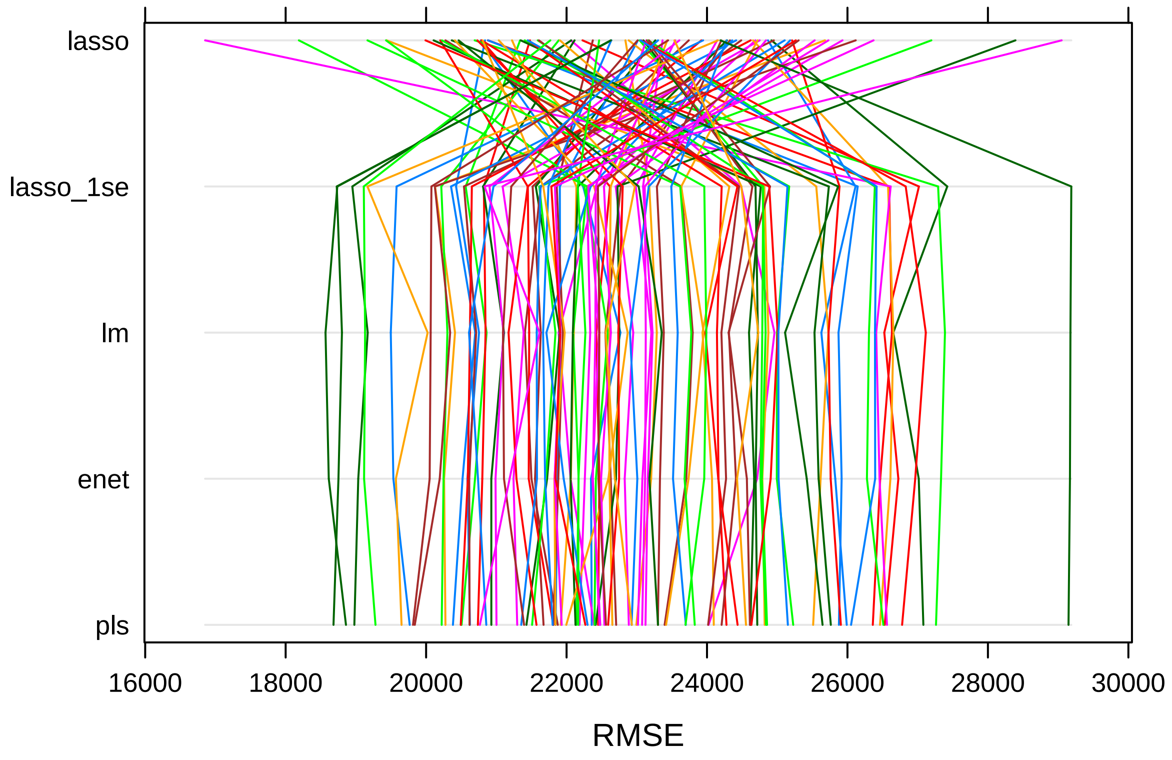
```
set.seed(123)
lm.fit <- train(train_x, train_y,
                method = "lm",
                trControl = ctrl5)
resamp <- resamples(list(lm = linear_model, lasso = lasso_fit, lasso_1se = lasso_fit_1se, enet = enet_fi
summary(resamp)
```

Comparing models

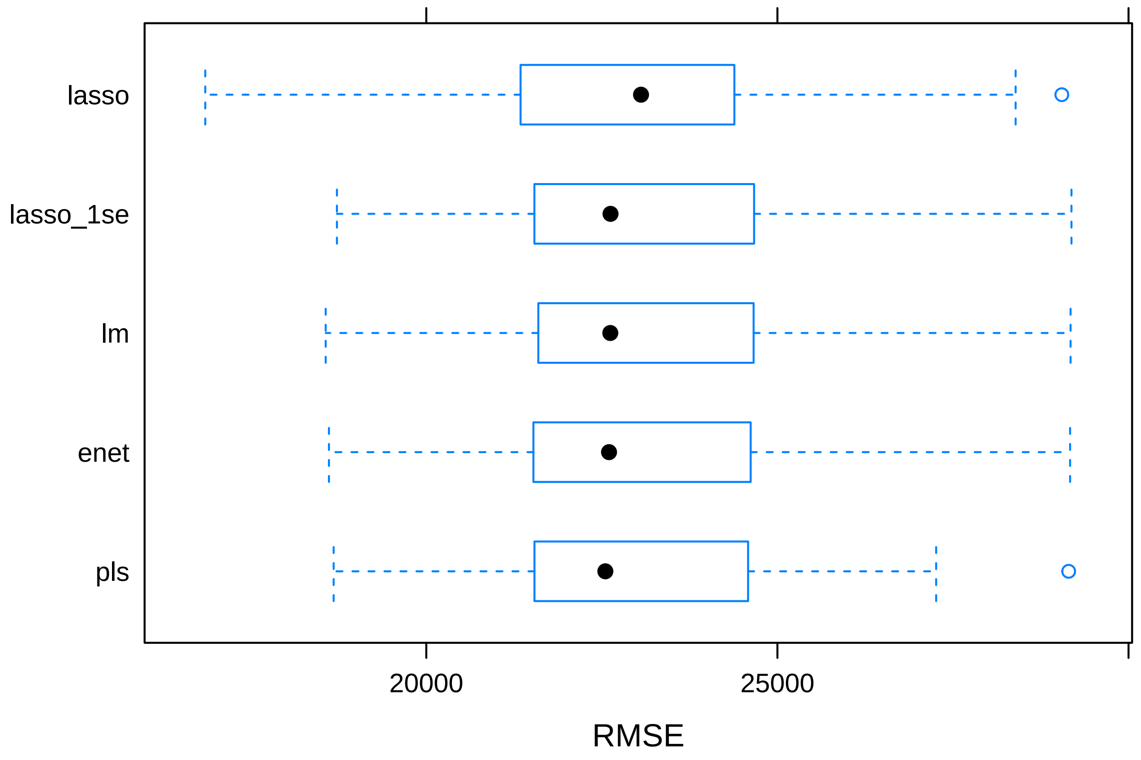
```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, lasso, lasso_1se, enet, pls
## Number of resamples: 100
##
## MAE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm        14016.07 15959.19 16799.05 16743.53 17451.73 19489.31    0
## lasso     12844.44 15748.29 16696.47 16658.32 17488.37 20008.50    0
## lasso_1se 13851.13 15709.07 16734.05 16644.09 17321.24 19563.77    0
## enet      13927.16 15859.48 16711.43 16680.44 17356.25 19479.31    0
## pls       14036.74 15811.72 16664.31 16648.10 17322.17 19488.49    0
```

```
##
## RMSE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm        18567.84 21605.40 22620.82 22991.69 24629.69 29174.78    0
## lasso     16852.83 21344.73 23058.00 22906.50 24373.01 29049.71    0
## lasso_1se 18728.17 21550.79 22623.76 22980.72 24656.84 29187.43    0
## enet      18614.31 21538.50 22602.58 22952.10 24593.42 29167.79    0
## pls       18681.29 21556.19 22550.20 22946.46 24569.02 29147.37    0
##
## Rsquared
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm        0.8573682 0.8931153 0.9036388 0.9029942 0.9140089 0.9367806    0
## lasso     0.8497985 0.8878451 0.9035938 0.9027507 0.9175094 0.9362876    0
## lasso_1se 0.8567461 0.8929416 0.9023858 0.9030708 0.9146153 0.9354656    0
## enet      0.8574253 0.8937000 0.9034213 0.9032979 0.9144807 0.9363132    0
## pls       0.8563520 0.8932821 0.9032911 0.9032911 0.9145114 0.9373245    0
```

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



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



The best model for predicting the sale price of a house is the lasso model since it has the lowest mean value of RMSE comparing to all other models.