p8106\_hw1\_yt2785

Yijing Tao yt2785

library(tidyverse)  
library(ISLR)  
library(glmnet)  
library(caret)  
library(corrplot)  
library(plotmo)  
library(FNN) # knn.reg()  
library(doBy) # which.minn()  
library(pls)

train\_df = read\_csv("./housing\_training.csv") %>%   
 data.frame() %>%   
 na.omit()  
train\_matrix <- model.matrix(Sale\_Price~ ., train\_df)[ ,-1]   
  
test\_df = read\_csv("./housing\_test.csv")  
 data.frame() %>%   
 na.omit()

## data frame with 0 columns and 0 rows

test\_matrix <- model.matrix(Sale\_Price~ ., test\_df)[ ,-1]   
   
# matrix of predictors (glmnet uses input matrix)  
x <- train\_matrix  
# vector of response  
y <- train\_df$Sale\_Price  
  
ctrl1 <- trainControl(method = "cv", selectionFunction = "oneSE")  
ctrl2 <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

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

lm.fit <- lm(Sale\_Price ~ ., data = train\_df)  
summary(lm.fit)

##   
## Call:  
## lm(formula = Sale\_Price ~ ., data = train\_df)  
##   
## 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 \*\*\*  
## TotRms\_AbvGrd -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

lm.pred <- predict(lm.fit, newdata = test\_df)

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

# test error  
test\_error\_lm = mean((lm.pred - test\_df$Sale\_Price)^2)  
test\_error\_lm

## [1] 447287652

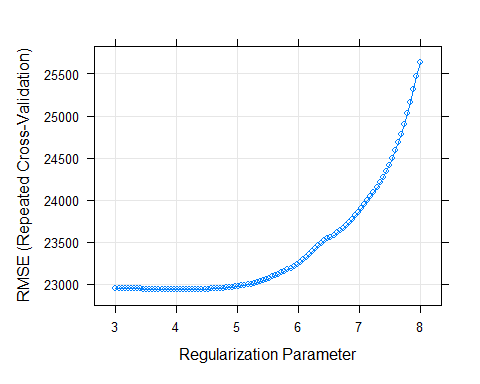
**The test error of the LS linear model is 4.4728765^{8}.** **The potential disadvantage of this model might be the correlation between different predictors.Although the adjusted R square is high in this model, there might be over fitting in this model. And the p-value of this model is less than 0.05, which means this model is not reliable.**

## b) Fit a lasso model on the training data and report the test error. When the 1SE rule is applied, how many predictors are included in the model?

# fit the minSE lasso model  
set.seed(1)  
lasso.fit.minse <- train(x, y,  
 method = "glmnet",  
 tuneGrid = expand.grid(alpha = 1,   
 lambda = exp(seq(8, 3, length=100))),  
 trControl = ctrl2)  
lasso.fit.minse$bestTune

## alpha lambda  
## 23 1 61.01447

plot(lasso.fit.minse, xTrans = log)



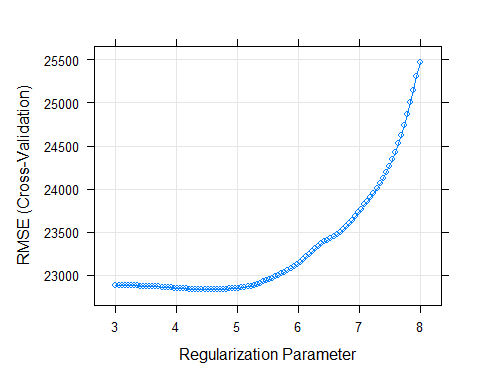
lasso.pred <- predict(lasso.fit.minse, newdata = test\_matrix)  
# test error  
test\_error\_lasso\_minse = mean((lasso.pred - test\_df$Sale\_Price)^2)  
test\_error\_lasso\_minse

## [1] 440396463

# apply the 1SE rule  
set.seed(1)  
lasso.fit.1se <- train(x, y,  
 method = "glmnet",  
 tuneGrid = expand.grid(alpha = 1,   
 lambda = exp(seq(8, 3, length=100))),  
 trControl = ctrl1)  
lasso.fit.1se$bestTune

## alpha lambda  
## 76 1 887.03

plot(lasso.fit.1se, xTrans = log)



precictor\_num = sum((coef(lasso.fit.1se$finalModel, lasso.fit.1se$bestTune$lambda)) != 0)  
precictor\_num

## [1] 30

lasso.pred <- predict(lasso.fit.1se, newdata = test\_matrix)  
# test error  
test\_error\_lasso\_1se = mean((lasso.pred - test\_df$Sale\_Price)^2)  
test\_error\_lasso\_1se

## [1] 421684149

**From the first plot we can decide that lambda should be set between exp(8) to exp(3), 30 predictors are included in the model when 1SE rule is applied. The test error of the minse lasso model is 4.4039646^{8} and the The test error of the 1se lasso model is 4.2168415^{8}**

## c) Fit an elastic net model on the training data. Report the selected tuning parameters and the test error.

set.seed(1)  
enet.fit <- train(x, y,  
 method = "glmnet",  
 tuneGrid = expand.grid(alpha = seq(0, 1, length = 20),   
 lambda = exp(seq(8, 4, length = 60))),  
 trControl = ctrl2)  
enet.alpha.best = enet.fit$bestTune$alpha  
enet.alpha.best

## [1] 0.05263158

enet.lambda.best = enet.fit$bestTune$lambda  
enet.lambda.best

## [1] 585.7431

enet.pred <- predict(enet.fit, newdata = test\_matrix)  
# test error  
test\_error\_enet = mean((enet.pred - test\_df$Sale\_Price)^2)  
test\_error\_enet

## [1] 438429127

**The selected best parameter is alpha = 0.0526316 and lambda = 585.7431342. The test error is 4.3842913^{8}.**

## 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(1)  
pls.mod <- plsr(Sale\_Price ~ .,   
 data = train\_df,   
 scale = TRUE,   
 validation = "CV")  
  
cv.mse <- RMSEP(pls.mod)  
ncomp.cv <- which.min(cv.mse$val[1,,])-1  
ncomp.cv

## 8 comps   
## 8

predy2.pls <- predict(pls.mod, newdata = test\_df,   
 ncomp = ncomp.cv)  
# test MSE  
test\_error\_pls = mean((predy2.pls - test\_df$Sale\_Price)^2)  
test\_error\_pls

## [1] 440217938

**8 components are included in the model. The test error of this model is 4.4021794^{8}.**

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

mse <- c(test\_error\_lasso\_minse, test\_error\_lasso\_1se, test\_error\_enet, test\_error\_pls)  
name <- c("Lasso(minse)", "Lasso(1se)", "Elastic", "PLS")  
MSE\_df <- cbind(name, mse)  
colnames(MSE\_df) <- c("Model", "MSE")  
MSE\_df <- as.data.frame(MSE\_df)  
MSE\_df

## Model MSE  
## 1 Lasso(minse) 440396462.693672  
## 2 Lasso(1se) 421684148.75126  
## 3 Elastic 438429126.893179  
## 4 PLS 440217937.923444

which.min(MSE\_df$MSE)

## [1] 2

**From question a) we can consider that the LS linear model is not reliable, so I decided not to include it in the comparision. I will choose the lasso model which applied 1SE since it’s MSE is the smallest among these models, which means it has the highest accuracy and efficiency.**