Homework 1

Yiying Wu (yw3996)

R packages

```
library(tidyverse)
library(caret)
library(tidymodels)
```

Input dataset

```
housing_train<-read_csv("./data/housing_training.csv")
housing_train <- na.omit(housing_train)
housing_test<-read_csv("./data/housing_test.csv")
housing_test <- na.omit(housing_test)
```

Response: Sale price

(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?

```
ctrl1 <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 5,
                      selectionFunction = "oneSE")

ctrl2 <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 5,
                      selectionFunction = "best")
```

Using the minimal MSE rule

```
set.seed(123)
lasso.fit2 <- train(Sale_Price ~ .,
                    data = housing_train,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                                           lambda = exp(seq(10, 0, length = 200))),
                    trControl = ctrl2)
# plot(lasso.fit1, xTrans = log)
```

Here's the selected tuning parameter when the minimal MSE rule is applied

```
lasso.fit2$bestTune
```

```
##    alpha   lambda
## 82     1 58.57756
```

The best tuning parameter is 58.578

And the test error is

```
lasso.pred2 <- predict(lasso.fit2, newdata = housing_test)
# test error
mean((lasso.pred2 - housing_test$Sale_Price)^2)
```

```
## [1] 440643616
```

MSE=4.4064362×10^8

Using 1SE rule

```
set.seed(123)
lasso.fit1 <- train(Sale_Price ~ .,
                    data = housing_train,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                                           lambda = exp(seq(10, 0, length = 200))),
                    trControl = ctrl1)
# plot(lasso.fit1, xTrans = log)
```

Here's the selected tuning parameter when 1SE rule is applied

```
lasso.fit1$bestTune
```

```
##     alpha   lambda
## 120     1 395.4006
```

The best tuning parameter is 395.401

And the test error is

```
lasso.pred1 <- predict(lasso.fit1, newdata = housing_test)
# test error
mean((lasso.pred1 - housing_test$Sale_Price)^2)
```

```
## [1] 420908683
```

MSE=4.2090868×10^8

coefficients in the final model are

```r
# coefficients in the final model
coef(lasso.fit1$finalModel, lasso.fit1$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -3.943320e+06
## Gr_Liv_Area                 6.108042e+01
## First_Flr_SF                9.449776e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.625960e+01
## Low_Qual_Fin_SF            -3.544037e+01
## Wood_Deck_SF                1.004731e+01
## Open_Porch_SF               1.212972e+01
## Bsmt_Unf_SF                -2.060695e+01
## Mas_Vnr_Area                1.293708e+01
## Garage_Cars                 3.503707e+03
## Garage_Area                 9.712008e+00
## Year_Built                  3.152703e+02
## TotRms_AbvGrd              -2.541567e+03
## Full_Bath                  -1.468739e+03
## Overall_QualAverage        -4.028594e+03
## Overall_QualBelow_Average  -1.088189e+04
## Overall_QualExcellent       8.701792e+04
## Overall_QualFair           -8.812746e+03
## Overall_QualGood            1.113730e+04
## Overall_QualVery_Excellent  1.557589e+05
## Overall_QualVery_Good       3.731925e+04
## Kitchen_QualFair           -1.452346e+04
## Kitchen_QualGood           -7.941694e+03
## Kitchen_QualTypical        -1.672150e+04
## Fireplaces                  8.250960e+03
## Fireplace_QuFair           -3.941525e+03
## Fireplace_QuGood            2.111744e+03
## Fireplace_QuNo_Fireplace    .
## Fireplace_QuPoor           -1.621116e+03
## Fireplace_QuTypical        -4.235985e+03
## Exter_QualFair             -1.699617e+04
## Exter_QualGood              .
## Exter_QualTypical          -4.789842e+03
## Lot_Frontage                8.694715e+01
## Lot_Area                    5.920111e-01
## Longitude                  -2.270837e+04
## Latitude                    3.807574e+04
## Misc_Val                    3.235947e-01
## Year_Sold                  -1.732221e+02
```

Therefore, there are 36 predictors 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.

Using the minimal MSE rule

```
set.seed(123)
enet.fit2 <- train(Sale_Price ~ .,
                   data = housing_train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                          lambda = exp(seq(10, 0, length = 200))),
                   trControl = ctrl2)
```

Here's the selected tuning parameter

```
enet.fit2$bestTune
```

```
##     alpha   lambda
## 327  0.05 562.0879
```

The best tuning parameter is 562.088

And the test error is

```
enet.pred2 <- predict(enet.fit2, newdata = housing_test)
# test error
mean((enet.pred2 - housing_test$Sale_Price)^2)
```

```
## [1] 438824460
```

MSE=4.3882446×10^8

Using the 1SE rule

```
set.seed(123)
enet.fit1 <- train(Sale_Price ~ .,
                   data = housing_train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                          lambda = exp(seq(10, 0, length = 200))),
                   trControl = ctrl1)
```

Here's the selected tuning parameter

```
enet.fit1$bestTune
```

```
##     alpha   lambda
## 173     0 5671.541
```

The best tuning parameter is 5671.541

And the test error is

```r
enet.pred1 <- predict(enet.fit1, newdata = housing_test)
# test error
mean((enet.pred1 - housing_test$Sale_Price)^2)
```

```
## [1] 426371024
```

MSE=4.2637102×10^8

Given the substantial difference in lambda values between the minimal MSE and the 1SE rule in the results, it suggests that the simpler model under the 1SE rule is significantly more regularized. Given that the 1SE rule led to a model with lower MSE on the test data, it would be reasonable to favor this approach for selecting tuning parameters in the elastic net model.

Also, the change from alpha = 0.05 to alpha = 0 under the 1SE rule indicates a shift from a slight Lasso preference towards a pure Ridge regression approach. In this way, all predictors are kept in the model, leading to models that may be less sparse but can handle multicollinearity better.

(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
# training data
x <- model.matrix(Sale_Price ~ ., housing_train)[, -1]
y <- housing_train$Sale_Price

# test data
x2 <- model.matrix(Sale_Price ~ .,housing_test)[, -1]
y2 <- housing_test$Sale_Price

set.seed(123)
pls.fit <- train(x, y,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:19),
                 trControl = ctrl2,
                 preProcess = c("center", "scale"))
```

the test error is

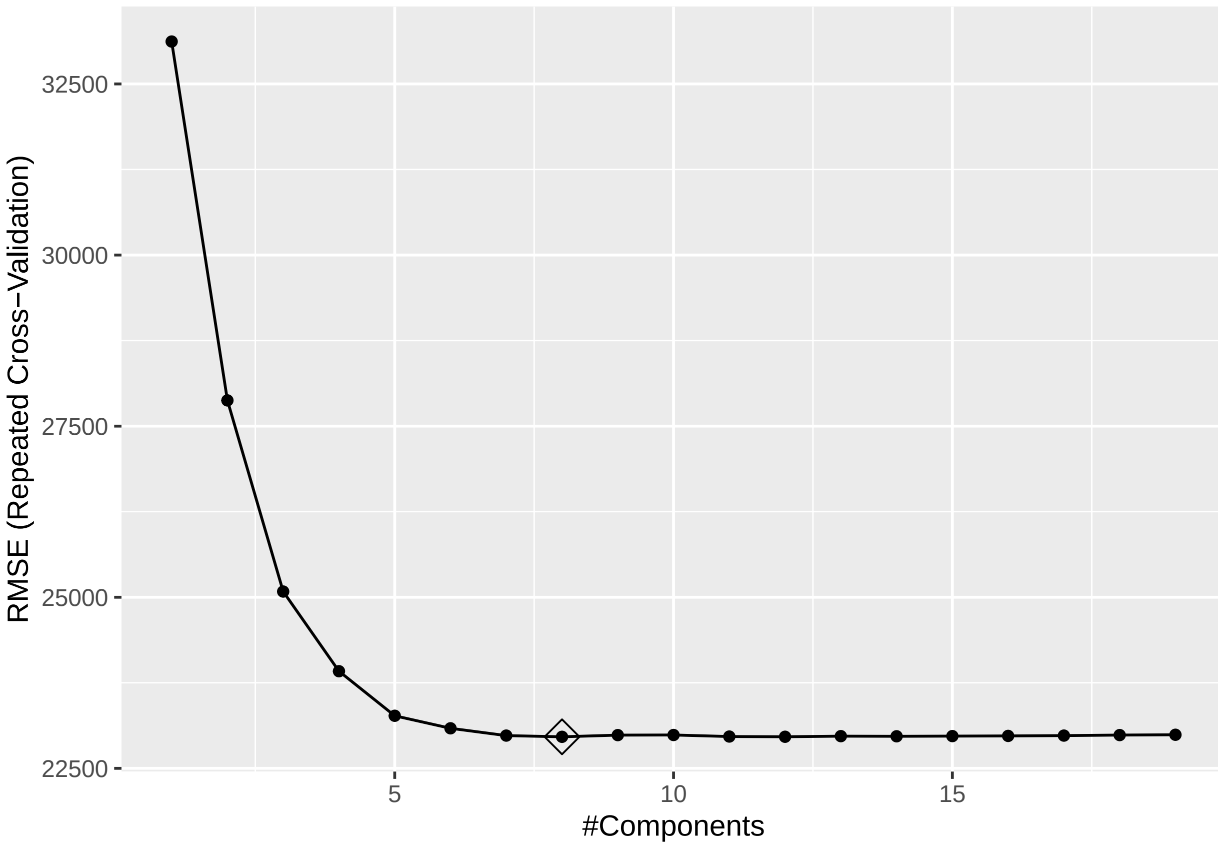
```r
predy2.pls2 <- predict(pls.fit, newdata = x2)
mean((y2 - predy2.pls2)^2)
```

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

MSE=4.4021794×10^8

Check the number of components included in the model

```r
ggplot(pls.fit, highlight = TRUE)
```



8 components are included in the model.

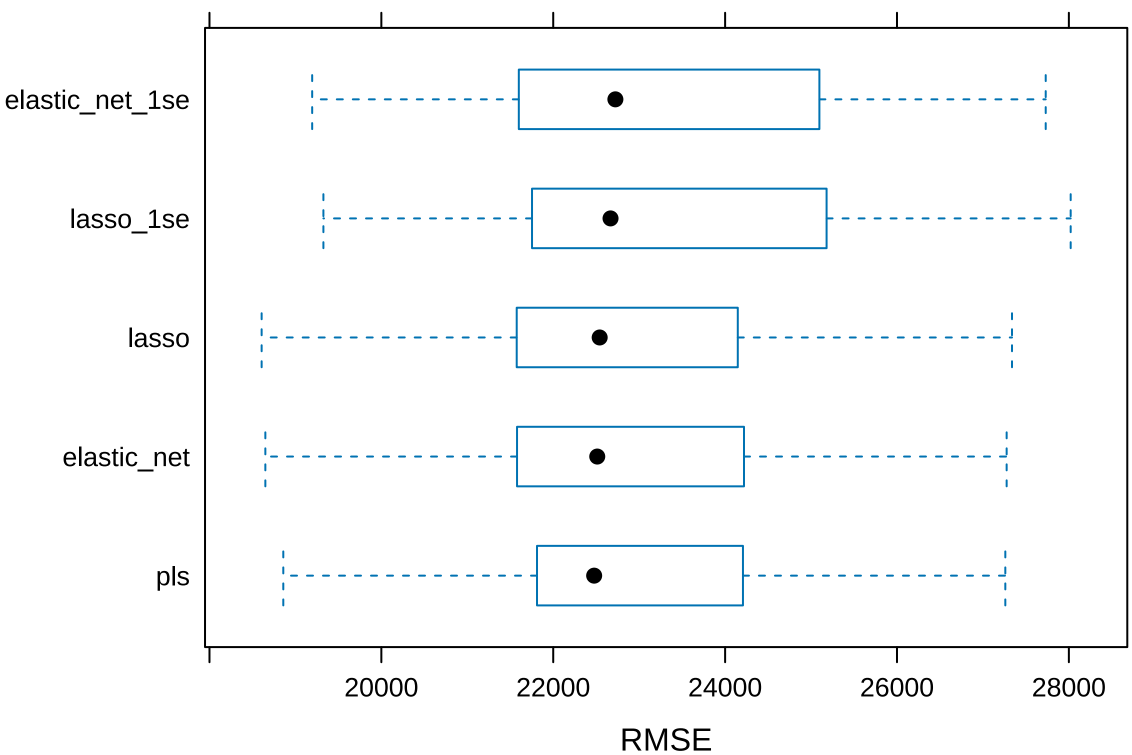
(d) Choose the best model for predicting the response and explain your choice.

```
resamp <- resamples(list(elastic_net = enet.fit2,
                         elastic_net_1se = enet.fit1,
                         lasso = lasso.fit2,
                         lasso_1se = lasso.fit1,
                         pls = pls.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: elastic_net, elastic_net_1se, lasso, lasso_1se, pls
## Number of resamples: 50
##
## MAE
##                     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## elastic_net     14139.08 15842.43 16688.92 16679.95 17298.91 19413.78    0
## elastic_net_1se 14328.64 15703.63 16703.03 16628.77 17157.20 19344.88    0
## lasso           14148.50 15881.60 16718.67 16712.47 17329.40 19493.28    0
## lasso_1se       14555.23 15695.82 16695.34 16680.84 17273.33 19756.29    0
## pls             14211.64 15851.87 16705.92 16675.42 17288.62 19488.49    0
```

```
##
## RMSE
##                      Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## elastic_net     18650.02 21596.91 22512.32 22954.43 24181.04 27276.03    0
## elastic_net_1se 19194.90 21654.89 22723.36 23226.85 24960.91 27730.55    0
## lasso           18607.14 21609.69 22540.55 22961.66 24129.11 27338.43    0
## lasso_1se       19325.73 21770.06 22666.74 23235.86 25161.51 28020.97    0
## pls             18859.10 21818.88 22476.36 22960.64 24179.95 27260.53    0
##
## Rsquared
##                       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## elastic_net     0.8626286 0.8948433 0.9015491 0.9035863 0.9160159 0.9344627
## elastic_net_1se 0.8589321 0.8916914 0.9020915 0.9016454 0.9157528 0.9321098
## lasso           0.8629714 0.8952234 0.9014908 0.9035391 0.9154400 0.9345749
## lasso_1se       0.8595795 0.8903466 0.9001323 0.9013075 0.9163390 0.9323004
## pls             0.8630125 0.8943839 0.9023356 0.9034479 0.9174745 0.9344308
##                 NA's
## elastic_net        0
## elastic_net_1se    0
## lasso              0
## lasso_1se          0
## pls                0
```

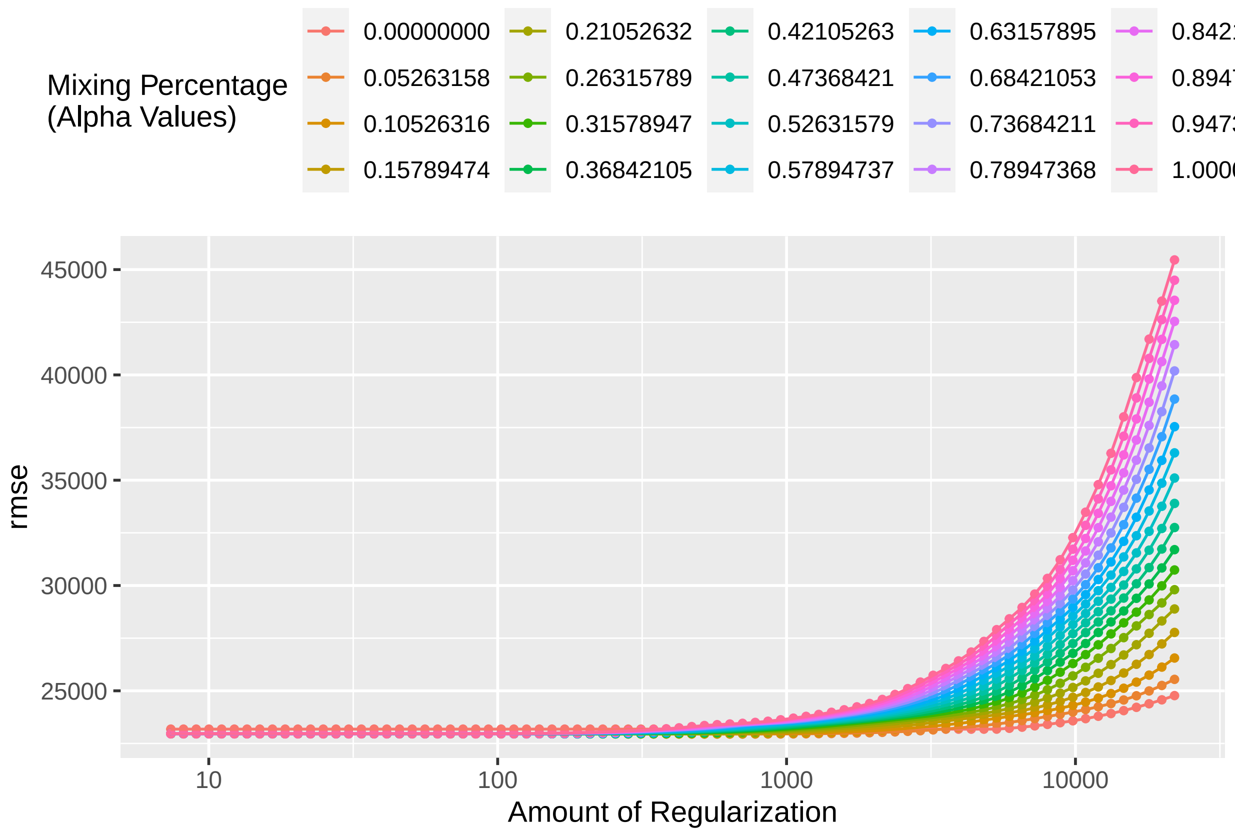
```
bwplot(resamp, metric = "RMSE")
```



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

(e) If "caret" was used for the elastic net in (b), retrain this model with "tidymodels", 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.

```
set.seed(123)
cv_folds <- vfold_cv(housing_train, v = 10)

enet_spec <- linear_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet") %>%
  set_mode("regression")

# enet_spec %>% extract_parameter_dials("mixture")
# enet_spec %>% extract_parameter_dials("penalty")

enet_grid_set <- parameters(penalty(range = c(2, 10), trans = log_trans()),
                            mixture(range = c(0, 1)))
enet_grid <- grid_regular(enet_grid_set, levels = c(80, 20))



enet_workflow <- workflow() %>%
  add_model(enet_spec) %>%
  add_formula(Sale_Price ~ .)

enet_tune <- tune_grid(
  enet_workflow,
  resamples = cv_folds,
  grid = enet_grid
)

autoplot(enet_tune, metric = "rmse") +
  theme(legend.position = "top") +
  labs(color = "Mixing Percentage\n(Alpha Values)")
```




```r
enet_best <- select_best(enet_tune, metric = "rmse")

final_enet_spec <- enet_spec %>%
  update(penalty = enet_best$penalty, mixture = enet_best$mixture)

enet_fit <- fit(final_enet_spec, formula = Sale_Price ~ ., data = housing_train)

# Get coefficients
enet_model <- extract_fit_engine(enet_fit)
coef(enet_model, s = enet_best$penalty)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                     s1
## (Intercept)                -5.116983e+06
## Gr_Liv_Area                 3.876993e+01
## First_Flr_SF                2.652728e+01
## Second_Flr_SF               2.523243e+01
## Total_Bsmt_SF               3.491619e+01
## Low_Qual_Fin_SF            -1.600533e+01
## Wood_Deck_SF                1.234299e+01
## Open_Porch_SF               1.691695e+01
## Bsmt_Unf_SF                -2.070780e+01
## Mas_Vnr_Area                1.176448e+01
## Garage_Cars                 4.028892e+03
## Garage_Area                 8.995813e+00
## Year_Built                  3.185633e+02
## TotRms_AbvGrd              -3.398156e+03
```

```
## Full_Bath                  -3.623807e+03
## Overall_QualAverage        -5.120058e+03
## Overall_QualBelow_Average  -1.268998e+04
## Overall_QualExcellent       7.606774e+04
## Overall_QualFair           -1.149393e+04
## Overall_QualGood            1.194998e+04
## Overall_QualVery_Excellent  1.368634e+05
## Overall_QualVery_Good       3.761727e+04
## Kitchen_QualFair           -2.346627e+04
## Kitchen_QualGood           -1.590470e+04
## Kitchen_QualTypical        -2.396788e+04
## Fireplaces                  1.077384e+04
## Fireplace_QuFair           -7.942480e+03
## Fireplace_QuGood            6.995319e+01
## Fireplace_QuNo_Fireplace    1.626087e+03
## Fireplace_QuPoor           -5.885946e+03
## Fireplace_QuTypical        -7.039470e+03
## Exter_QualFair             -3.247087e+04
## Exter_QualGood             -1.408204e+04
## Exter_QualTypical          -1.870298e+04
## Lot_Frontage                9.990108e+01
## Lot_Area                    6.030866e-01
## Longitude                  -3.512839e+04
## Latitude                    5.757898e+04
## Misc_Val                    8.615076e-01
## Year_Sold                  -5.673446e+02
```

select tuning parameters using tidymodels package

```
enet_best$penalty
```

```
## [1] 636.3166
```

The selected tuning parameters is 636.317, which is different from that is part **(b)**. This maybe because different partitions are used in tidymodels and caret, which likely contributes to discrepancies in the chosen parameters for elastic net models between the two frameworks.