DS2_HW1

Yixiao Sun

2024-02-14

```r
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tibble       3.2.1
## v dplyr        1.1.3     v tidyr        1.3.0
## v infer        1.0.5     v tune         1.1.2
## v modeldata    1.2.0     v workflows    1.1.3
## v parsnip      1.1.1     v workflowsets 1.0.1
## v purrr        1.0.2     v yardstick    1.2.0
## v recipes      1.0.8

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()         masks scales::discard()
## x tidyr::expand()          masks Matrix::expand()
## x dplyr::filter()          masks stats::filter()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x tidyr::pack()            masks Matrix::pack()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## x tidyr::unpack()          masks Matrix::unpack()
## x recipes::update()        masks Matrix::update(), stats::update()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggplot2)
library(plotmo)
```

```
## Loading required package: Formula
```

```
## Loading required package: plotrix
```

```
##
## Attaching package: 'plotrix'
```

```
## The following object is masked from 'package:scales':
##
##     rescale
```

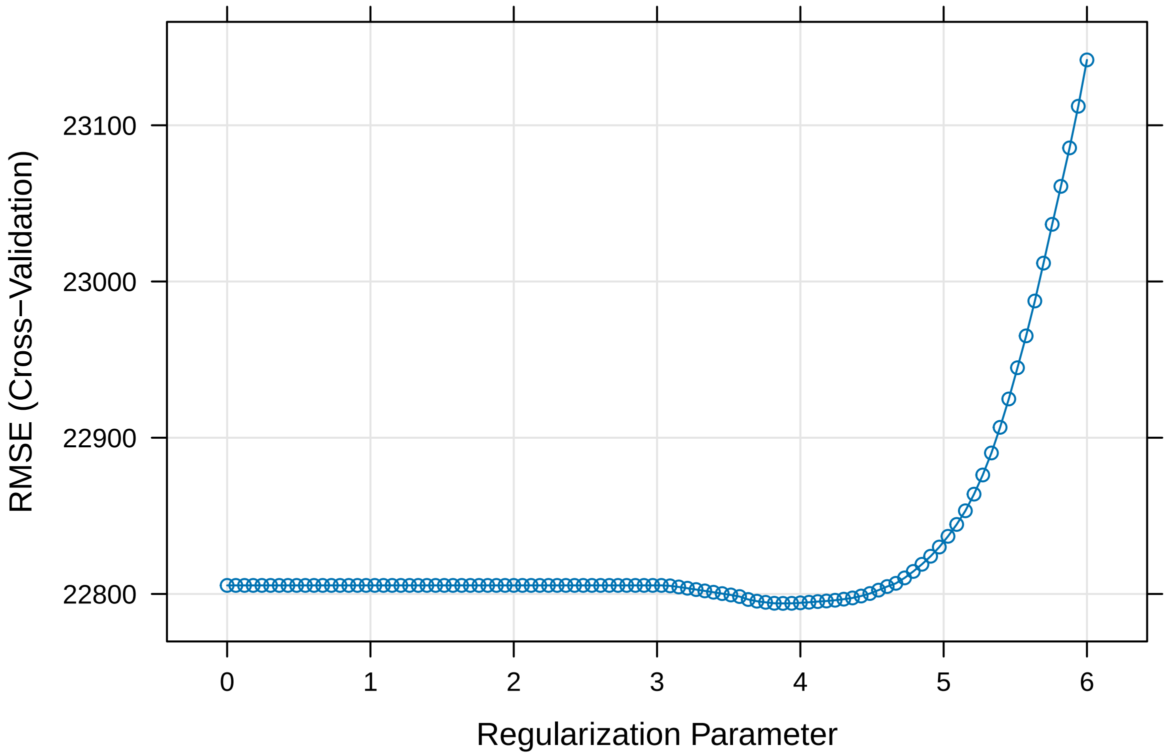
```
## Loading required package: TeachingDemos
```

```
library(ggrepel)
```

```
train_data <- read.csv("~/Desktop/P8106 Data Science 2/ds2_hw1/housing_training.csv")
test_data <- read.csv("~/Desktop/P8106 Data Science 2/ds2_hw1/housing_test.csv")
x<-model.matrix(Sale_Price ~., train_data)[,-26]
y<-train_data[,"Sale_Price"]
```

Lasso Model

```
set.seed(2)
ctrl1 <- trainControl(method = "cv", number = 10)
ctrl2 <- trainControl(method = 'cv', number = 10, selectionFunction = "oneSE")

lasso.fit <- train(Sale_Price ~ .,
                   data = train_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(6, 0, length = 100))),
                   trControl = ctrl1)
plot(lasso.fit, xTrans = log)
```




```r
lasso.fit$bestTune # Best Tune
```

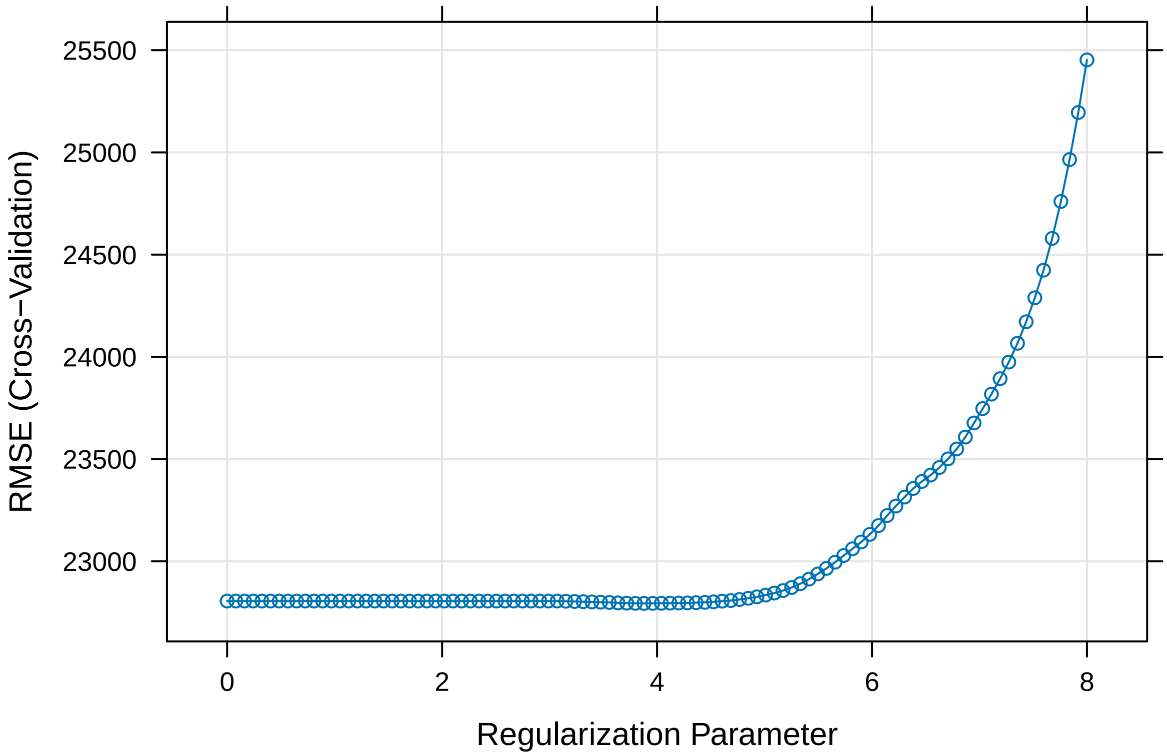
```
##    alpha   lambda
## 65     1 48.36555
```

```r
lasso.pred <- predict(lasso.fit, newdata = test_data)

lasso_mse <- mean((lasso.pred - test_data[,"Sale_Price"])^2) # LASSO MSE
lasso_mse
```

```
## [1] 441688534
```

The selected tuning parameter for the Lasso Model is 48.36555, the test error is 441688534

```r
set.seed(2)
lasso.fit_2 <- train(Sale_Price ~ .,
                   data = train_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,lambda = exp(seq(8, 0, length = 100))),
                   trControl = ctrl2)
plot(lasso.fit_2, xTrans = log)
```




```r
lasso.fit_2$bestTune
```

```
##    alpha   lambda
## 80     1 592.1964
```

```r
coef(lasso.fit_2$finalModel, lasso.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)            -4.868222e+06
## Gr_Liv_Area             6.562231e+01
## First_Flr_SF            7.916290e-01
## Second_Flr_SF           .
## Total_Bsmt_SF           3.536497e+01
## Low_Qual_Fin_SF        -4.118547e+01
## Wood_Deck_SF            1.173330e+01
## Open_Porch_SF           1.563329e+01
## Bsmt_Unf_SF            -2.088957e+01
## Mas_Vnr_Area            1.078734e+01
## Garage_Cars             4.118864e+03
## Garage_Area             8.075432e+00
## Year_Built              3.238250e+02
## TotRms_AbvGrd          -3.672935e+03
## Full_Bath              -3.964421e+03
## Overall_QualAverage    -4.898586e+03
```

```
## Overall_QualBelow_Average  -1.254807e+04
## Overall_QualExcellent       7.484859e+04
## Overall_QualFair           -1.085766e+04
## Overall_QualGood            1.216348e+04
## Overall_QualVery_Excellent  1.344955e+05
## Overall_QualVery_Good       3.792258e+04
## Kitchen_QualFair           -2.529790e+04
## Kitchen_QualGood           -1.761182e+04
## Kitchen_QualTypical        -2.568787e+04
## Fireplaces                  1.075501e+04
## Fireplace_QuFair           -7.712658e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.779347e+03
## Fireplace_QuPoor           -5.685166e+03
## Fireplace_QuTypical        -7.014698e+03
## Exter_QualFair             -3.444815e+04
## Exter_QualGood             -1.612849e+04
## Exter_QualTypical          -2.055399e+04
## Lot_Frontage                1.003421e+02
## Lot_Area                    6.043848e-01
## Longitude                  -3.339367e+04
## Latitude                    5.604964e+04
## Misc_Val                    8.519923e-01
## Year_Sold                  -5.815597e+02
```

```
sum(coef(lasso.fit_2$finalModel, s=lasso.fit_2$bestTune$lambda) != 0) - 1
```

```
## [1] 35
```

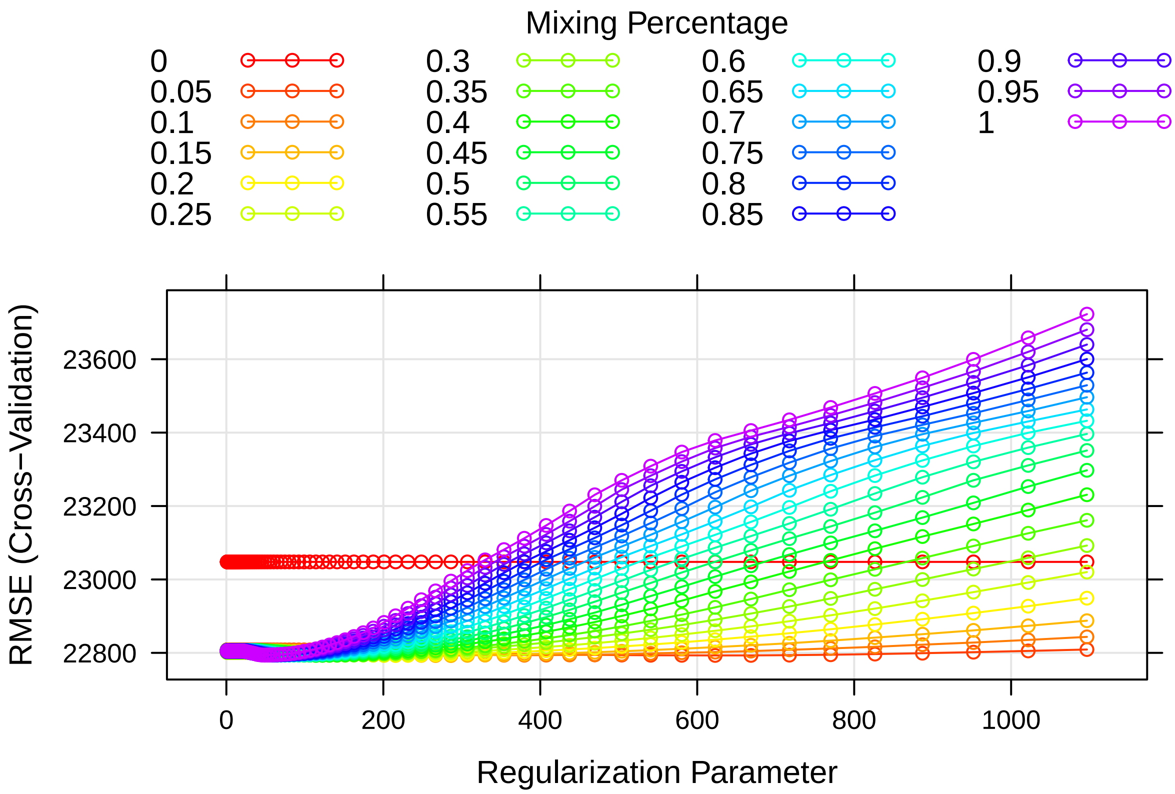
WHen the 1SE rule applied, 35 predictors are inclueded in the model.

Elastic Net

```
set.seed(2)
enet.fit <- train(Sale_Price ~ .,
                  data = train_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(7, 0, length = 100))),
                  trControl = ctrl1)
enet.fit$bestTune # ELastic Net Best Tune
```

```
##     alpha   lambda
## 381  0.15 286.1642
```

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

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




```
enet.predic <- predict(enet.fit, newdata = test_data)
enet.error <- mean((enet.predic - test_data[,"Sale_Price"])^2)
enet.error
```

```
## [1] 439998442
```

```
enet.fit_2 <- train(Sale_Price ~ .,
                  data = train_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(7, 0, length = 100))),
                  trControl = ctrl2)
enet.fit_2$bestTune
```

```
##     alpha   lambda
## 100     0 1096.633
```

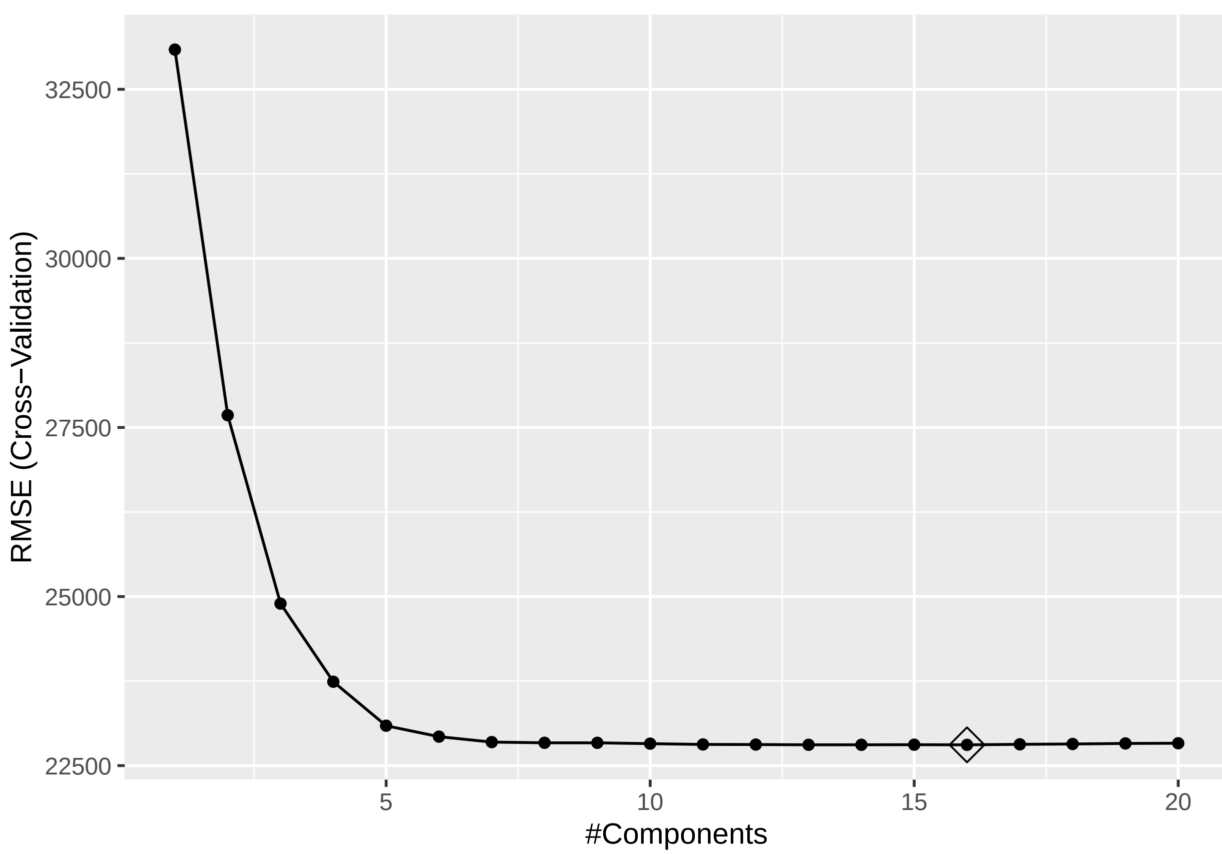
The selected tuning parameter for the elastic net model is 286.1642, the test error is 439998442 The 1SE rule can't be applied to this model, since the best tune given by the 1SE model print out an alpha with 0.

Least Square Model

```r
set.seed(2)
x2 <- model.matrix(Sale_Price ~ .,test_data)[, -26]
y2 <- test_data$Sale_Price
pls.fit <- train(Sale_Price ~.,
                 data = train_data,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:20),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))
predy2.pls2 <- predict(pls.fit, newdata = test_data)

mean((y2 - predy2.pls2)^2) # least square model MSE
```

```
## [1] 446775692
```

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



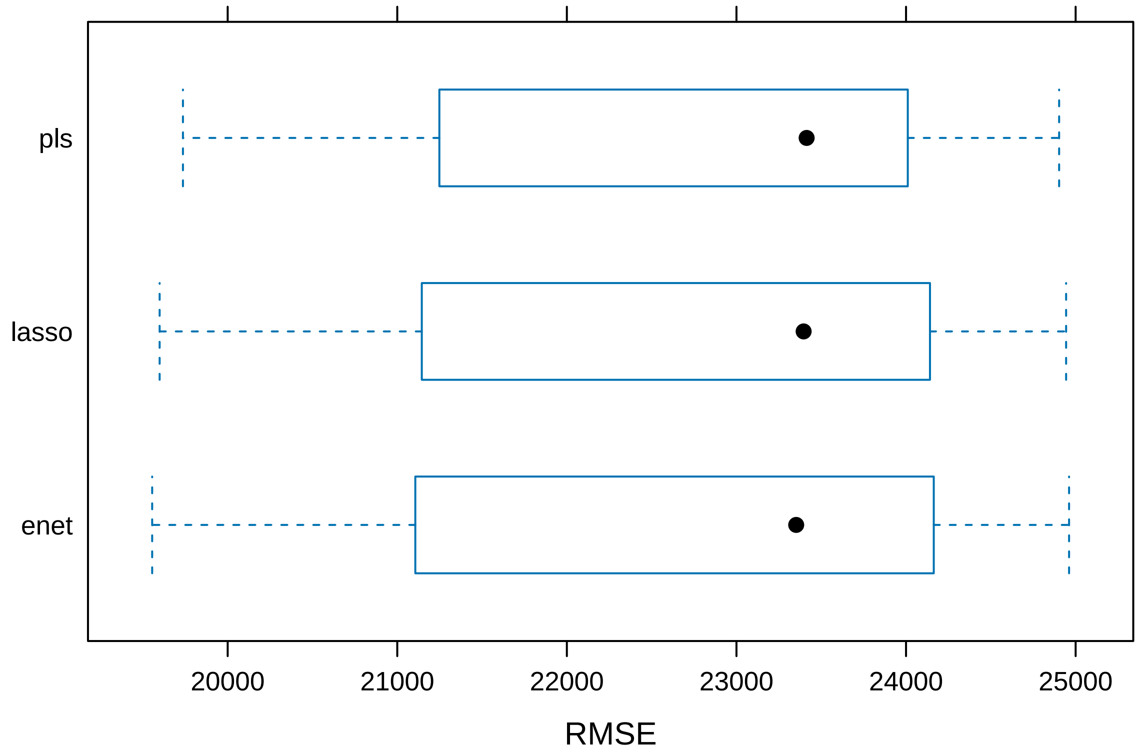
The test error for the least square model is 446775692. 16 Components are included in this model based on the plot.

Model Comparison

```r
set.seed(2)
lm.fit <- train(Sale_Price ~ .,data = train_data, method = "lm", trControl = ctrl1)
resamp <- resamples(list(enet = enet.fit, lasso = lasso.fit, pls = pls.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: enet, lasso, pls
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  14478.62 15705.19 16576.39 16578.81 17526.31 18436.99    0
## lasso 14524.52 15733.97 16600.29 16597.58 17545.81 18444.32    0
## pls   14585.07 15807.81 16639.83 16629.39 17570.18 18442.95    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  19554.72 21551.89 23352.35 22792.47 24093.38 24961.37    0
## lasso 19598.86 21576.32 23396.09 22793.95 24062.41 24943.84    0
## pls   19736.20 21657.28 23413.90 22807.42 23936.71 24902.72    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet  0.8739298 0.8878946 0.9061908 0.9039321 0.9193882 0.9264909    0
## lasso 0.8736728 0.8881479 0.9058904 0.9039037 0.9195208 0.9265658    0
## pls   0.8730979 0.8894494 0.9055091 0.9038628 0.9194429 0.9257294    0
```

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

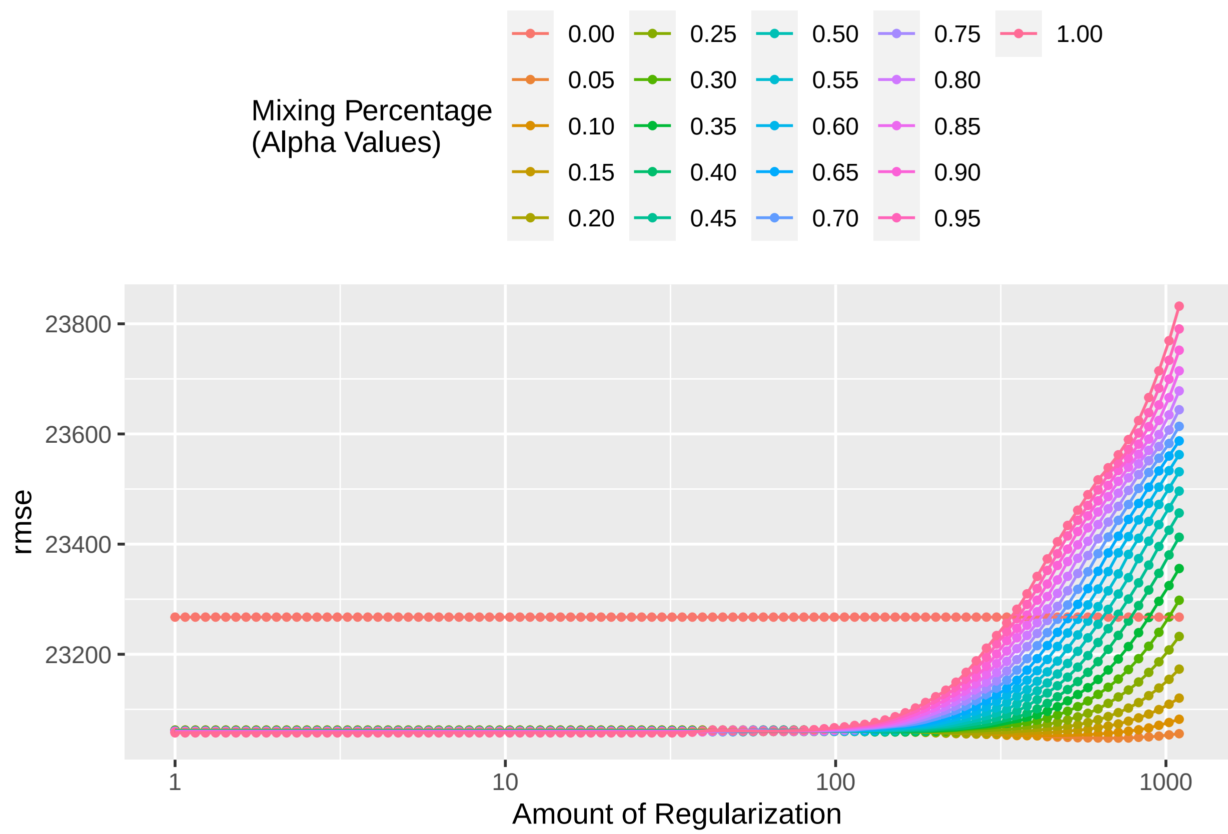


Based on the graph given, the Elastic net model is the best model for predicting the response since it has the lowest RMSE.

Tidymodels

```
set.seed(2)
cv_folds <- vfold_cv(train_data, v = 10)

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

enet_grid_set <- parameters(penalty(range = c(0, 7),
                                    trans = log_trans()),
                            mixture(range = c(0, 1)))

enet_grid <- grid_regular(enet_grid_set, levels = c(100, 21))

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
```




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

enet_best
```

```
## # A tibble: 1 x 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1    669.    0.05 Preprocessor1_Model0193
```

The tuning parameter for the tidymodel is 668.5094. The potential reasons for the differences when chosing parameters can be due to the different ways they grid search which produce the wanted value.