P8106_HW1

Naomi Simon-Kumar ns3782

2/16/2025

Contents

Loading libraries

```
library(ISLR)
library(glmnet)
library(caret)
library(tidymodels)
library(corrplot)
library(ggplot2)
library(plotmo)
library(ggrepel)
```

Question (a): Lasso Model

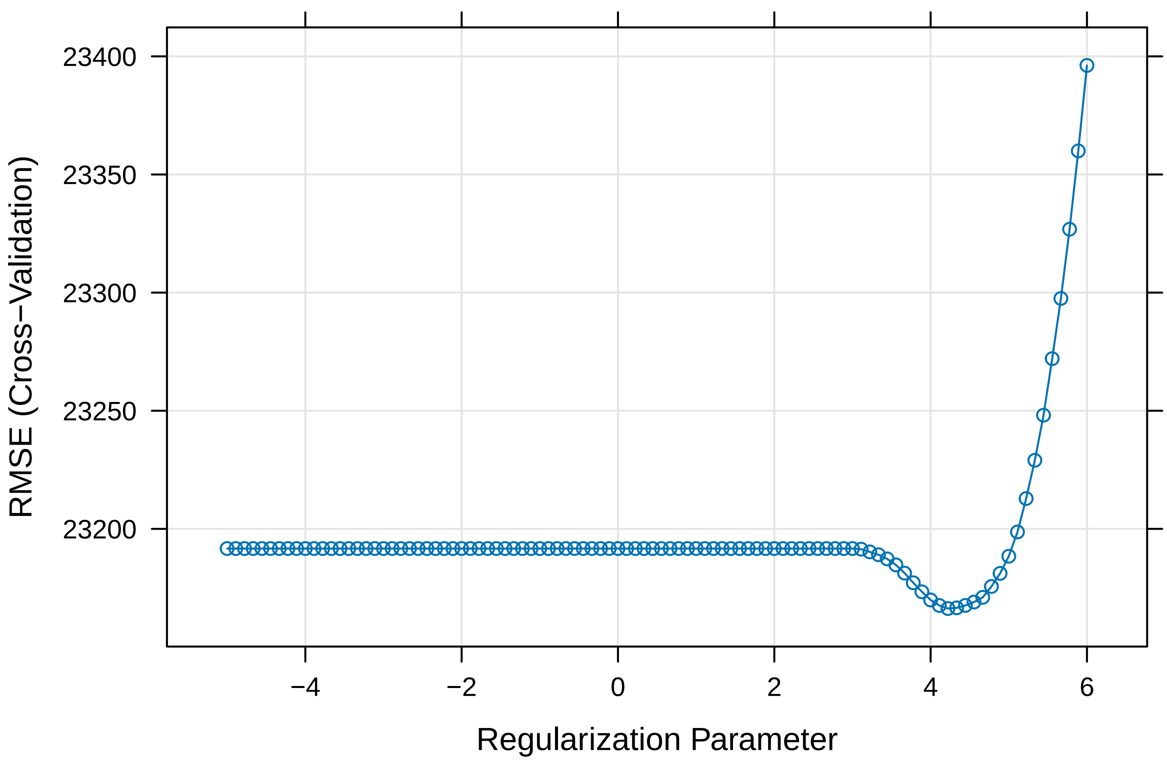
To start, we load the training and testing data and subsequently set a seed for reproducibility.

Next, we initialise 10-fold cross-validation to partition the training data into 10 equal subsets. This allows training the model on 9 folds while validating on the final fold. This ensures we evaluate the performance of the model, while avoiding overfitting.

```
# Load training and testing data

training_data <- read.csv("housing_training.csv")
testing_data <- read.csv("housing_test.csv")

set.seed(29)  # Ensure results are reproducible

# Using 10 fold cross-validation

ctrl1 <- trainControl(method = "cv", number = 10)
```

Next, we proceed to fit a lasso regression model using the training data. Sale_Price is the outcome variable, with all other variables as predictors. The lasso model is tuned over a sequence of 100 lambda values ranging from exp(6) to exp(-5).

```
set.seed(29)  # Ensure results are reproducible

# Fit the Lasso model

lasso.fit  <- train(
  Sale_Price ~ .,
  data = training_data,
  method = "glmnet",
  tuneGrid = expand.grid(alpha = 1,
                         lambda = exp(seq(6, -5, length = 100))),
  trControl = ctrl1
)

# Plot

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



Based on the plot, it appears as though the optimal lambda value is around exp(4), as this is where the RMSE is minimised. Higher lambda values (i.e., greater penalisation) appear to result in poorer model performance, likely due to excessive shrinkage forcing too many coefficients to zero, leading to underfitting.

```r
set.seed(29)  # Ensure results are reproducible

# Find optimal tuning parameter

lasso.fit$bestTune
```

```
##    alpha   lambda
## 84     1 68.18484
```

```r
# Extracting coefficients for each predictor, at the optimal lambda

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

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                     s1
## (Intercept)                -4.820791e+06
## Gr_Liv_Area                 6.534680e+01
## First_Flr_SF                8.043483e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.542591e+01
## Low_Qual_Fin_SF            -4.089879e+01
## Wood_Deck_SF                1.161853e+01
## Open_Porch_SF               1.539927e+01
## Bsmt_Unf_SF                -2.088675e+01
## Mas_Vnr_Area                1.091770e+01
## Garage_Cars                 4.078354e+03
## Garage_Area                 8.182394e+00
## Year_Built                  3.232484e+02
## TotRms_AbvGrd              -3.607362e+03
## Full_Bath                  -3.820746e+03
## Overall_QualAverage        -4.845814e+03
## Overall_QualBelow_Average  -1.244202e+04
## Overall_QualExcellent       7.559703e+04
## Overall_QualFair           -1.073410e+04
## Overall_QualGood            1.211373e+04
## Overall_QualVery_Excellent  1.358907e+05
## Overall_QualVery_Good       3.788544e+04
## Kitchen_QualFair           -2.476713e+04
## Kitchen_QualGood           -1.713660e+04
## Kitchen_QualTypical        -2.525278e+04
## Fireplaces                  1.051146e+04
## Fireplace_QuFair           -7.657866e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.385656e+03
## Fireplace_QuPoor           -5.632703e+03
## Fireplace_QuTypical        -7.010013e+03
## Exter_QualFair             -3.316061e+04
## Exter_QualGood             -1.492745e+04
## Exter_QualTypical          -1.936658e+04
## Lot_Frontage                9.952901e+01
## Lot_Area                    6.042265e-01
## Longitude                  -3.285809e+04
## Latitude                    5.492284e+04
```

```
## Misc_Val                        8.240622e-01
## Year_Sold                      -5.568142e+02
```

Note that at the optimal lambda value, most of the predictors remain in the model. However, some are shrunk to zero (i.e., Second_Flr_SF, Fireplace_QuGood) during the variable selection process, and removed from the model. Therefore, this final model includes **37 predictors**.

```r
set.seed(29)  # Ensure results are reproducible

# Finding RMSE

lasso_preds <- predict(lasso.fit, newdata = testing_data)
rmse <- sqrt(mean((lasso_preds - testing_data$Sale_Price)^2))
print(rmse)
```

```
## [1] 20969.2
```

For the lasso model, the optimal tuning parameter lambda is **68.18484**, representing where RMSE is minimised. The test error (RMSE) at this lambda is **20969.2**.

```r
set.seed(29)  # Ensure results are reproducible

# Using 1se cross-validation.
# Code from: https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/oneSE

ctrl_1se <- trainControl(
  method = "cv",
  selectionFunction = "oneSE"
)

# Fit the lasso model using 1se

lasso_1se_fit <- train(
  Sale_Price ~ .,
  data = training_data,
  method = "glmnet",
  tuneGrid = expand.grid(
    alpha = 1,
    lambda = exp(seq(6, -5, length = 100))
  ),
  trControl = ctrl_1se
)

# Optimal lambda using 1SE

lasso_lambda_1se <- lasso_1se_fit$bestTune$lambda
print(lasso_lambda_1se)
```

```
## [1] 403.4288
```

```r
# Extracting coefficients for each predictor, at the optimal lambda

coef(lasso_1se_fit$finalModel, s = lasso_lambda_1se)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                     s1
## (Intercept)              -3.919159e+06
## Gr_Liv_Area               6.099153e+01
## First_Flr_SF              9.477449e-01
## Second_Flr_SF             .
## Total_Bsmt_SF             3.627699e+01
## Low_Qual_Fin_SF          -3.523480e+01
## Wood_Deck_SF              1.000632e+01
## Open_Porch_SF             1.203918e+01
## Bsmt_Unf_SF              -2.059528e+01
## Mas_Vnr_Area              1.297320e+01
## Garage_Cars               3.491107e+03
## Garage_Area               9.740129e+00
## Year_Built                3.150805e+02
## TotRms_AbvGrd            -2.518326e+03
## Full_Bath                -1.415647e+03
## Overall_QualAverage      -4.006722e+03
## Overall_QualBelow_Average -1.084480e+04
## Overall_QualExcellent     8.719850e+04
## Overall_QualFair         -8.763236e+03
## Overall_QualGood          1.111389e+04
## Overall_QualVery_Excellent 1.559964e+05
## Overall_QualVery_Good     3.730815e+04
## Kitchen_QualFair         -1.420320e+04
## Kitchen_QualGood         -7.639239e+03
## Kitchen_QualTypical      -1.644274e+04
## Fireplaces                8.190689e+03
## Fireplace_QuFair         -3.809974e+03
## Fireplace_QuGood          2.196176e+03
## Fireplace_QuNo_Fireplace  .
## Fireplace_QuPoor         -1.484163e+03
## Fireplace_QuTypical      -4.125304e+03
## Exter_QualFair           -1.695505e+04
## Exter_QualGood            .
## Exter_QualTypical        -4.790664e+03
## Lot_Frontage              8.663344e+01
## Lot_Area                  5.915806e-01
## Longitude                -2.246220e+04
## Latitude                  3.767830e+04
## Misc_Val                  3.093854e-01
## Year_Sold                -1.654627e+02
```

Using the 1SE rule, the optimal lambda is **403.4288**. During the variable selection process, variables Second_Flr_SF, Fireplace_QuNo_Fireplace, and Exter_QualGood are removed from the model. When the 1SE rule is applied, there are **36 predictors** included in the model, which is 1 fewer than the original lasso model.

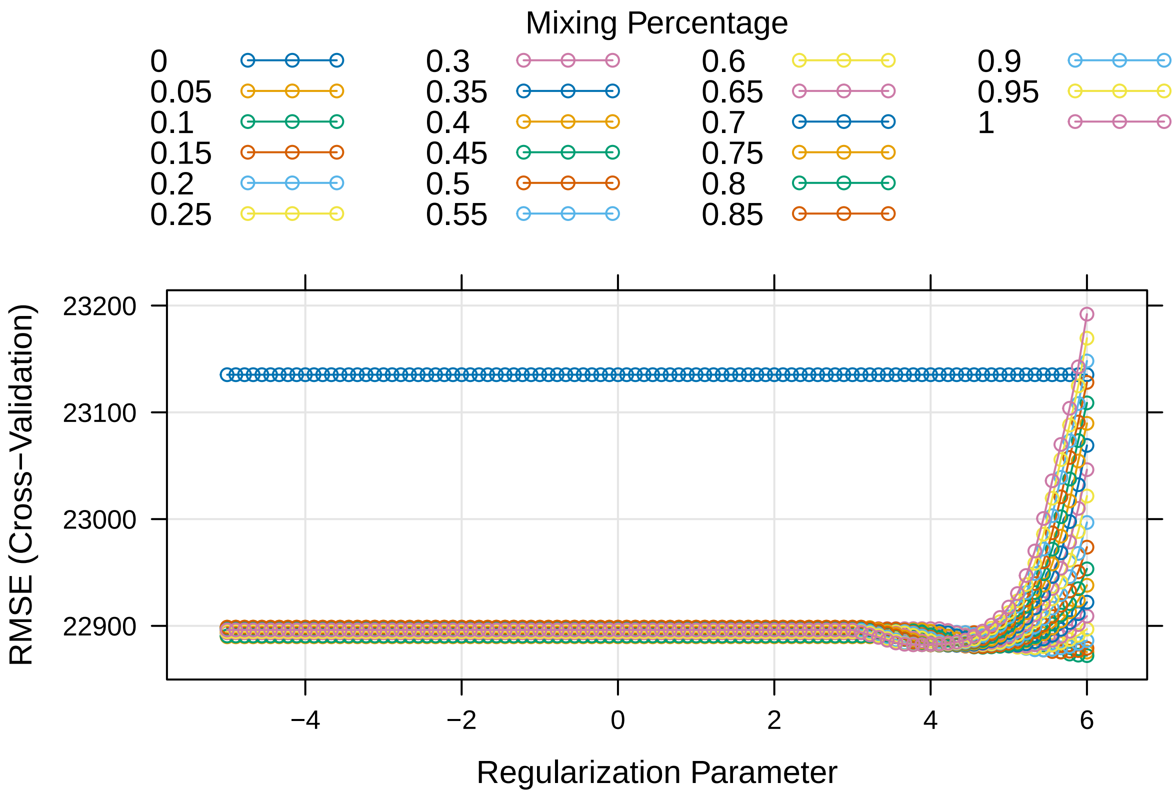
Question (b): Elastic Net

To fit the elastic net model, I began with a wide lambda range.

```r
# Set seed to ensure reproducibility

set.seed(16)

# Fit elastic net model
# Tuning the different lambda ranges

enet.fit <- train(Sale_Price ~ .,
                  data = training_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(6, -5, length = 100))),
                  trControl = ctrl1)

# Results

print(enet.fit$bestTune)
```

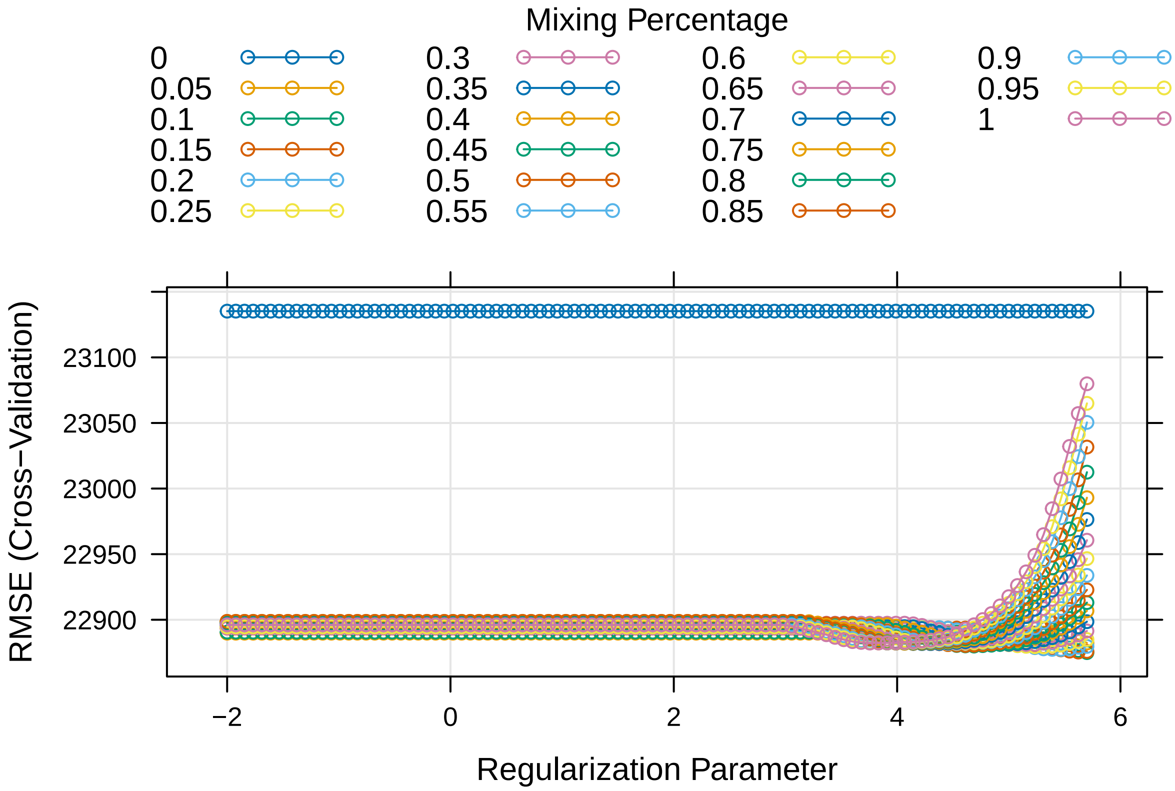
```
##     alpha   lambda
## 300   0.1 403.4288
```

```r
# Cross validation plot

plot(enet.fit, xTrans = log)
```



After reviewing the cross-validation plot, I refined the lambda range.

```r
# Set seed to ensure reproducibility

set.seed(16)

# Adjusting

enet.fit <- train(Sale_Price ~ .,
                  data = training_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(5.7, -2, length = 100))),
                  trControl = ctrl1)

# Cross validation plot

plot(enet.fit, xTrans = log)
```




```r
# Optimal lambda

print(enet.fit$bestTune)
```

```
##     alpha   lambda
## 300   0.1 298.8674
```

The cross validation plot shows the RMSE values were fairly stable at lower regularisation values, but increasing steeply when log(lambda)   6. Therefore, the selected tuning parameters are **alpha = 0.1** and **lambda = 298.8674**.

```r
# Set seed to ensure reproducibility

set.seed(16)

# Predictions using testing dataset

enet.pred <- predict(enet.fit, newdata = testing_data)

# Test error

test_mse <- mean((enet.pred - testing_data$Sale_Price)^2)

# Results

print(test_mse)
```

```
## [1] 440832286
```

From this, the test error of the model is **440832286**.