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)
library(pls)
library(knitr)
```

Question (a): Lasso Model

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

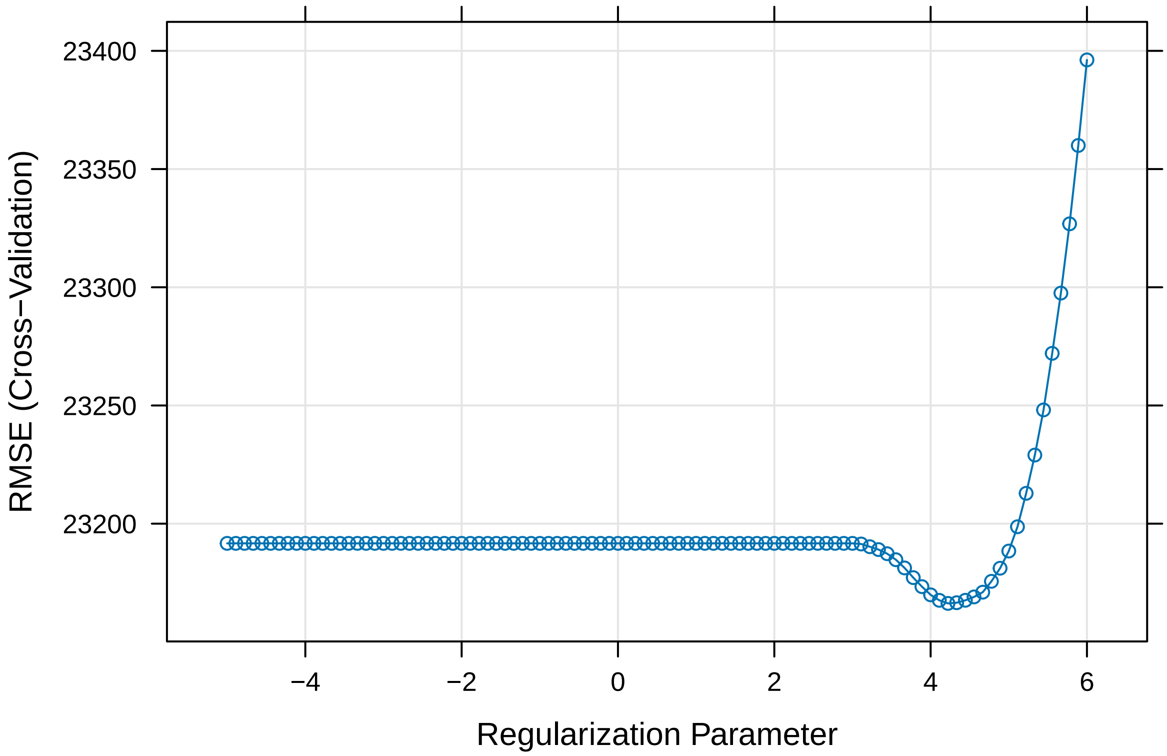
Next, we initialise 10-fold cross-validation to partition the training data into 10 equal subsets. This allows training the model on 9 folds while validating on the final fold. This ensures we evaluate the performance of the model, while avoiding overfitting.

```
# Load training and testing data

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

```r
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).

```r
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.

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

# Find optimal tuning parameter

lasso.fit$bestTune
```

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

```
# 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)

lasso_rmse <- sqrt(mean((lasso_preds - testing_data$Sale_Price)^2))

print(lasso_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
```

```r
# Lasso 1SE RMSE

lasso_1SE_preds <- predict(lasso_1se_fit, newdata = testing_data)

lasso_1SE_rmse <- sqrt(mean((lasso_1SE_preds - testing_data$Sale_Price)^2))

print(lasso_1SE_rmse)
```

```
## [1] 20511.62
```

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

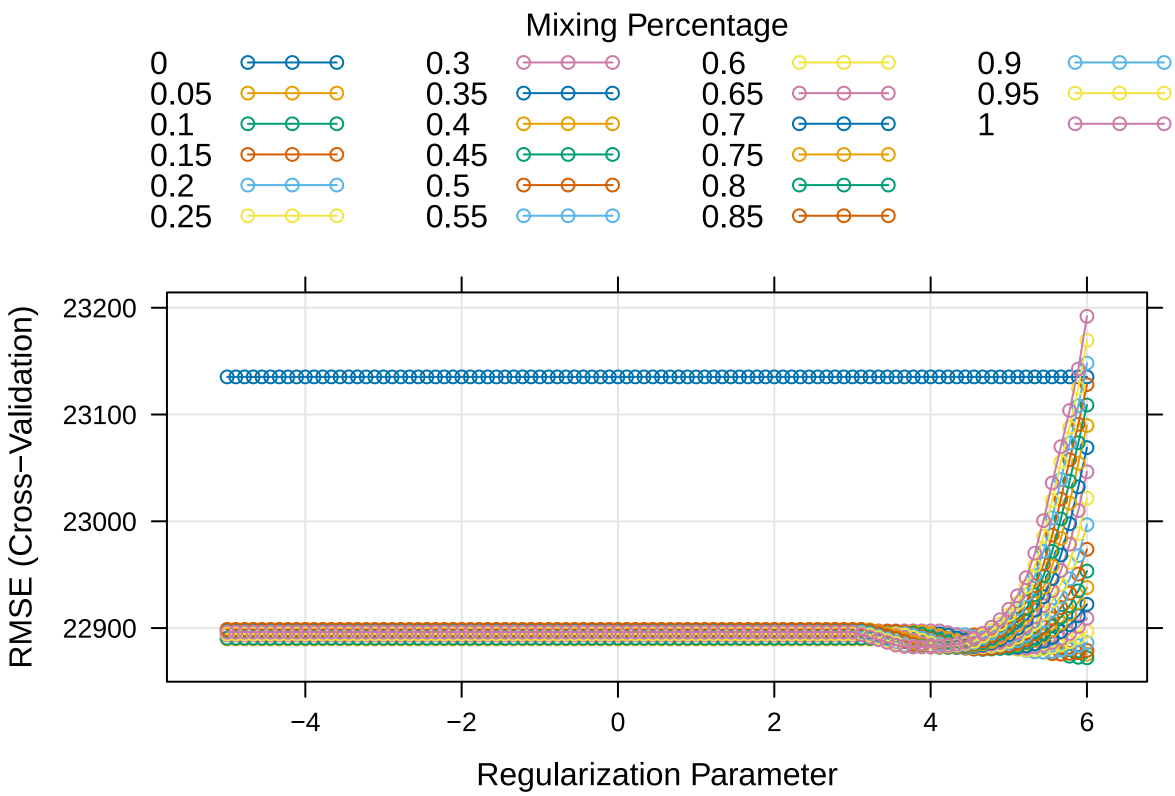
To fit the elastic net model, I began with a wide lambda range.

```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
```

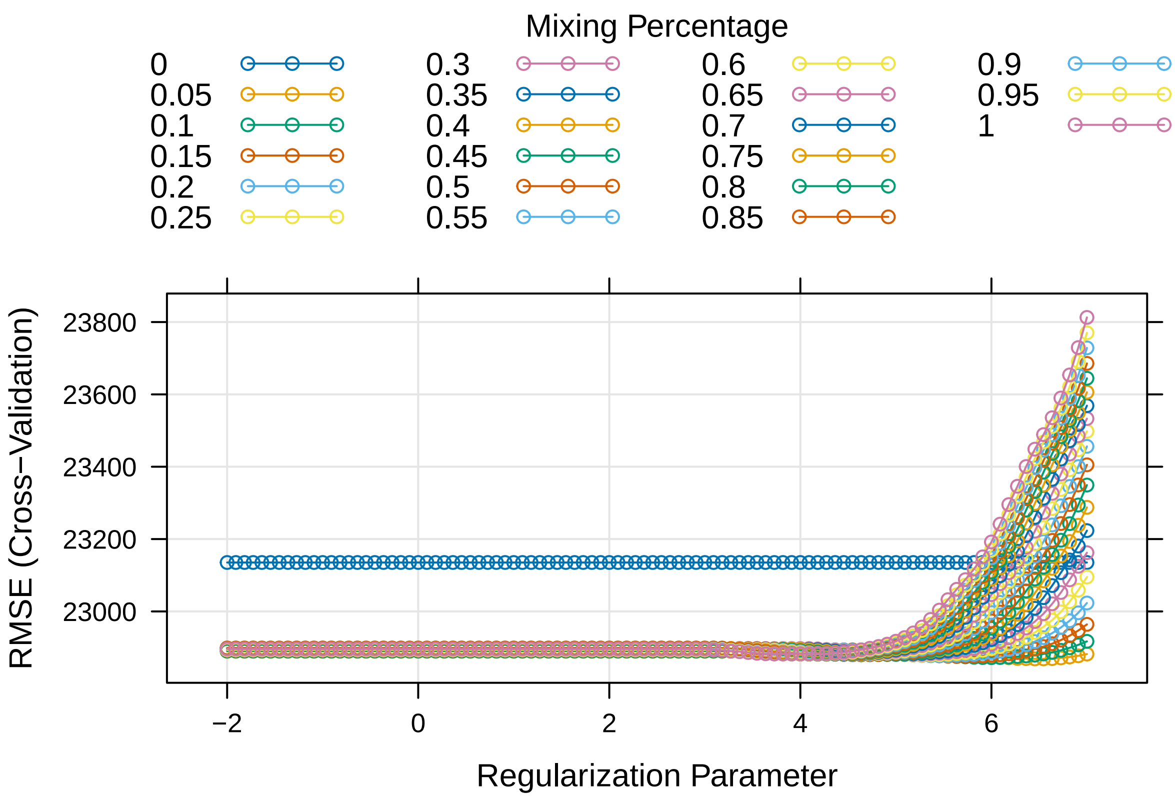
```
# Cross validation plot

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



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

```
# Set seed to ensure reproducibility

set.seed(16)

# Adjusting

enet.fit <- train(Sale_Price ~ .,
```

```r
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                          lambda = exp(seq(7, -2, length = 100))),
                   trControl = ctrl1)

# Cross validation plot

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




```r
# Optimal lambda

print(enet.fit$bestTune)
```

```
##     alpha   lambda
## 194  0.05 635.5848
```

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 = 0.05** and **lambda = 635.5848**.

```r
# Set seed to ensure reproducibility

set.seed(16)
```

```r
# Predictions using testing dataset

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

# Test error

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

# Results

print(enet_test_mse)
```

```
## [1] 438041526
```

From this, the test error of the model is **438041526**.

```r
# Set seed to ensure reproducibility

set.seed(16)

# Applying 1SE rule to elastic net model

enet_1se_fit <- train(
  Sale_Price ~ .,
  data = training_data,
  method = "glmnet",
  tuneGrid = expand.grid(
    alpha = seq(0, 1, length = 21),
    lambda = exp(seq(7, -2, length = 100))
  ),
  trControl = ctrl_1se
)

enet_1se_fit$bestTune$lambda
```

```
## [1] 1096.633
```

```r
enet_1se_fit$bestTune$alpha
```

```
## [1] 0
```

Yes, it is possible to apply the 1SE rule to selecting tuning parameters for elastic net. The elastic net method includes penalties from both ridge regression and lasso (the mixing parameter alpha that determines the balance between ridge and lasso penalties, and the overall regularisation strength lambda). The 1SE rule is defined as the most regularised model such that error is within one standard error of the minimum.

Therefore, using the 1SE rule, it is possible to select the most regularised model (i.e., the largest lambda) for each alpha value that has error within one standard error of the minimum, then compare across different alpha values to give the effective regularisation via the ridge-type penalty and feature selection via the lasso penalty, as determined by cross-validation.

Based on our data, the 1SE rule model parameters are alpha = **0** and lambda = **1096.633**. Given that alpha = 0, this indicates ridge regression was the optimal model.

I proceeded to find the test error of the 1SE model.

```r
# Set seed to ensure reproducibility

set.seed(16)

# Predictions using testing dataset for 1SE model

enet_1se_pred <- predict(enet_1se_fit, newdata = testing_data)

# Test error

enet_1se_test_mse <- mean((enet_1se_pred - testing_data$Sale_Price)^2)

# Results

print(enet_1se_test_mse)
```

```
## [1] 426357707
```

The test error for the 1SE rule elastic net model is **426357707**.

Question (c): Partial least squares

I proceeded with fitting the partial least squares model using caret.

```r
# Set seed for reproducibility

set.seed(29)

# Fit pls model using caret

pls_fit <- train(Sale_Price ~ .,
                 data = training_data,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:19),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))

# Determine the optimal number of components

pls_ncomp <- pls_fit$bestTune$ncomp

print(pls_ncomp)
```

```
## [1] 8
```

```r
# Predict on test data

predy_caret <- predict(pls_fit, newdata = testing_data)
```

```r
# Find Test MSE

caret_test_mse <- mean((testing_data$Sale_Price - predy_caret)^2)

print(caret_test_mse)
```

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

Based on the computation above, the optimal number of components is **8**. The test error for this model is **440217938**.

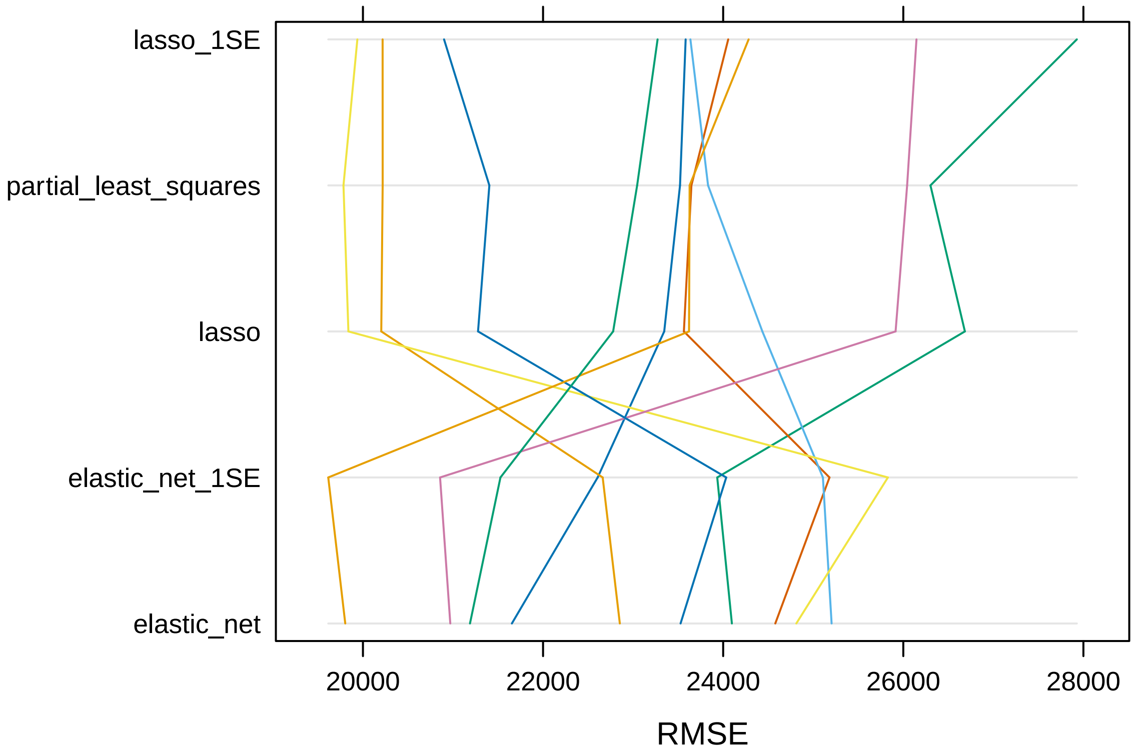
Question (d): Choose the best model for predicting the response and explain your choice.

```r
# Comparison of models using Training Data

set.seed(29)

lm.fit <- train(Sale_Price ~ .,
                data = training_data,
                method = "lm",
                trControl = ctrl1)

resamp <- resamples(list(lasso = lasso.fit,
                         lasso_1SE = lasso_1se_fit,
                         elastic_net = enet.fit,
                         elastic_net_1SE = enet_1se_fit,
                         partial_least_squares = pls_fit))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, lasso_1SE, elastic_net, elastic_net_1SE, partial_least_squares
## Number of resamples: 10
##
## MAE
##                           Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## lasso                 15146.33 15940.00 16968.25 16727.34 17496.76 18105.84
## lasso_1SE             14995.07 15777.68 16837.41 16679.07 17703.57 18362.96
## elastic_net           15273.74 15602.87 16632.95 16595.16 16749.81 19081.91
## elastic_net_1SE       15089.45 15923.82 16562.85 16567.48 16688.19 18868.62
## partial_least_squares 15130.60 15997.29 17030.97 16696.61 17539.64 17637.70
##                       NA's
## lasso                    0
## lasso_1SE                0
## elastic_net              0
## elastic_net_1SE          0
```

```
## partial_least_squares    0
##
## RMSE
##                           Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## lasso                 19838.57 21653.16 23454.69 23166.28 24231.76 26684.10
## lasso_1SE             19938.32 21493.52 23609.78 23396.17 24226.15 27927.38
## elastic_net           19803.71 21304.90 23189.65 22868.74 24458.29 25204.22
## elastic_net_1SE       19615.04 21796.89 23298.12 23135.25 24839.06 25829.27
## partial_least_squares 19783.77 21813.99 23574.14 23142.37 23786.08 26301.62
##                       NA's
## lasso                    0
## lasso_1SE                0
## elastic_net              0
## elastic_net_1SE          0
## partial_least_squares    0
##
## Rsquared
##                            Min.   1st Qu.    Median      Mean   3rd Qu.
## lasso                 0.8483459 0.8994293 0.9064960 0.9013017 0.9122969
## lasso_1SE             0.8436723 0.8951481 0.9005136 0.8988820 0.9133469
## elastic_net           0.8729801 0.8978269 0.9062437 0.9058184 0.9147487
## elastic_net_1SE       0.8718892 0.8946599 0.9053510 0.9040963 0.9131652
## partial_least_squares 0.8463744 0.8985566 0.9059684 0.9011446 0.9114279
##                            Max. NA's
## lasso                 0.9186015    0
## lasso_1SE             0.9191941    0
## elastic_net           0.9303233    0
## elastic_net_1SE       0.9288921    0
## partial_least_squares 0.9185280    0
```

```r
# Plot RMSE

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



Based on this plot and resampling summary, **elastic net** appears to have the lowest training RMSE range, with a mean RMSE of 22868.74.

```
# Comparison table of models
# Code from: https://bookdown.org/yihui/rmarkdown-cookbook/kable.html

set.seed(29)

# Convert MSE to RMSE for comparison
enet_test_rmse <- sqrt(enet_test_mse)
pls_test_rmse <- sqrt(caret_test_mse)
enet_1se_test_rmse <- sqrt(enet_1se_test_mse)


comparison_table <- tibble(
  Model = c("Lasso", "Lasso 1SE", "Elastic Net", "Elastic Net 1SE", "Partial Least Square Regression"),
  Test_RMSE = c(lasso_rmse, lasso_1SE_rmse, enet_test_rmse, enet_1se_test_rmse, pls_test_rmse)
)

# Using kable to present table

knitr::kable(comparison_table)
```

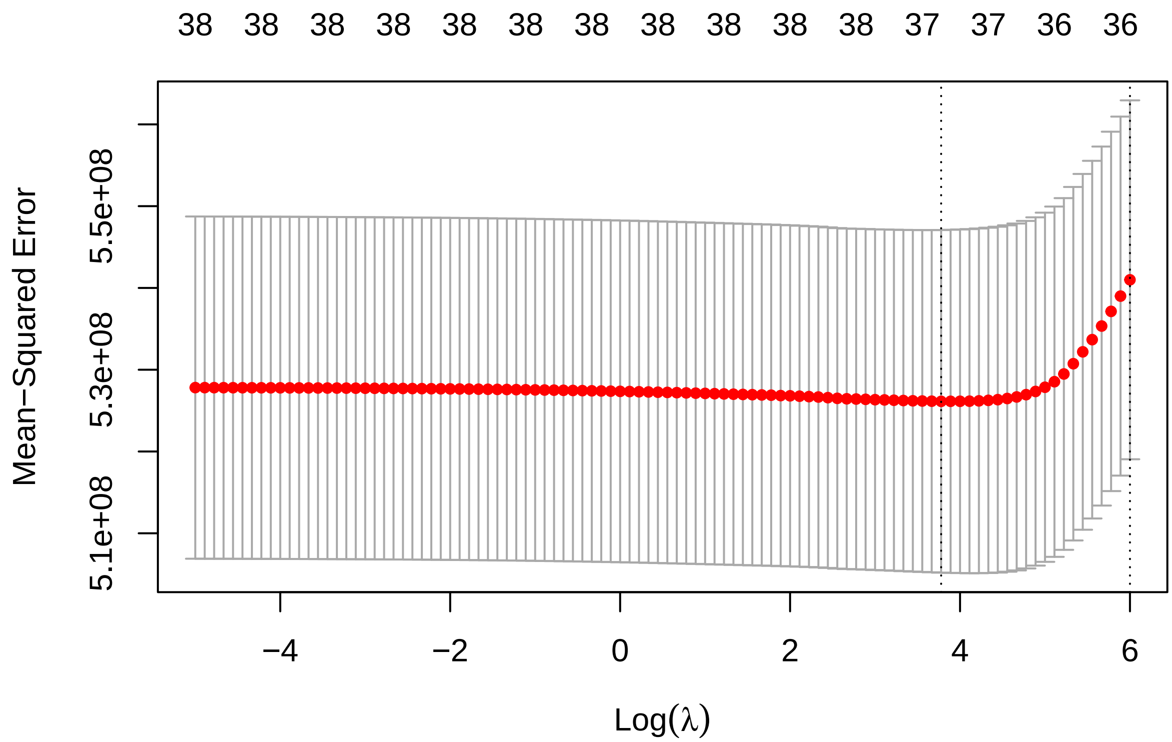
Model	Test_RMSE
Lasso	20969.20
Lasso 1SE	20511.62
Elastic Net	20929.44
Elastic Net 1SE	20648.43
Partial Least Square Regression	20981.37

Based on this, the lowest testing RMSE is from the Lasso 1SE model (RMSE = 20511.62).

Since model selection should be based on training RMSE (to avoid bias from the test set), Elastic Net is the best model for prediction in this case, even though the lasso 1SE model has a lower testing RMSE.

Question (e): Retrain model using glmnet

```
# Set seed for reproducibility

set.seed(29)

# Matrix of training data predictors for glmnet

x.train <- model.matrix(Sale_Price ~ ., training_data)[,-1]

y.train <- training_data$Sale_Price

# Matrix of predictors for test data

x.test <- model.matrix(Sale_Price ~ ., testing_data)[,-1]


# Fit lasso

lasso_glmnet <- glmnet(x.train, y.train,
                       alpha = 1,
                       lambda = exp(seq(6, -5, length = 100)))

# Next, cross-validation for optimal lambda

cv.lasso <- cv.glmnet(x.train, y.train,
                      alpha = 1,
                      lambda = exp(seq(6, -5, length = 100)))

plot(cv.lasso)
```




```
# Best lambda that minimises RMSE

print(cv.lasso$lambda.min)
```

```
## [1] 43.71878
```

```
# Coefficients at optimal lambda

predict(cv.lasso, s = "lambda.min", type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                 lambda.min
## (Intercept)                  -4.895849e+06
## Gr_Liv_Area                   6.563920e+01
## First_Flr_SF                  7.866606e-01
## Second_Flr_SF                 .
## Total_Bsmt_SF                 3.534723e+01
## Low_Qual_Fin_SF              -4.125144e+01
## Wood_Deck_SF                  1.175170e+01
## Open_Porch_SF                 1.563393e+01
## Bsmt_Unf_SF                  -2.089478e+01
## Mas_Vnr_Area                  1.075014e+01
## Garage_Cars                   4.109831e+03
## Garage_Area                   8.092846e+00
## Year_Built                    3.235015e+02
```

```
## TotRms_AbvGrd               -3.679742e+03
## Full_Bath                   -3.994482e+03
## Overall_QualAverage         -4.909818e+03
## Overall_QualBelow_Average  -1.257695e+04
## Overall_QualExcellent       7.464381e+04
## Overall_QualFair            -1.091948e+04
## Overall_QualGood             1.216881e+04
## Overall_QualVery_Excellent  1.341508e+05
## Overall_QualVery_Good       3.792195e+04
## Kitchen_QualFair            -2.550193e+04
## Kitchen_QualGood            -1.779096e+04
## Kitchen_QualTypical         -2.585294e+04
## Fireplaces                   1.082636e+04
## Fireplace_QuFair            -7.714837e+03
## Fireplace_QuGood             .
## Fireplace_QuNo_Fireplace     1.883694e+03
## Fireplace_QuPoor            -5.696096e+03
## Fireplace_QuTypical         -7.002341e+03
## Exter_QualFair              -3.478720e+04
## Exter_QualGood              -1.640951e+04
## Exter_QualTypical           -2.086944e+04
## Lot_Frontage                 1.005252e+02
## Lot_Area                     6.046880e-01
## Longitude                   -3.364932e+04
## Latitude                     5.645067e+04
## Misc_Val                     8.584413e-01
## Year_Sold                   -5.875701e+02
```

The final model includes 37 predictors, which is the same as the number of predictors identified in (a) using lasso (caret method).

Compared to the caret method implemented in Question (a), the tuning parameter lambda is notably different (glmnet = 43.71878, and caret = 68.18484). Both methods do use 10-fold cross validation to find the optimal lambda.

However, in glmnet, the built in cross validation function cv.glmnet() performs 10-fold cross validation once, computing RMSE for each lambda based on the 10 validation sets (Hastie et al., 2024). Whereas the caret package's trainControl(method = "cv", number = 10) also applies 10-fold cross validation, but averages the RMSE obtained across multiple resampling instances (Kuhn, 2020).

References

Kuhn, M. (2020). Caret package documentation.

Hastie, T., Tibshirani, R., & Friedman, J. (2024). An Introduction to glmnet.