HW 05

TUAN BUI

10/3/2021

```
housing_data <- read.csv('~/OneDrive - Stony Brook University/SBU/MAT +
AMS/Fall 2021/AMS 380/hw/05/Ames_Housing_Data.csv', header = T)

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse
1.3.1 ──

## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.4     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   2.0.1     ✓ forcats 0.5.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(glmnet)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-2
```

Question 1

```
colSums(is.na(housing_data))

##           Id      LotArea  OverallQual  OverallCond    YearBuilt
YearRemodAdd
##            0            0            0            0            0
0
##   CentralAir    X1stFlrSF    X2ndFlrSF    GrLivArea     FullBath
HalfBath
##            0            0            0            0            0
0
## BedroomAbvGr KitchenAbvGr TotRmsAbvGrd   Fireplaces   GarageCars
GarageArea
##            0            0            0            0            0
0
##       YrSold    SalePrice
##            0            0

## There is no missing values observation in this data
```

Question 2

```
# Dividing the data into 75% training and 25% testing
set.seed(123)
housing.samples <- housing_data$SalePrice %>%
  createDataPartition(p = 0.75, list = FALSE)
train.data  <- housing_data[housing.samples, ]
test.data <- housing_data[-housing.samples, ]
```

Question 3

```
# Predictor variables
x <- model.matrix(SalePrice~., train.data)[,-1]
# Outcome variable
y <- train.data$SalePrice
```

3.a.

```
# Make predictions on the test data
cv_3 <- cv.glmnet(x, y, alpha = 0)
# Display the best lambda value
cv_3$lambda.min

## [1] 6452.856

## The best lambda value is 6452.856
```

3.b.

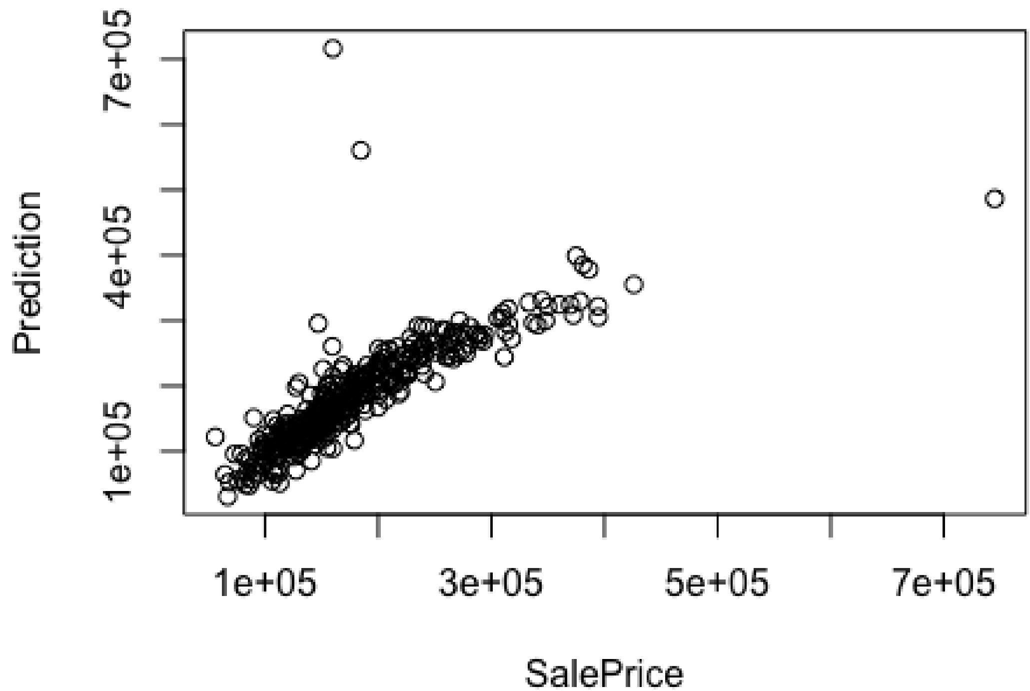
```r
model_3 <- glmnet(x, y, alpha = 0, lambda = cv_3$lambda.min)
# Display the coefficients of the fitted model
coef(model_3)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  -5.868377e+04
## Id           -2.400086e+00
## LotArea       6.829978e-01
## OverallQual   1.552083e+04
## OverallCond   4.734443e+03
## YearBuilt     4.367701e+02
## YearRemodAdd  2.036837e+02
## CentralAirY  -9.163648e+03
## X1stFlrSF     5.141967e+01
## X2ndFlrSF     1.721007e+01
## GrLivArea     3.865641e+01
## FullBath     -2.576679e+03
## HalfBath      3.410324e+02
## BedroomAbvGr -1.072097e+04
## KitchenAbvGr -3.025974e+04
## TotRmsAbvGrd  4.279323e+03
## Fireplaces    5.279621e+03
## GarageCars    2.424886e+03
## GarageArea    3.650612e+01
## YrSold       -6.243367e+02
```

3.c.

```r
# Make predictions on the test data
x.test <- model.matrix(SalePrice ~., test.data)[,-1]
predictions_3 <- model_3 %>% predict(x.test) %>% as.vector()

# Plot predictions
plot(test.data$SalePrice ,predictions_3, xlab = "SalePrice", ylab =
"Prediction")
```




```
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions_3, test.data$SalePrice),
  Rsquare = R2(predictions_3, test.data$SalePrice)
)

##       RMSE   Rsquare
## 1 46713.63 0.6691365
```

Question 4

4.a.

```
# Make predictions on the test data
cv_4 <- cv.glmnet(x, y, alpha = 1)
# Display the best lambda value
cv_4$lambda.min

## [1] 352.4729

## The best lambda value is 352.4729
```

4.b.

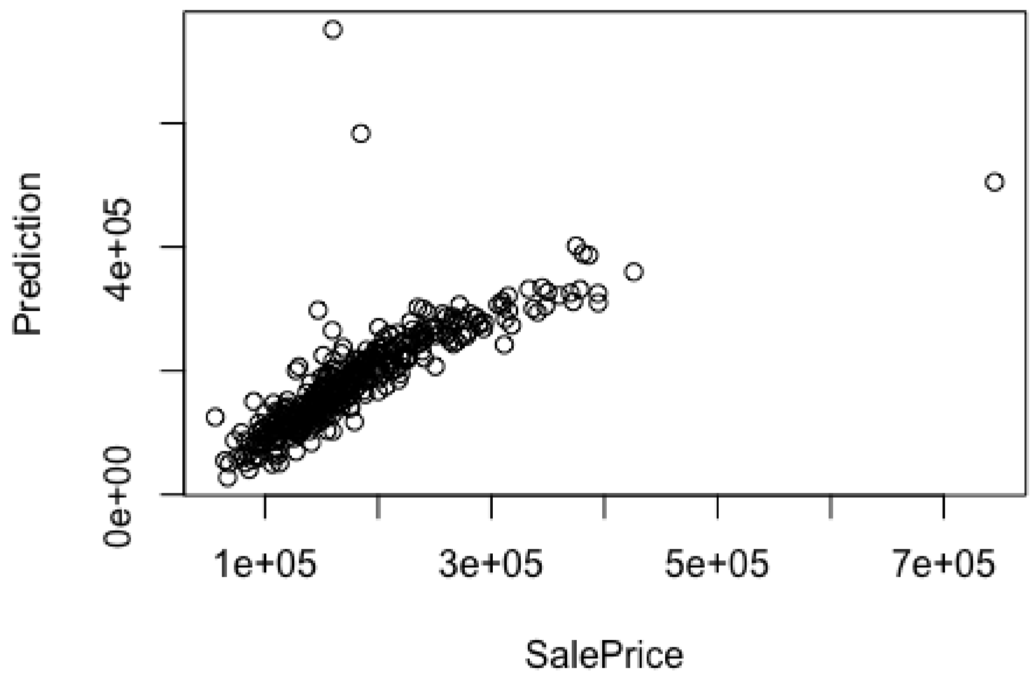
```
model_4 <- glmnet(x, y, alpha = 1, lambda = cv_4$lambda.min)
# Display the coefficients of the fitted model
coef(model_4)

## 20 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)  -6.160530e+05
## Id           -1.359380e+00
## LotArea       6.885348e-01
## OverallQual   1.673469e+04
## OverallCond   5.628004e+03
## YearBuilt     5.183887e+02
## YearRemodAdd  1.105492e+02
## CentralAirY  -1.011034e+04
## X1stFlrSF     3.613856e+01
## X2ndFlrSF     .
## GrLivArea     6.455739e+01
## FullBath     -4.871532e+03
## HalfBath     -2.083079e+02
## BedroomAbvGr -1.188420e+04
## KitchenAbvGr -2.869665e+04
## TotRmsAbvGrd  3.101192e+03
## Fireplaces    3.087910e+03
## GarageCars    .
## GarageArea    3.684554e+01
## YrSold       -3.393338e+02
```

4.c.

```
# Make predictions on the test data
predictions_4 <- model_4 %>% predict(x.test) %>% as.vector()

# Plot predictions
plot(test.data$SalePrice ,predictions_4, xlab = "SalePrice", ylab =
"Prediction")
```




```
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions_4, test.data$SalePrice),
  Rsquare = R2(predictions_4, test.data$SalePrice)
)

##       RMSE   Rsquare
## 1 48460.31 0.6606179
```

Question 5

5.a.

```
model_5 <- train(
  SalePrice ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneLength = 10
)
# Best tuning parameter
model_5$bestTune

##   alpha   lambda
## 5   0.1 849.1095
```

5.b.

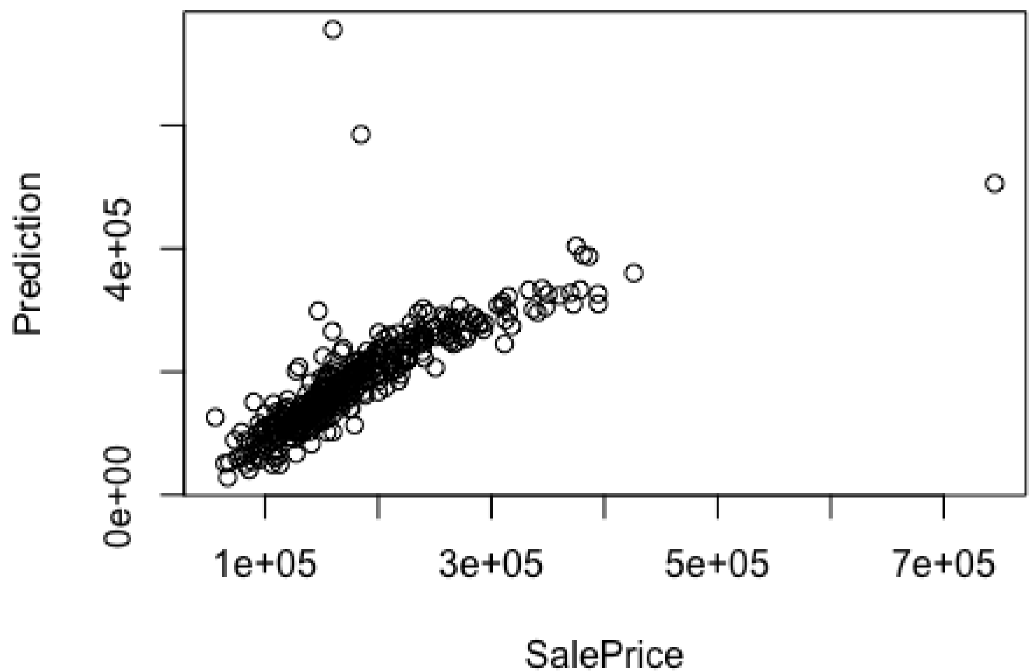
```
coef(model_5$finalModel, model_5$bestTune$lambda)

## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  -3.184591e+05
## Id           -2.065248e+00
## LotArea       6.999641e-01
## OverallQual   1.641517e+04
## OverallCond   5.931914e+03
## YearBuilt     5.464418e+02
## YearRemodAdd  1.242764e+02
## CentralAirY  -1.246334e+04
## X1stFlrSF     5.519033e+01
## X2ndFlrSF     2.097999e+01
## GrLivArea     4.537295e+01
## FullBath     -6.947849e+03
## HalfBath     -2.248932e+03
## BedroomAbvGr -1.256102e+04
## KitchenAbvGr -3.025522e+04
## TotRmsAbvGrd  3.844605e+03
## Fireplaces    3.509950e+03
## GarageCars    .
## GarageArea    3.757935e+01
## YrSold       -5.263536e+02
```

5.c.

```
predictions_5 <- model_5 %>% predict(test.data)

#Plot
plot(test.data$SalePrice ,predictions_5, xlab = "SalePrice", ylab =
"Prediction")
```




```
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions_5, test.data$SalePrice),
  Rsquare = R2(predictions_5, test.data$SalePrice)
)

##        RMSE   Rsquare
## 1 48709.54 0.6592282
```

5.d.

```
lambda_1 <- 10^seq(-3, 3, length = 100)

# Ridge model
ridge <- train(
  SalePrice ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 0, lambda = lambda_1)
  )

# Lasso model
lasso <- train(
  SalePrice ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
```

```r
  tuneGrid = expand.grid(alpha = 1, lambda = lambda_1)
  )

# Elastic model
elastic <- train(
  SalePrice ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneLength = 10
  )

# Compare
models <- list(ridge = ridge, lasso = lasso, elastic = elastic)
resamples(models) %>% summary(metric = "RMSE")

##
## Call:
## summary.resamples(object = ., metric = "RMSE")
##
## Models: ridge, lasso, elastic
## Number of resamples: 10
##
## RMSE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## ridge   25407.72 28499.66 31543.68 33244.48 36674.04 45814.19    0
## lasso   27048.45 30155.21 30940.18 33129.10 34034.23 43217.22    0
## elastic 27627.21 31063.40 32297.70 33287.69 35716.51 40038.10    0

## It can be seen that the Lasso net model has the lowest median RMSE. Hence,
Lasso model is the best for the Ames Housing data.
```