CAT 2

79546 - Stephen K. Ng'etich

Table of Contents

Pre-requisite

load package

```
# Clear variables
rm(list=ls())

library(readxl)

## Warning: package 'readxl' was built under R version 4.1.3

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.1.3

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.8
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Warning: package 'dplyr' was built under R version 4.1.3

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

library(caret)

## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: lattice

##
## Attaching package: 'caret'

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

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.1.3

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

## Loaded glmnet 4.1-3

## set the seed to make your partition reproducible
set.seed(123)
```

Load dataset

```
# Load Dataset
dataset <- read_excel("dataset/TestData.xlsx")
```

Question

(a) Split the data set into 75% training set and 25% test set.

```
## 75% of the sample size
sample_size <- floor(0.75 * nrow(dataset))


train_ind <- sample(seq_len(nrow(dataset)), size = sample_size)
```

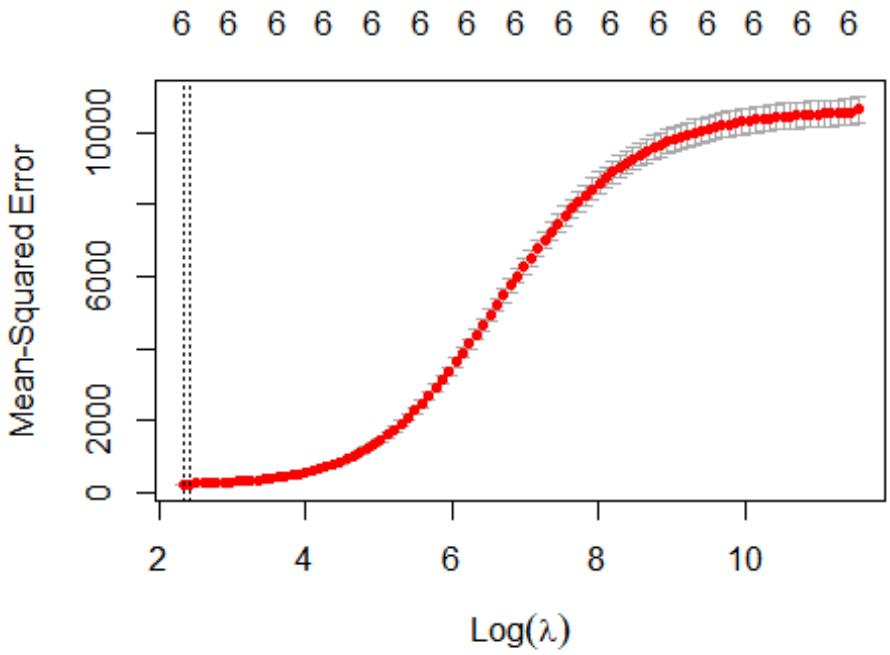
```
train <- dataset[train_ind, ]
test <- dataset[-train_ind, ]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
lm_model = lm(Response ~ . , data=train)
#summary(lm_model)

predictions = predict.lm(lm_model,newdata = test)

#Model performance metrics
ml_performance.lse=data.frame(
            MODEL = "Least Squares",
            s1 = R2(predictions, test$Response),
            RMSE = RMSE(predictions, test$Response),
            MAE = MAE(predictions, test$Response))

ml_performance.lse

##           MODEL        s1     RMSE      MAE
## 1 Least Squares 0.9908415 9.955736 8.279993
```

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
train.matrix = model.matrix(Response~., data = train)
test.matrix = model.matrix(Response~., data = test)

#Choose lambda using cross-validation
crossvalidation = cv.glmnet(train.matrix,train$Response,alpha=0)
plot(crossvalidation)
```

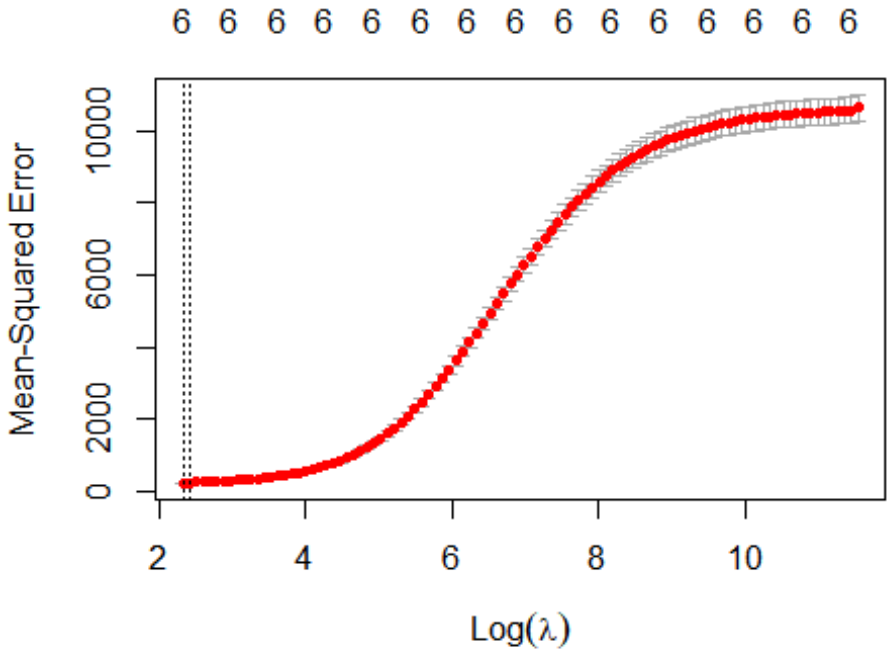



```
bestlamda = crossvalidation$lambda.min
bestlamda

## [1] 10.24674

#Fit a ridge regression
ridge_model = glmnet(train.matrix,train$Response,alpha = 0)
#Make predictions
ridge_predictions = predict(ridge_model,s=bestlamda,newx = test.matrix)
#Calculate test error

#Model performance metrics
ml_performance.ridge = data.frame(
                        MODEL = "Ridge regression",
            R2 = R2(ridge_predictions, test$Response),
            RMSE = RMSE(ridge_predictions, test$Response),
            MAE = MAE(ridge_predictions, test$Response))
ml_performance.ridge

##              MODEL        s1     RMSE      MAE
## 1 Ridge regression 0.9809305 14.99081 11.34165
```

##(d) Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
train.matrix = model.matrix(Response~., data = train)
test.matrix = model.matrix(Response~., data = test)
```

```r
#Choose lambda using cross-validation
crossvalidation1 = cv.glmnet(train.matrix,train$Response,alpha=1)
plot(crossvalidation)
```




```r
bestlamda1 = crossvalidation1$lambda.min
bestlamda1
```

```
## [1] 0.5597055
```

```r
#Fit lasso model
# Note alpha=1 for lasso only and can blend with ridge penalty down to
lasso_model = glmnet(train.matrix,train$Response,alpha=1)
#Make predictions
lasso_predictions = predict(lasso_model,s=bestlamda1,newx=test.matrix)
#Model performance metrics
ml_performance.lasso = data.frame(
  MODEL = "Lasso regression",
  R2 = R2(lasso_predictions, test$Response),
            RMSE = RMSE(lasso_predictions, test$Response),
            MAE = MAE(lasso_predictions, test$Response))
ml_performance.lasso
```

```
##              MODEL        s1     RMSE      MAE
## 1 Lasso regression 0.9901851 10.33082 8.359939
```

(e) Comment on the results obtained. How accurately can we predict the response variable? Is there much difference among the test errors resulting from these three approaches? Present and discuss results for the approaches

The following table represents model performance

```
t = rbind(ml_performance.lse,ml_performance.lasso,ml_performance.ridge)
knitr::kable(t,"simple")
```

MODEL	s1	RMSE	MAE
Least Squares	0.9908415	9.955736	8.279993
Lasso regression	0.9901851	10.330821	8.359939
Ridge regression	0.9809305	14.990815	11.341649
The best model in p	redicting th	e responce v	ariable is `Least Squares` since it has the least Root Mean Square Error while Ridge regression has the worst model perfomance.

`Ridge Regression` has the highest margin in test errors compare to the other 2 regression models