ams580\_midterm1

Mustafa Yigit Isik

2023-03-29

#PART 1 - Question 1  
#Import Libraries  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(neuralnet)  
library(keras)  
library(tidyverse)

## ── Attaching packages  
## ───────────────────────────────────────  
## tidyverse 1.3.2 ──

## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0   
## ✔ readr 2.1.4 ✔ forcats 1.0.0   
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::compute() masks neuralnet::compute()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::lift() masks caret::lift()

library(dplyr)  
library(cloudml)

## Loading required package: tfruns

library(randomForest)

## randomForest 4.7-1.1  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine  
##   
## The following object is masked from 'package:ggplot2':  
##   
## margin

library(rpart)  
library(rattle)

## Loading required package: bitops  
## Rattle: A free graphical interface for data science with R.  
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.  
##   
## Attaching package: 'rattle'  
##   
## The following object is masked from 'package:randomForest':  
##   
## importance

library(caTools)  
  
#Read data  
data <- read.csv('/Users/mustafayigitisik/Desktop/stuff/semesters/spring 2023/ams 580/midterm/Puzzle.csv')  
#Categorize $15 column   
data$x15\_1 <- ifelse(data$x15 == 1, 1, 0)  
data$x15\_2 <- ifelse(data$x15 == 2, 1, 0)  
data <- subset(data, select = -c(x15))  
#Remove missing values  
data <- na.omit(data)  
cat("There are", nrow(data)," values left\n")

## There are 4601 values left

str(data)

## 'data.frame': 4601 obs. of 17 variables:  
## $ x1 : num 0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...  
## $ x2 : num 0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...  
## $ x3 : num 0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...  
## $ x4 : num 0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...  
## $ x5 : num 0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...  
## $ x6 : num 0 0.07 0.06 0 0 0 0 0 0 0 ...  
## $ x7 : num 0 0.43 0.06 0 0 0 0 0 0.15 0 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0.06 0 0 0 0 0 0 0 ...  
## $ x10 : num 0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...  
## $ x11 : num 0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...  
## $ x12 : num 3.76 5.11 9.82 3.54 3.54 ...  
## $ x13 : num 0 0.067 0.0508 0 0 ...  
## $ x14 : num 2.4 4.04 4.42 1.1 1.1 ...  
## $ y : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...

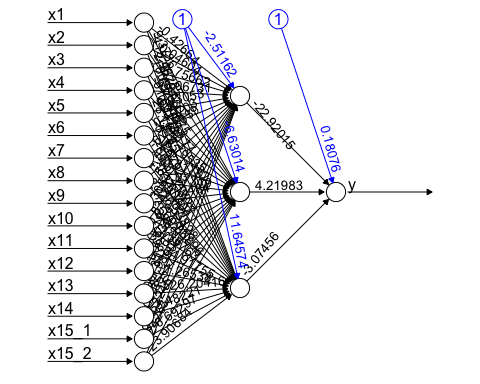
#Split data 80-20  
set.seed(123)  
training.samples <- data$y %>%   
 createDataPartition(p=.8, list = FALSE)  
train.data <- data[training.samples, ]  
test.data <- data[-training.samples, ]  
str(train.data) #3681 obs

## 'data.frame': 3681 obs. of 17 variables:  
## $ x1 : num 0 0.21 0.19 0.31 0.31 0 0.3 0.38 0.96 0.25 ...  
## $ x2 : num 0 0.07 0.12 0.63 0.63 0 0 0 0 0 ...  
## $ x3 : num 0 0.21 0.38 0.31 0.31 0.96 0.76 0 0.96 0.12 ...  
## $ x4 : num 0.64 0.79 0.45 0.31 0.31 1.28 0.92 0.64 0 0.12 ...  
## $ x5 : num 0.32 0.14 0.06 0.31 0.31 0.96 0 0 0 0 ...  
## $ x6 : num 0 0.07 0.06 0 0 0 0 0 0 0 ...  
## $ x7 : num 0 0.43 0.06 0 0 0 0.15 0 0 0 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0.06 0 0 0 0 0 0 0 ...  
## $ x10 : num 0.778 0.372 0.276 0.137 0.135 0.164 0.181 0.244 0.462 0.663 ...  
## $ x11 : num 0 0.18 0.184 0 0 0.054 0.203 0.081 0 0 ...  
## $ x12 : num 3.76 5.11 9.82 3.54 3.54 ...  
## $ x13 : num 0 0.067 0.0508 0 0 ...  
## $ x14 : num 2.4 4.04 4.42 1.1 1.1 ...  
## $ y : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...

str(test.data) #920 obs

## 'data.frame': 920 obs. of 17 variables:  
## $ x1 : num 0 0 0.42 0 0 1.66 0 0.48 0.06 0.06 ...  
## $ x2 : num 1.85 1.88 0 0 0 0 0 0 0.06 0.33 ...  
## $ x3 : num 0 0 0 0 0 0 0 0 0.06 0.39 ...  
## $ x4 : num 0 0 0 0 0.58 0 0 0.97 0.54 0.73 ...  
## $ x5 : num 0 0 1.27 2.94 1.16 0 3.33 0.48 0 0.26 ...  
## $ x6 : num 0 0 0 0 0 0 0 0.97 0.72 0 ...  
## $ x7 : num 0 0 0.42 0 0 3.33 0 0 0.06 0.26 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x10 : num 0 0 0.572 0.809 0.667 0.368 0.352 0.488 0.01 0.278 ...  
## $ x11 : num 0 0 0.063 0 0 0 0 0.244 0.167 0.23 ...  
## $ x12 : num 3 2.45 5.66 4.86 1.13 ...  
## $ x13 : num 0 0 0.036 0 0 ...  
## $ x14 : num 0 0 0 0 0.656 ...  
## $ y : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...

#PART 1 - Question 2  
#Set seed  
set.seed(123)  
#Build model  
model2 <- neuralnet(y~., data = train.data, hidden=3, err.fct = "sse",act.fct = "logistic", linear.output = F)  
#Plot perceptron  
plot(model2, rep = "best")



probabilities = predict(model2, test.data)  
pred2 = ifelse(probabilities > 0.5, 1, 0)  
#Confusion Matrix  
c2 = confusionMatrix(factor(pred2), factor(test.data$y), positive = "1")  
#Add predicted class label to testing data  
test.data.with\_pred2 <- cbind(test.data, pred2)  
str(test.data.with\_pred2)

## 'data.frame': 920 obs. of 18 variables:  
## $ x1 : num 0 0 0.42 0 0 1.66 0 0.48 0.06 0.06 ...  
## $ x2 : num 1.85 1.88 0 0 0 0 0 0 0.06 0.33 ...  
## $ x3 : num 0 0 0 0 0 0 0 0 0.06 0.39 ...  
## $ x4 : num 0 0 0 0 0.58 0 0 0.97 0.54 0.73 ...  
## $ x5 : num 0 0 1.27 2.94 1.16 0 3.33 0.48 0 0.26 ...  
## $ x6 : num 0 0 0 0 0 0 0 0.97 0.72 0 ...  
## $ x7 : num 0 0 0.42 0 0 3.33 0 0 0.06 0.26 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x10 : num 0 0 0.572 0.809 0.667 0.368 0.352 0.488 0.01 0.278 ...  
## $ x11 : num 0 0 0.063 0 0 0 0 0.244 0.167 0.23 ...  
## $ x12 : num 3 2.45 5.66 4.86 1.13 ...  
## $ x13 : num 0 0 0.036 0 0 ...  
## $ x14 : num 0 0 0 0 0.656 ...  
## $ y : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ pred2: num 0 0 1 1 0 1 1 1 1 1 ...

#PART 1 - Question 3  
#build the best random forest to predict the class label using the training data  
train.data$y = as.factor(train.data$y)  
test.data$y = as.factor(test.data$y)  
  
set.seed(123)  
model3 = train(  
 y ~., data = train.data, method = "rf",  
 trControl = trainControl("cv", number = 10),  
 importance = TRUE  
 )  
  
model3$bestTune

## mtry  
## 1 2

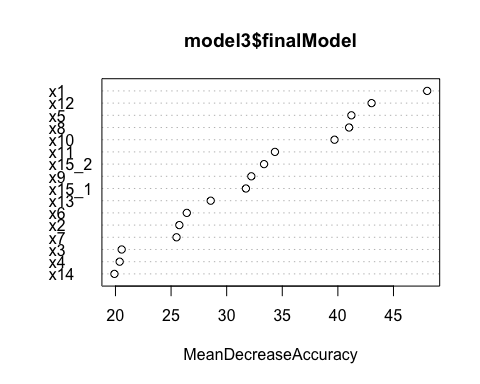
model3$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 5.98%  
## Confusion matrix:  
## 0 1 class.error  
## 0 2177 65 0.02899197  
## 1 155 1284 0.10771369

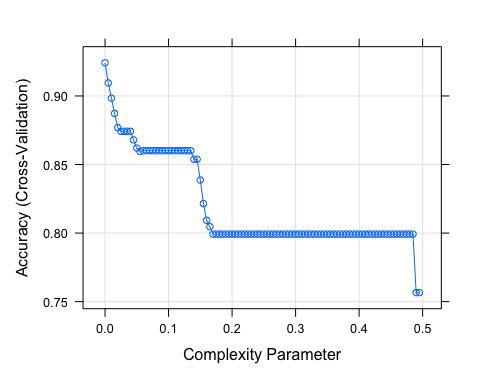
pred3 = model3 %>% predict(test.data)  
#Add predicted class label to testing data  
test.data.with\_pred3 <- cbind(test.data, pred3)  
str(test.data.with\_pred3)

## 'data.frame': 920 obs. of 18 variables:  
## $ x1 : num 0 0 0.42 0 0 1.66 0 0.48 0.06 0.06 ...  
## $ x2 : num 1.85 1.88 0 0 0 0 0 0 0.06 0.33 ...  
## $ x3 : num 0 0 0 0 0 0 0 0 0.06 0.39 ...  
## $ x4 : num 0 0 0 0 0.58 0 0 0.97 0.54 0.73 ...  
## $ x5 : num 0 0 1.27 2.94 1.16 0 3.33 0.48 0 0.26 ...  
## $ x6 : num 0 0 0 0 0 0 0 0.97 0.72 0 ...  
## $ x7 : num 0 0 0.42 0 0 3.33 0 0 0.06 0.26 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x10 : num 0 0 0.572 0.809 0.667 0.368 0.352 0.488 0.01 0.278 ...  
## $ x11 : num 0 0 0.063 0 0 0 0 0.244 0.167 0.23 ...  
## $ x12 : num 3 2.45 5.66 4.86 1.13 ...  
## $ x13 : num 0 0 0.036 0 0 ...  
## $ x14 : num 0 0 0 0 0.656 ...  
## $ y : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ pred3: Factor w/ 2 levels "0","1": 1 1 2 2 1 2 2 2 2 2 ...

#Confusion Matrix  
c3 = confusionMatrix(pred3, factor(test.data$y), positive = "1")  
#Plot the variables importance measures using MeanDecreaseAccuracy  
varImpPlot(model3$finalModel, type = 1)



#PART 1 - Question 4  
set.seed(123)  
  
model4 = train(  
 y ~., data = train.data, method = "rpart",  
 trControl = trainControl("cv", number = 10),  
 tuneLength = 100)  
  
plot(model4)

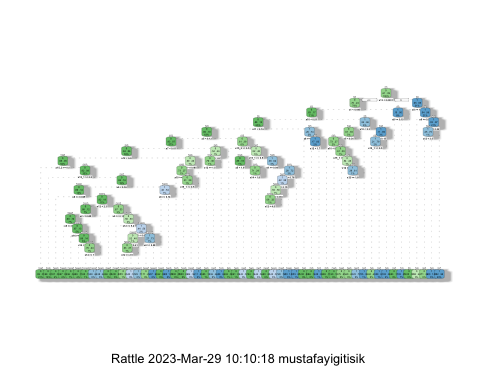


model4$bestTune

## cp  
## 1 0

fancyRpartPlot(model4$finalModel)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



pred4 = predict(model4, newdata = test.data)  
#Add predicted class label to testing data  
test.data.with\_pred4 <- cbind(test.data, pred4)  
str(test.data.with\_pred4)

## 'data.frame': 920 obs. of 18 variables:  
## $ x1 : num 0 0 0.42 0 0 1.66 0 0.48 0.06 0.06 ...  
## $ x2 : num 1.85 1.88 0 0 0 0 0 0 0.06 0.33 ...  
## $ x3 : num 0 0 0 0 0 0 0 0 0.06 0.39 ...  
## $ x4 : num 0 0 0 0 0.58 0 0 0.97 0.54 0.73 ...  
## $ x5 : num 0 0 1.27 2.94 1.16 0 3.33 0.48 0 0.26 ...  
## $ x6 : num 0 0 0 0 0 0 0 0.97 0.72 0 ...  
## $ x7 : num 0 0 0.42 0 0 3.33 0 0 0.06 0.26 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x10 : num 0 0 0.572 0.809 0.667 0.368 0.352 0.488 0.01 0.278 ...  
## $ x11 : num 0 0 0.063 0 0 0 0 0.244 0.167 0.23 ...  
## $ x12 : num 3 2.45 5.66 4.86 1.13 ...  
## $ x13 : num 0 0 0.036 0 0 ...  
## $ x14 : num 0 0 0 0 0.656 ...  
## $ y : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ pred4: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 2 2 2 ...

#Confusion Matrix  
c4 = confusionMatrix(pred4, factor(test.data$y), positive = "1")

#PART 1 - Question 5  
#Majority Voting  
# pred = cbind(pred2, pred3, pred4)  
# pred.m = apply(pred,1,function(x) names(which.max(table(x))))  
# pred.m = as.numeric(pred.m)  
# confusionMatrix(factor(pred.m), factor(test.data$y), positive = '1')

#PART 2 - Question 1  
#Read data  
data\_q2\_raw <- read.csv('/Users/mustafayigitisik/Desktop/stuff/semesters/spring 2023/ams 580/midterm/Maze.csv')  
#Convert $x16 to 0-1  
data\_q2\_raw$x16 = ifelse(data\_q2\_raw$x16 == 'y', 1, 0)  
#Take mean and sd for future reference  
mean = mean(data\_q2\_raw$y)  
sd = sd(data\_q2\_raw$y)  
#Normalize data  
data\_q2\_raw = data.frame(scale(data\_q2\_raw))  
#Take out empties  
data\_q2\_clean = na.omit(data\_q2\_raw)  
  
#Report how many observations with missing values  
cat('There are', nrow(data\_q2\_raw) - nrow(data\_q2\_clean), 'missing values.\n')

## There are 0 missing values.

#Split 80-20  
set.seed(123)  
training.samples\_q2 <- data\_q2\_clean$y %>%  
 createDataPartition(p = 0.8, list = FALSE)  
train.data\_q2 <- data[training.samples\_q2, ]  
test.data\_q2 <- data[-training.samples\_q2, ]  
str(train.data\_q2)

## 'data.frame': 1169 obs. of 17 variables:  
## $ x1 : num 0 0.21 0.31 0.31 0 0 0 0.3 0.38 0.25 ...  
## $ x2 : num 0 0.07 0.63 0.63 1.85 0 1.88 0 0 0 ...  
## $ x3 : num 0 0.21 0.31 0.31 0 0.96 0 0.76 0 0.12 ...  
## $ x4 : num 0.64 0.79 0.31 0.31 0 1.28 0 0.92 0.64 0.12 ...  
## $ x5 : num 0.32 0.14 0.31 0.31 0 0.96 0 0 0 0 ...  
## $ x6 : num 0 0.07 0 0 0 0 0 0 0 0 ...  
## $ x7 : num 0 0.43 0 0 0 0 0 0.15 0 0 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x10 : num 0.778 0.372 0.137 0.135 0 0.164 0 0.181 0.244 0.663 ...  
## $ x11 : num 0 0.18 0 0 0 0.054 0 0.203 0.081 0 ...  
## $ x12 : num 3.76 5.11 3.54 3.54 3 ...  
## $ x13 : num 0 0.067 0 0 0 ...  
## $ x14 : num 2.4 4.04 1.1 1.1 0 ...  
## $ y : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...

str(test.data\_q2)

## 'data.frame': 3432 obs. of 17 variables:  
## $ x1 : num 0.19 0.96 0 0.65 0 0 0 0 0 0 ...  
## $ x2 : num 0.12 0 0 0 0 0 0 0 0 0 ...  
## $ x3 : num 0.38 0.96 0 0.65 0 0 0 0 0 0 ...  
## $ x4 : num 0.45 0 0.69 0.65 0 0 0.97 1.26 2.53 0 ...  
## $ x5 : num 0.06 0 0.34 0.65 0 2.94 0.48 0 0 0.23 ...  
## $ x6 : num 0.06 0 0 1.3 0 0 0 0 0 0.23 ...  
## $ x7 : num 0.06 0 0 0.65 0 0 0 1.26 0 0.23 ...  
## $ x8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ x9 : num 0.06 0 0 0 0 0 0 0 0 0 ...  
## $ x10 : num 0.276 0.462 0.786 0.091 0 0.873 0.963 0.198 0 0.175 ...  
## $ x11 : num 0.184 0 0 0 0.302 0 0 0.596 0 0 ...  
## $ x12 : num 9.82 1.31 3.73 2.69 1.7 ...  
## $ x13 : num 0.0508 0 0 0 0 ...  
## $ x14 : num 4.42 0 2.57 1.75 0 ...  
## $ y : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_1: num 1 1 1 1 1 1 1 1 1 1 ...  
## $ x15\_2: num 0 0 0 0 0 0 0 0 0 0 ...

#PART 2 - Question 2  
library(tensorflow)

##   
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':  
##   
## train

train\_x = as.matrix(subset(train.data\_q2, select = -y))  
train\_y = as.matrix(subset(train.data\_q2, select = y))  
test\_x = as.matrix(subset(test.data\_q2, select = -y))  
test\_y = as.matrix(subset(test.data\_q2, select = y))  
  
set\_random\_seed(123)  
model <- keras\_model\_sequential()   
model %>% layer\_dense(units = 3, input\_shape = c(16)) %>%   
 layer\_dense(units = 1, activation = "linear")  
model %>% compile(loss='mse',optimizer='adam',metrics='mse')  
summary(model)

## Model: "sequential"  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## Layer (type) Output Shape Param #   
## ================================================================================  
## dense\_1 (Dense) (None, 3) 51   
## dense (Dense) (None, 1) 4   
## ================================================================================  
## Total params: 55  
## Trainable params: 55  
## Non-trainable params: 0  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

preds <- predict(model, test\_x)  
  
# scaled test RMSE  
RMSE(test.data$y, preds)

[1]12.6474

# test RMSE  
RMSE(test.data$y\*sd+mean, preds\*sd+mean)

[1] 10044.39

#PART 2 - Question 3  
detach(package:keras,unload=TRUE)  
detach(package:tensorflow,unload=TRUE)  
  
set.seed(123)  
  
# I kept getting this error so I commented out the parts to submit  
  
#Error in cut.default(y, breaks, include.lowest = TRUE) :  
#invalid number of intervals  
  
# model3\_q2 <- train(  
# y ~., data = train.data\_q2, method = "rf",  
# trControl = trainControl("cv", number = 10)  
# )  
# model3\_q2$bestTune  
#  
# predictions\_q2 <- model3\_q2 %>% predict(test.data\_q2)  
# #Add predicted class label to testing data  
# test.data\_q2.3 <- cbind(test.data\_q2, predictions\_q2)  
# str(test.data\_q2.3)  
#   
# # scaled test RMSE  
# RMSE(test.data\_q2$y, predictions\_q2)  
# # test RMSE  
# RMSE(test.data\_q2$y\*sd+mean, predictions\_q2\*sd+mean)

#PART 2 - Question 4  
library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following object is masked from 'package:bitops':  
##   
## %&%

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

## Loaded glmnet 4.1-6

# model\_q2.4 <- train(  
# y~., data = train.data\_q2, method = "glmnet",  
# trControl = trainControl("cv", number = 10),  
# tuneLength = 10  
# )  
# model\_q2.4$bestTune  
#   
# coef(model$finalModel, model$bestTune$lambda)

#PART 2 - Question 5  
# Didn't have time left, sorry