ams580\_final

Mustafa Yigit Isik

2023-05-17

# Part 1

## Loading data and packages  
if (!requireNamespace("tidyverse")) install.packages('tidyverse')

## Loading required namespace: tidyverse

if (!requireNamespace("caret")) install.packages('caret')

## Loading required namespace: caret

if (!requireNamespace("neuralnet")) install.packages('neuralnet')

## Loading required namespace: neuralnet

if (!requireNamespace("keras")) install.packages('keras')

## Loading required namespace: keras

if (!requireNamespace("randomForest")) install.packages('randomForest')

## Loading required namespace: randomForest

if (!requireNamespace("rpart")) install.packages('rpart')  
if (!requireNamespace("rattle")) install.packages('rattle')

## Loading required namespace: rattle

library(tidyverse)

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

## ✔ ggplot2 3.4.0 ✔ purrr 1.0.1   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0   
## ✔ readr 2.1.4 ✔ forcats 1.0.0   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(neuralnet)

##   
## Attaching package: 'neuralnet'  
##   
## The following object is masked from 'package:dplyr':  
##   
## compute

library(keras)  
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)

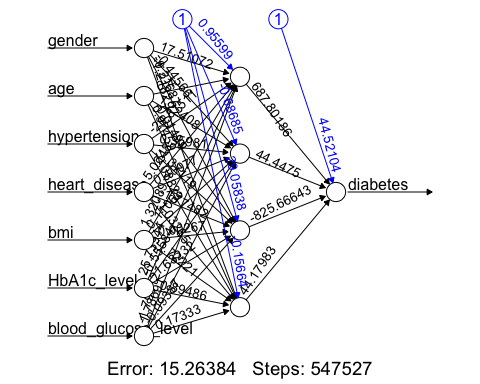
## Loading required package: bitops  
## Rattle: A free graphical interface for data science with R.  
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.  
##   
## Attaching package: 'rattle'  
##   
## The following object is masked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

## Question 1  
diabetes\_data <- read.csv('/Users/mustafayigitisik/Desktop/stuff/semesters/spring 2023/ams 580/final/diabetes\_prediction\_dataset\_updated.csv')  
diabetes\_data <- na.omit(diabetes\_data)  
diabetes\_data = subset(diabetes\_data, select = -c(smoking\_history))  
diabetes\_data$gender <- ifelse(diabetes\_data$gender == "Male", 1, 0)  
dim(diabetes\_data)[1]

## [1] 2000

set.seed(123)  
salary\_training.samples <- diabetes\_data$diabetes %>% createDataPartition(p = 0.75, list = FALSE)  
train.data <- diabetes\_data[salary\_training.samples, ]  
test.data <- diabetes\_data[-salary\_training.samples, ]

## Question 2  
### 2.a  
set.seed(123)  
model <- neuralnet(diabetes~., data = train.data, hidden = 4, err.fct = "sse",act.fct = "logistic", linear.output = F, stepmax = 1e7)  
plot(model, rep = "best")



probabilities <- model %>% predict(test.data) %>% as.vector()  
predicted.diabetes <- ifelse(probabilities > 0.5, 1, 0)  
confusionMatrix(factor(predicted.diabetes), factor(test.data$diabetes), positive = '1')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 441 24  
## 1 6 29  
##   
## Accuracy : 0.94   
## 95% CI : (0.9155, 0.9592)  
## No Information Rate : 0.894   
## P-Value [Acc > NIR] : 0.0002317   
##   
## Kappa : 0.6277   
##   
## Mcnemar's Test P-Value : 0.0019108   
##   
## Sensitivity : 0.5472   
## Specificity : 0.9866   
## Pos Pred Value : 0.8286   
## Neg Pred Value : 0.9484   
## Prevalence : 0.1060   
## Detection Rate : 0.0580   
## Detection Prevalence : 0.0700   
## Balanced Accuracy : 0.7669   
##   
## 'Positive' Class : 1   
##

### 2.b  
set.seed(123)  
#I don't know why but this bit doesn't compile, it takes so long and no progress.  
#model <- neuralnet(diabetes~., data = train.data, hidden = 4, err.fct = "ce", act.fct = "logistic", linear.output = F, stepmax = 1e7)  
#plot(model, rep = "best")  
  
#because of the problem above, can't run here as well  
#probabilities <- model %>% predict(test.data)  
#predicted.diabetes <- ifelse(probabilities > 0.5, 1, 0)  
#nn.diabetes = factor(predicted.diabetes)  
#confusionMatrix(factor(predicted.diabetes), factor(test.data$diabetes), positive = '1')

## Question 3

### 3.a

train.data$diabetes <- factor(train.data$diabetes)  
test.data$diabetes <- factor(test.data$diabetes)  
  
set.seed(123)  
model <- train(  
 diabetes ~., data = train.data, method = "rf",  
 trControl = trainControl("cv", number = 10),  
 importance = TRUE  
 )  
# Best tuning parameter  
model$bestTune

## mtry  
## 1 2

model$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: 2%  
## Confusion matrix:  
## 0 1 class.error  
## 0 1399 0 0.0000000  
## 1 30 71 0.2970297

(1399+71)/(1399+71+30+0) # Overall Accuracy

## [1] 0.98

71/101 # Sensitivity

## [1] 0.7029703

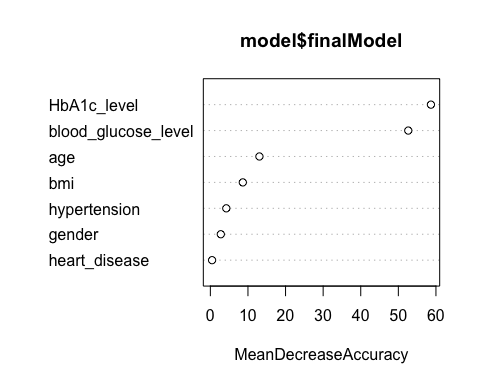
1399/1399 # Specificity

## [1] 1

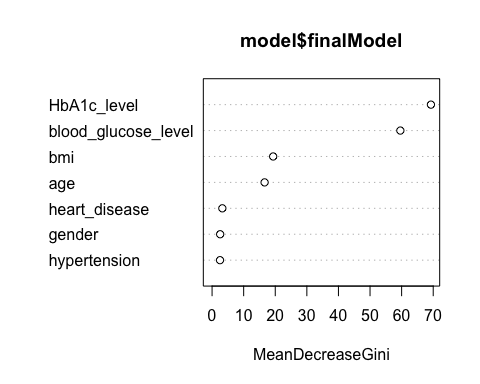
### 3.b  
pred <- model %>% predict(test.data)  
rf.diabetes = pred  
confusionMatrix(pred, test.data$diabetes, positive = '1')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 447 19  
## 1 0 34  
##   
## Accuracy : 0.962   
## 95% CI : (0.9413, 0.977)  
## No Information Rate : 0.894   
## P-Value [Acc > NIR] : 1.946e-08   
##   
## Kappa : 0.7619   
##   
## Mcnemar's Test P-Value : 3.636e-05   
##   
## Sensitivity : 0.6415   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9592   
## Prevalence : 0.1060   
## Detection Rate : 0.0680   
## Detection Prevalence : 0.0680   
## Balanced Accuracy : 0.8208   
##   
## 'Positive' Class : 1   
##

### 3.c  
# Plot MeanDecreaseAccuracy  
varImpPlot(model$finalModel, type = 1)



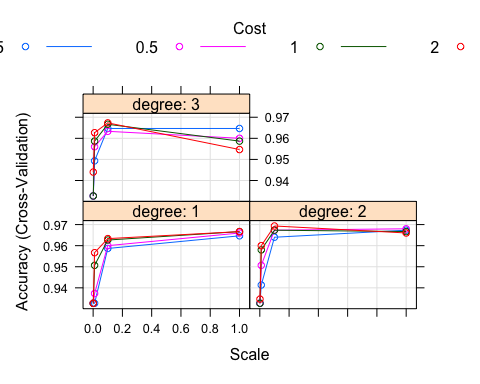
# Plot MeanDecreaseGini  
varImpPlot(model$finalModel, type = 2)



### 3.d  
varImp(model, type = 1)

## rf variable importance  
##   
## Overall  
## HbA1c\_level 100.000  
## blood\_glucose\_level 89.646  
## age 21.632  
## bmi 14.043  
## hypertension 6.495  
## gender 3.993  
## heart\_disease 0.000

## Question 4  
set.seed(123)  
model <- train(  
 diabetes ~., data = train.data, method = "svmPoly",  
 trControl = trainControl("cv", number = 4),  
 tuneLength = 4  
 )  
plot(model)



model$bestTune

## degree scale C  
## 28 2 0.1 2

svm.diabetes <- predict(model, newdata = test.data)  
confusionMatrix(svm.diabetes, test.data$diabetes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 447 22  
## 1 0 31  
##   
## Accuracy : 0.956   
## 95% CI : (0.9341, 0.9722)  
## No Information Rate : 0.894   
## P-Value [Acc > NIR] : 4.211e-07   
##   
## Kappa : 0.7159   
##   
## Mcnemar's Test P-Value : 7.562e-06   
##   
## Sensitivity : 1.0000   
## Specificity : 0.5849   
## Pos Pred Value : 0.9531   
## Neg Pred Value : 1.0000   
## Prevalence : 0.8940   
## Detection Rate : 0.8940   
## Detection Prevalence : 0.9380   
## Balanced Accuracy : 0.7925   
##   
## 'Positive' Class : 0   
##

## Question 5  
# had to comment out nn.diabetes due to compiling issues mentioned above  
#pred = cbind(rf.diabetes, svm.diabetes)  
#pred.m = apply(pred,1,function(x) names(which.max(table(x)))) # Majority vote  
#pred.m = factor(pred.m, levels = c('1','2'), labels = c('0','1'))  
  
#confusionMatrix(pred.m, test.data$diabetes, positive = '1')

# Part 2

## Loading data and packages  
if (!requireNamespace("tidyquant")) install.packages('tidyquant')

## Loading required namespace: tidyquant

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

if (!requireNamespace("magrittr")) install.packages('magrittr')  
if (!requireNamespace("tensorflow")) install.packages('tensorflow')  
if (!requireNamespace("zoo")) install.packages('zoo')  
library(tidyquant)

## Loading required package: lubridate

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

## Loading required package: PerformanceAnalytics

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

##   
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':  
##   
## first, last

##   
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':  
##   
## legend

## Loading required package: quantmod

## Loading required package: TTR

library(magrittr)

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':  
##   
## set\_names

## The following object is masked from 'package:tidyr':  
##   
## extract

library(tensorflow)

##   
## Attaching package: 'tensorflow'

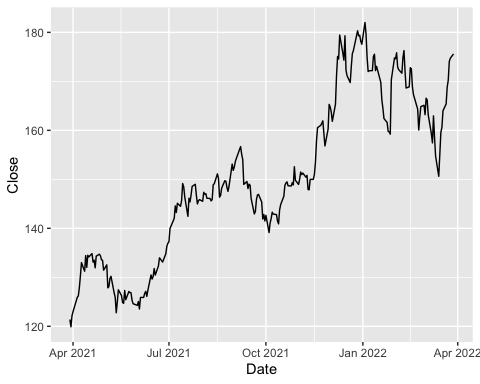
## The following object is masked from 'package:caret':  
##   
## train

library(zoo)

## Question 1  
data <- read.csv('/Users/mustafayigitisik/Desktop/stuff/semesters/spring 2023/ams 580/final/AAPL.csv', header = T)  
# transform the date from 'chr' to 'Date'  
data$Date = as.Date(data$Date)  
# visualize  
knitr::kable(head(data))

| Date | Open | High | Low | Close | Adj.Close | Volume |
| --- | --- | --- | --- | --- | --- | --- |
| 2021-03-29 | 121.65 | 122.58 | 120.73 | 121.39 | 120.6728 | 80819200 |
| 2021-03-30 | 120.11 | 120.40 | 118.86 | 119.90 | 119.1916 | 85671900 |
| 2021-03-31 | 121.65 | 123.52 | 121.15 | 122.15 | 121.4283 | 118323800 |
| 2021-04-01 | 123.66 | 124.18 | 122.49 | 123.00 | 122.2733 | 75089100 |
| 2021-04-05 | 123.87 | 126.16 | 123.07 | 125.90 | 125.1561 | 88651200 |
| 2021-04-06 | 126.50 | 127.13 | 125.65 | 126.21 | 125.4643 | 80171300 |

ggplot(data, aes(x=Date, y = Close)) + geom\_line()



# normalize the stock price by using the 'min-max scaler'  
data$min\_lagged = lag(data$Low)  
data$max\_lagged = lag(data$High)  
data$Close\_norm = (data$Close - data$min\_lagged) / (data$max\_lagged - data$min\_lagged)  
model\_data = matrix(data$Close\_norm[-1])  
# The last 10 scalded close prices  
knitr::kable(tail(model\_data,10))

|  |  |
| --- | --- |
| [243,] | 1.241294 |
| [244,] | 1.774564 |
| [245,] | 1.111913 |
| [246,] | 1.884273 |
| [247,] | 1.190680 |
| [248,] | 1.739519 |
| [249,] | 1.175169 |
| [250,] | 1.286574 |
| [251,] | 1.147584 |
| [252,] | 1.126485 |

## Question 2  
train\_data = head(model\_data,-10)  
test\_data = tail(model\_data, 20)  
cat(dim(train\_data)[1], ' days are divided into the training set.')

## 242 days are divided into the training set.

## Question 3  
prediction = 10  
lag = prediction  
# Training X  
# we lag the data 10 times and arrange that into columns  
train\_X = t(sapply(  
 1:(length(train\_data) - lag - prediction + 1),  
 function(x) train\_data[x:(x + lag - 1), 1]  
 ))  
# now we transform it into 3D form  
train\_X <- array(  
 data = as.numeric(unlist(train\_X)),  
 dim = c(  
 nrow(train\_X),  
 lag,  
 1  
 )  
)  
# Training y  
train\_y <- t(sapply(  
 (1 + lag):(length(train\_data) - prediction + 1),  
 function(x) train\_data[x:(x + prediction - 1)]  
))  
train\_y <- array(  
 data = as.numeric(unlist(train\_y)),  
 dim = c(  
 nrow(train\_y),  
 prediction,  
 1  
 )  
)  
# Testing X  
test\_X = t(sapply(  
 1:(length(test\_data) - lag - prediction + 1),  
 function(x) test\_data[x:(x + lag - 1), 1]  
 ))  
test\_X <- array(  
 data = as.numeric(unlist(test\_X)),  
 dim = c(  
 nrow(test\_X),  
 lag,  
 1  
 )  
)  
# Testing y  
test\_y <- t(sapply(  
 (1 + lag):(length(test\_data) - prediction + 1),  
 function(x) test\_data[x:(x + prediction - 1)]  
))  
test\_y <- array(  
 data = as.numeric(unlist(test\_y)),  
 dim = c(  
 nrow(test\_y),  
 prediction,  
 1  
 )  
)  
dim(train\_X)

## [1] 223 10 1

dim(train\_y)

## [1] 223 10 1

dim(test\_X)

## [1] 1 10 1

dim(test\_y)

## [1] 1 10 1

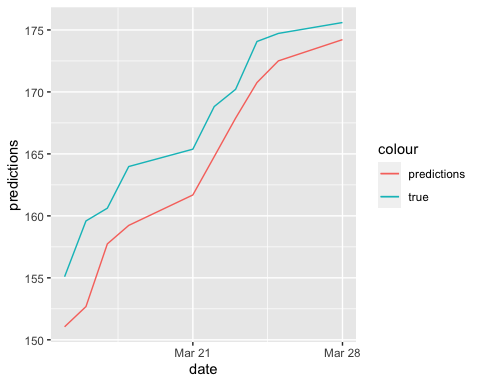
## Question 4  
set\_random\_seed(123)  
model <- keras\_model\_sequential()  
model %>%  
 layer\_lstm(units = 200, input\_shape = dim(train\_X)[2:3])  
model %>%  
 layer\_dense(units = dim(test\_y)[2])  
  
summary(model)

## Model: "sequential"  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## Layer (type) Output Shape Param #   
## ================================================================================  
## lstm (LSTM) (None, 200) 161600   
## dense (Dense) (None, 10) 2010   
## ================================================================================  
## Total params: 163,610  
## Trainable params: 163,610  
## Non-trainable params: 0  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

model %>% compile(loss = 'mse',  
 optimizer = 'adam',  
 metrics = 'mse')  
history <- model %>% fit(  
 x = train\_X,  
 y = train\_y,  
 batch\_size =16,  
 epochs = 50,  
 validation\_split = 0.1,  
 shuffle = FALSE  
)  
  
preds\_norm = t(predict(model, test\_X))  
preds\_complete = cbind(preds\_norm, tail(data, prediction))  
preds = preds\_complete$preds\_norm\*(preds\_complete$max\_lagged - preds\_complete$min\_lagged) + preds\_complete$min\_lagged  
predictions = data.frame(predictions = preds, true = preds\_complete$Close, date = preds\_complete$Date)  
# Test MSE  
(MSE.lstm = RMSE(predictions$true, predictions$predictions)^2)

## [1] 14.80454

# Plot forecast  
ggplot(data = predictions, aes(x = date)) +  
 geom\_line(aes(y = predictions, color = 'predictions')) +  
 geom\_line(aes(y = true, color = 'true'))



# stock price of 3/28/22  
preds\_norm1 = predict(model, test\_y)[1]  
(preds1 = preds\_norm1\*(data$High[252] - data$Low[252]) + data$Low[252])

## [1] 174.6274

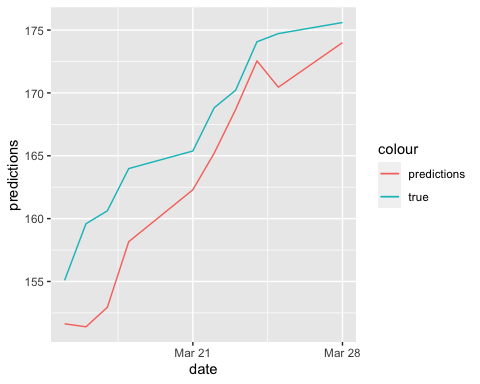
## Question 5  
set\_random\_seed(123)  
model <- keras\_model\_sequential()  
model %>%  
 layer\_simple\_rnn(units = 200, input\_shape = dim(train\_X)[2:3])  
model %>%  
 layer\_dense(units = dim(test\_y)[2])  
  
summary(model)

## Model: "sequential\_1"  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## Layer (type) Output Shape Param #   
## ================================================================================  
## simple\_rnn (SimpleRNN) (None, 200) 40400   
## dense\_1 (Dense) (None, 10) 2010   
## ================================================================================  
## Total params: 42,410  
## Trainable params: 42,410  
## Non-trainable params: 0  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

model %>% compile(loss = 'mse',  
 optimizer = 'adam',  
 metrics = c('mse'))  
history <- model %>% fit(  
 x = train\_X,  
 y = train\_y,  
 batch\_size =16,  
 epochs = 50,  
 validation\_split = 0.1,  
 shuffle = FALSE  
)  
  
preds\_norm = t(predict(model, test\_X))  
preds\_complete = cbind(preds\_norm, tail(data, prediction))  
preds = preds\_complete$preds\_norm\*(preds\_complete$max\_lagged - preds\_complete$min\_lagged) + preds\_complete$min\_lagged  
predictions = data.frame(predictions = preds, true = preds\_complete$Close, date = preds\_complete$Date)  
# Test MSE  
(MSE.rnn = RMSE(predictions$true, predictions$predictions)^2)

## [1] 21.98363

# Plot forecast  
ggplot(data = predictions, aes(x = date)) +  
 geom\_line(aes(y = predictions, color = 'predictions')) +  
 geom\_line(aes(y = true, color = 'true'))



# stock price of 2022-05-16  
preds\_norm1 = predict(model, test\_y)[1]  
(preds1 = preds\_norm1\*(data$High[252] - data$Low[252]) + data$Low[252])

## [1] 174.3665

## Question 6  
MSE.lstm

## [1] 14.80454

MSE.rnn

## [1] 21.98363

# Part 3

## Loading data and packages  
if (!requireNamespace("tidyverse")) install.packages('tidyverse')  
if (!requireNamespace("caret")) install.packages('caret')  
if (!requireNamespace("glmnet")) install.packages('glmnet')

## Loading required namespace: glmnet

if (!requireNamespace("caTools")) install.packages('caTools')

## Loading required namespace: caTools

library(tidyverse)  
library(caret)  
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

library(caTools)  
  
salarydata <-   
 read.csv('/Users/mustafayigitisik/Desktop/stuff/semesters/spring 2023/ams 580/final/ds\_salaries.csv', header = T)  
#salarydata <- subset(salarydata, select = -c(salary, salary\_currency))  
salarydata <- na.omit(salarydata)  
cat('There were', sum(is.na(salarydata)), 'missing rows.')

## There were 0 missing rows.

set.seed(123)  
salary\_training.samples <- salarydata$salary\_in\_usd %>% createDataPartition(p = 0.75, list = FALSE)  
train.salarydata <- salarydata[salary\_training.samples, ]  
test.salarydata <- salarydata[-salary\_training.samples, ]  
dim(train.salarydata) # 2817 obs

## [1] 2817 11

dim(test.salarydata) # 938 obs

## [1] 938 11

#Error in glmnet(x, y, weights = weights, offset = offset, lambda = lambda, :  
#number of observations in y (2817) not equal to the number of rows of x (938)  
  
  
#RIDGE  
# x <- model.matrix(salary~., test.salarydata)[,-1]   
# y <- train.salarydata$salary\_in\_usd  
# cv <- cv.glmnet(x, y, alpha = 0)  
# cv$lambda.min

# model <- glmnet(x, y, alpha = 0, lambda = cv$lambda.min) # alpha=0: ridge  
# coef(model)  
# ```  
# ```{r}  
# x.test <- model.matrix(salary\_in\_usd ~., test.salarydata)[,-1]  
# predictions <- model %>% predict(x.test) %>% as.vector()  
# data.frame(  
# RMSE = RMSE(predictions, test.salarydata$salary\_in\_usd),  
# Rsquare = R2(predictions, test.salarydata$salary\_in\_usd)  
# )  
# plot(y = predictions,x = test.salarydata$salary\_in\_usd, xlab='Observed response',ylab='Estimated response')  
# abline(1,1)

#LASSO  
# cv <- cv.glmnet(x, y, alpha = 1)  
# cv$lambda.min

# model <- glmnet(x, y, alpha = 1, lambda = cv$lambda.min) # alpha=1: lasso  
# coef(model)  
# ```  
# ```{r}  
# x.test <- model.matrix(salary\_in\_usd ~., test.salarydata)[,-1]  
# predictions <- model %>% predict(x.test) %>% as.vector()  
#   
# data.frame(  
# RMSE = RMSE(predictions, test.salarydata$salary\_in\_usd),  
# Rsquare = R2(predictions, test.salarydata$salary\_in\_usd)  
# )  
#   
# plot(y = predictions,x = test.salarydata$salary\_in\_usd, xlab='Observed predictions',ylab='Estimated response')  
# abline(1,1)

#ELASTIC  
# model <- train(  
# salary\_in\_usd ~., data = train.salarydata, method = "glmnet",  
# trControl = trainControl("cv", number = 10),  
# tuneLength = 10  
# )  
# model$bestTune

# coef(model$finalModel, model$bestTune$lambda)