R script:

```
# Aadhithya Dinesh
# MIS 545 Section 02
# Lab13DineshA.R
# to import a dataset of hotel reviews and conduct text-mining
sentiment
# analysis to predict if a review is good or bad based on the text.
# We will be importing csv files, assigning data types, performing
text-based
# data preprocessing, conducting sentiment analysis, and interpreting
the results.

# install.packages("Matrix")
# install.packages("Rcpp")
# install.packages("quanteda")
# install.packages("caret")
# install.packages("doSNOW")

library(tidyverse)
library(Matrix)
library(quanteda)
library(caret)
library(doSNOW)
library(Rcpp)

# set the working directory
setwd("~/MIS/Classes/MIS545/Assignments/Lab13")

hotelReviews <- read_csv(file = "HotelReviews1000.csv",
                         col_types = "ci",
                         col_names = TRUE)

# print the hotelReviews tibble
print(hotelReviews)

# print the structure of hotelReviews
print(str(hotelReviews))

# print the summary of hotelReviews
print(summary(hotelReviews))

# remove 3-star reviews from the dataset
hotelReviews <- hotelReviews %>% filter(Stars !=3)

# show a count of reviews by star rating
```

```
print(hotelReviews %>% count(Stars))

# create a new feature called Rating that hs value of bad or good
based on the
# star rating
hotelReviews <- hotelReviews %>%
  mutate(Rating = factor(ifelse(Stars == 1 | Stars == 2, "bad",
"good")))

# drop the original Stars feature
hotelReviews <- hotelReviews %>% select(-Stars)

# display a summary of the reviews table
summary(hotelReviews)

# drop records with missing data
hotelReviews <- drop_na(hotelReviews)

# add feature for number of characters in the review
hotelReviews <- hotelReviews %>%
  mutate(ReviewLength = nchar(Text))

# does review length correlate with good/bad review?
print(hotelReviews %>%
        group_by(Rating) %>%
        summarize(mean(ReviewLength)))

# tokenize the reviews
reviewTokens <- tokens( x= hotelReviews$Text,
                        what = "word",
                        remove_numbers = TRUE,
                        remove_punct = TRUE,
                        remove_symbols = TRUE,
                        split_hyphens = TRUE,
                        remove_url = TRUE)
# lowercase the tokens
reviewTokens <- tokens_tolower(reviewTokens)

# view a specific review
hotelReviews[506,]
reviewTokens[506]

# view a list of 175 predifined "stop words" in quanteda
print(stopwords())

# remove the stop words (the, a, and, as, but, if)
```

```
reviewTokens <- tokens_select(reviewTokens,
                              stopwords(),
                              selection = "remove")
# view a specific review
hotelReviews[506,]
reviewTokens[506]

# combine stemmed words in tokens (e.g run, runs, running)
reviewTokens <- tokens_wordstem(reviewTokens,
                                language = "english")

# view a specific review
hotelReviews[506,]
reviewTokens[506]

# generate a document feature matrix
reviewTokensDFM <- dfm(reviewTokens)

# conver the dfm into a matrix
reviewTokensMatrix <- as.matrix(reviewTokensDFM)

# view the dimensions of reviewToknesMatrix
print(dim(reviewTokensMatrix))

# view a subset of the matrix ( first 20 rows and 100 columns)
View(reviewTokensMatrix[1:20, 1:100])

# generate a feature data frame with labels
reviewTokensDataFrame <- cbind(Label = hotelReviews$Rating,
                               data.frame(reviewTokensDFM))

# clean column names to prevent errors with invalid labels
names(reviewTokensDataFrame) <-
make.names(names(reviewTokensDataFrame))

# set a random seed
set.seed(511)

# set up the stratified cross validation parameters
crossValidationFolds <- createMultiFolds(y= hotelReviews$Rating,
                                         k =10,
                                         times = 3)
# setup the training process
crossValidationControl <- trainControl(method = "repeatedcv",
                                       number = 10,
                                       repeats = 3,
```

```
                                        index = crossValidationFolds)

# create a cluster to work on 3 logical cores
cluster <- makeCluster(7, type = "SOCK")
registerDoSNOW(cluster)

# single decision tree algorithm
reviewDecisionTree <- train(label ~.,
                            data = reviewTokensDataFrame,
                            method = "rpart",
                            trControl = crossValidationControl,
                            tuneLength = 7)

# stop cluster when processing is finished
stopCluster(cluster)

# view results
print(reviewDecisionTree)
```

Answers:

1. The **complexity parameter of 0.1418742** yielded the highest accuracy based on the lowest **Root Mean Squared Error (RMSE) value of 0.0118.**
2. **Over fitting** is one of the biggest downsides to using the same data for training and testing the model. We will not know how this model is going to perform if it is fed with data not seen before.
3. Being able to analyze hundreds of thousands of reviews in a span of seconds and deciding if the overall sentiment is good or bad can have huge potential in seeing if hotel is doing well in general. This can further be used to advertise the hotel if the result turns out to be good or lead to further analysis of what's going wrong if the sentiment turns out to be bad.