~~~
---
output:
  word_document: default
  pdf_document: default
  html_document:
    df_print: paged
---


1)


```{r}

#install.packages('devtools')

#remove.packages('rlang')

#install.packages('rlang')

library(devtools)
#slam_url <- "https://cran.r-project.org/src/contrib/Archive/slam/
slam_0.1-37.tar.gz"
#install_url(slam_url)
install.packages("tm")
install.packages("SnowballC")
install.packages("wordcloud")
install.packages("tm.plugin.webmining")
library(dplyr)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
library(tm.plugin.webmining)
library(tm)
library(SnowballC)
library(wordcloud)
library(MASS)
library(caTools)

```

```{r}



tableAccuracy <- function(test, pred) {
  t = table(test, pred)
  a = sum(diag(t))/length(test)
  return(a)
}


questions = read.csv("ggplot2questions2016_17.csv", stringsAsFactors=FALSE)
~~~

```
questions$Useful = as.factor(as.numeric(questions$Score >= 1))
questions$Score <- NULL


table(questions$Useful)

corpus_text = Corpus(VectorSource(questions$Title))
corpus_body = Corpus(VectorSource(questions$Body))



corpus_text = tm_map(corpus_text, tolower)
# Lets check:
strwrap(corpus_text[[1]])

corpus_body = tm_map(corpus_body, tolower)
# Lets check:
strwrap(corpus_body[[1]])

# Step 2.5: Remove Tags
clean <- function(HTML){
  return(gsub("<.*?>", "", HTML))
}
corpus_body = tm_map(corpus_body, clean)


# Step 3: Remove all punctuation
corpus_text = tm_map(corpus_text, removePunctuation)
# Take a look:
strwrap(corpus_text[[1]])

corpus_body = tm_map(corpus_body, removePunctuation)
# Take a look:
strwrap(corpus_body[[1]])

# Step 4: Remove stop words
# First, take a look at tm's stopwords:
stopwords("english")[1:10]
length(stopwords("english"))
# Just remove stopwords:
# corpus = tm_map(corpus, removeWords, stopwords("english"))
# Remove stopwords and "apple" - this is a word common to all of our tweets
corpus_text = tm_map(corpus_text, removeWords, c(stopwords("english")))
# Take a look:
strwrap(corpus_text[[1]])

corpus_body = tm_map(corpus_body, removeWords, c(stopwords("english")))
# Take a look:
strwrap(corpus_body[[1]])

# Step 5: Stem our document
# Recall, this means chopping off the ends of words that aren't maybe
# as necessary as the rest, like 'ing' and 'ed'
corpus_text = tm_map(corpus_text, stemDocument)
# Take a look:
strwrap(corpus_text[[1]])

corpus_body = tm_map(corpus_body, stemDocument)
# Take a look:
strwrap(corpus_body[[1]])

# Step 6: Create a word count matrix (rows are tweets, columns are words)
# We've finished our basic cleaning, so now we want to calculate frequencies
# of words across the tweets
frequencies_text = DocumentTermMatrix(corpus_text)

frequencies_body = DocumentTermMatrix(corpus_body)


# Step 7: Account for sparsity
# We currently have way too many words, which will make it hard to train
# our models and may even lead to overfitting.
# Use findFreqTerms to get a feeling for which words appear the most

# Words that appear at least 50 times:
findFreqTerms(frequencies_text, lowfreq=50)
# Words that appear at least 20 times:
findFreqTerms(frequencies_text, lowfreq=20)

findFreqTerms(frequencies_body, lowfreq=50)
# Words that appear at least 20 times:
findFreqTerms(frequencies_body, lowfreq=20)

# Our solution to the possibility of overfitting is to only keep terms
# that appear in x% or more of the tweets. For example:
# 1% of the tweets or more (= 12 or more)
sparse_text = removeSparseTerms(frequencies_text, 0.99)
sparse_body = removeSparseTerms(frequencies_body, 0.99)

# How many did we keep?
sparse_text
sparse_body

# Step 8: Create data frame from the document-term matrix
questionsTM_text = as.data.frame(as.matrix(sparse_text))
questionsTM_body = as.data.frame(as.matrix(sparse_body))

# We have some variable names that start with a number,
# which can cause R some problems. Let's fix this before going
# any further
colnames(questionsTM_text) = make.names(colnames(questionsTM_text))
colnames(questionsTM_body) = make.names(colnames(questionsTM_body))

questionsTM <- merge(questionsTM_text, questionsTM_body, by=0)
#questionsTM <- merge(questionsTM_text, questionsTM_body)
# This isn't our original dataframe, so we need to bring that column
# with the dependent variable into this new one
questionsTM$Useful = questions$Useful


set.seed(7)
spl = sample.split(questionsTM$Useful, SplitRatio = 0.7)

questionsTrain = questionsTM %>% filter(spl == TRUE)
questionsTest = questionsTM %>% filter(spl == FALSE)


table(questionsTrain$Useful)
table(questionsTest$Useful)

questionsTrain$Row.names = NULL
questionsTest$Row.names = NULL

questionsRF = randomForest(Useful ~ ., data=questionsTrain)

PredictRF = predict(questionsRF, newdata = questionsTest)
table(questionsTest$Useful, PredictRF)
tableAccuracy(questionsTest$Useful, PredictRF)


# Cross-validated CART model
set.seed(7)
train.cart = train(Useful ~ .,
                   data = questionsTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp=seq(0, 0.03, 0.002)),
                   trControl = trainControl(method="cv", number=5))
train.cart
train.cart$results

ggplot(train.cart$results, aes(x = cp, y = Accuracy)) + geom_point(size = 2) +
geom_line() +
  ylab("CV Accuracy") + theme_bw() +
  theme(axis.title=element_text(size=18), axis.text=element_text(size=18))

mod.cart = train.cart$finalModel
prp(mod.cart)

predict.cart = predict(mod.cart, newdata = questionsTest, type = "class")
table(questionsTest$Useful, predict.cart)
tableAccuracy(questionsTest$Useful, predict.cart)

questionsLog = glm(Useful ~ ., data = questionsTrain, family = "binomial")
summary(questionsLog)

# Predictions on test set
PredictLog = predict(questionsLog, newdata = questionsTest, type = "response")
table(questionsTest$Useful, PredictLog > 0.5)
tableAccuracy(questionsTest$Useful, PredictLog > 0.5)
```

~~~
# But what about training set?
PredictLogTrain = predict(questionsLog, type = "response")
table(questionsTrain$Useful, PredictLogTrain > 0.3)
tableAccuracy(questionsTrain$Useful, PredictLogTrain > 0.3)

# What about CART on training set?
PredictCARTTrain = predict(mod.cart, type = "class")
table(questionsTrain$Useful, PredictCARTTrain)
tableAccuracy(questionsTrain$Useful, PredictCARTTrain)
# Quite similar to test set performance



library(MASS)
lda.mod = lda(Useful ~ ., data = questionsTrain)

predict.lda = predict(lda.mod, newdata = questionsTest)$class
table(questionsTest$Useful, predict.lda)
tableAccuracy(questionsTest$Useful, predict.lda)




```
~~~