Conner Capdau

Due: April 26, 2019

```
# needed packages
library(tm)

## Loading required package: NLP

library(e1071)
library(mda)

## Loading required package: class

## Loaded mda 0.4-10
```

Explore data

```
# read in data and check summary of data
data = read.csv("Womens Clothing E-Commerce Reviews.csv", stringsAsFactors=F)
summary(data)

##        X          Clothing.ID          Age          Title
##  Min.   :    0   Min.   :   0.0   Min.   :18.0   Length:23486
##  1st Qu.: 5871   1st Qu.: 861.0   1st Qu.:34.0   Class :character
##  Median :11742   Median : 936.0   Median :41.0   Mode  :character
##  Mean   :11742   Mean   : 918.1   Mean   :43.2
##  3rd Qu.:17614   3rd Qu.:1078.0   3rd Qu.:52.0
##  Max.   :23485   Max.   :1205.0   Max.   :99.0
##  Review.Text            Rating      Recommended.IND
##  Length:23486       Min.   :1.000   Min.   :0.0000
##  Class :character   1st Qu.:4.000   1st Qu.:1.0000
##  Mode  :character   Median :5.000   Median :1.0000
##                     Mean   :4.196   Mean   :0.8224
##                     3rd Qu.:5.000   3rd Qu.:1.0000
##                     Max.   :5.000   Max.   :1.0000
##  Positive.Feedback.Count Division.Name      Department.Name
##  Min.   :  0.000         Length:23486       Length:23486
##  1st Qu.:  0.000         Class :character   Class :character
##  Median :  1.000         Mode  :character   Mode  :character
##  Mean   :  2.536
##  3rd Qu.:  3.000
##  Max.   :122.000
##   Class.Name
##  Length:23486
##  Class :character
##  Mode  :character
```

```
##
##
##

# X can be removed because it is only the row numbers of the csv
data$X = NULL

# check structure of the variables
str(data)
```

```
## 'data.frame':    23486 obs. of  10 variables:
##  $ Clothing.ID            : int  767 1080 1077 1049 847 1080 858 858 1077
1077 ...
##  $ Age                    : int  33 34 60 50 47 49 39 39 24 34 ...
##  $ Title                  : chr  "" "" "Some major design flaws" "My
favorite buy!" ...
##  $ Review.Text            : chr  "Absolutely wonderful - silky and sexy
and comfortable" "Love this dress!  it's sooo pretty.  i happened to find it
in a store, and i'm glad i did bc i never would have"| __truncated__ "I had
such high hopes for this dress and really wanted it to work for me. i
initially ordered the petite small "| __truncated__ "I love, love, love this
jumpsuit. it's fun, flirty, and fabulous! every time i wear it, i get nothing
but great compliments!" ...
##  $ Rating                 : int  4 5 3 5 5 2 5 4 5 5 ...
##  $ Recommended.IND        : int  1 1 0 1 1 0 1 1 1 1 ...
##  $ Positive.Feedback.Count: int  0 4 0 0 6 4 1 4 0 0 ...
##  $ Division.Name          : chr  "Initmates" "General" "General" "General
Petite" ...
##  $ Department.Name        : chr  "Intimate" "Dresses" "Dresses" "Bottoms"
...
##  $ Class.Name             : chr  "Intimates" "Dresses" "Dresses" "Pants"
...
```

```
# change Recommended.IND to be factor instead of integer, and
division/class/department to factor instead of characters
data$Recommended.IND = as.factor(data$Recommended.IND)
data$Division.Name = as.factor(data$Division.Name)
data$Department.Name = as.factor(data$Department.Name)
data$Class.Name = as.factor(data$Class.Name)

# I'm just curious to how many different articles of clothing that are
reviewed
length(unique(data$Clothing.ID)) # there are 1179
```

```
## [1] 1206
```

```
# since we'll be using reviews for predicting, make sure there are no empty
reviews
length(which(data$Review.Text==""))
```

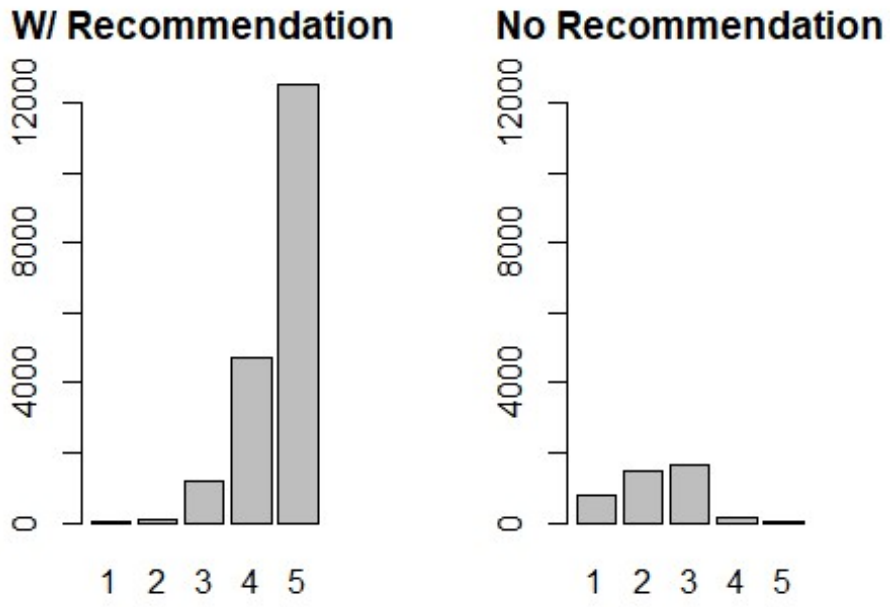
```
## [1] 845
```

```
# there are 845 empty reviews those reviews need to be removed
data = data[!(data$Review.Text==""),]
```

When looking at the summary of data, we see variable 'X' is not needed because it only marks the row numbers from the CSV. We also see that 82% of reviews recommend the clothing the review is for. We also see the majority of ratings are 5 (since the median is equal to the max value), and the mean rating is 4.2. This makes sense given the high percentage of recommendations from the reviews.

```
# view barplot of ratings based on recommended or not
par(mfrow=c(1,2)) # to see plots side-by-side

barplot(table(data$Rating[which(data$Recommended.IND==1)]), main="W/
Recommendation", ylim=c(0,12000))
barplot(table(data$Rating[which(data$Recommended.IND==0)]), main="No
Recommendation", ylim=c(0,12000))
```




```
# reset to have one plot shown at a time
par(mfrow=c(1,1))
```

We can see from the barplots that almost all 4 and 5 ratings resulted in recommendations, 1 and 2 ratings were almost always no recommendation, and 3 ratings were fairly evenly split between recommending and not recommending.

Text Analysis

```
# create corpus of words from data
corpus_text = Corpus(VectorSource(data$Review.Text))

# make all words lowercase
corpus_text = tm_map(corpus_text, tolower)

## Warning in tm_map.SimpleCorpus(corpus_text, tolower): transformation drops
## documents

# remove punctuation
corpus_text = tm_map(corpus_text, removePunctuation)

## Warning in tm_map.SimpleCorpus(corpus_text, removePunctuation):
## transformation drops documents

# remove stop words
corpus_text = tm_map(corpus_text, removeWords, c(stopwords("english")))

## Warning in tm_map.SimpleCorpus(corpus_text, removeWords,
## c(stopwords("english"))): transformation drops documents

# remove numbers
corpus_text = tm_map(corpus_text, removeNumbers)

## Warning in tm_map.SimpleCorpus(corpus_text, removeNumbers): transformation
## drops documents

# remove extra white space
corpus_text = tm_map(corpus_text, stripWhitespace)

## Warning in tm_map.SimpleCorpus(corpus_text, stripWhitespace):
## transformation drops documents

# stem words
corpus_text = tm_map(corpus_text, stemDocument, language="english")

## Warning in tm_map.SimpleCorpus(corpus_text, stemDocument, language =
## "english"): transformation drops documents

# the line below shows how the first review looks after the above changes
print(as.character(corpus_text[[1]]))

## [1] "absolut wonder silki sexi comfort"

# convert corpus into a term matrix
corpus_dtm = DocumentTermMatrix(corpus_text)

# check dimensions of document term matrix
dim(corpus_dtm) # there are 13,542 words

## [1] 22641 13542
```

```r
# the number of words need to be simplified by keeping higher frequency words
corpus_final = removeSparseTerms(corpus_dtm, sparse=0.9)
dim(corpus_final) # only has 39 words; much more manageable
```

```
## [1] 22641    39
```

Classification with SVM

```r
# split the data between a training and testing set; along with splitting y
from corpus words input for training and testing
set.seed(5) # set seed for reproducability
train_index = sample(1:nrow(data), nrow(data)*0.8, replace=F)
train = as.matrix(corpus_final[train_index,])
train_y = data$Recommended.IND[train_index]
test = as.matrix(corpus_final[-train_index,])
test_y = data$Recommended.IND[-train_index]

# create model using svm
model.svm = svm(train_y ~ ., data = train)
# run prediction using svm model
pred.svm = predict(model.svm, test)

# check results
results = confusion(pred.svm, test_y); results
```

```
##           true
## predicted    0    1
##         0  124   61
##         1  697 3647
```

```r
# accuracy
100*(3647+124)/(124+61+697+3647)
```

```
## [1] 83.26341
```

```r
sum(diag(results)) / sum(results)
```

```
## [1] 0.8326341
```

```r
# 0.833

# accuracy if 1 was always guessed
(3647+61)/4529
```

```
## [1] 0.8187238
```

```r
# 0.819
```

The results show the model is 83.3% accurate which sounds good until you compare it with how most of the reviews are recommended. If a model were to always predict a review is recommended then the model would be 81.9% accurate. At least this model is slightly more accurate than always predicting Recommended.IND equal to 1.