Text based analysis to determine the usefulness of an online review

Rob Rupprath rob@omega.net (mailto:rob@omega.net) / rkr2@illinois.edu (mailto:rkr2@illinois.edu)

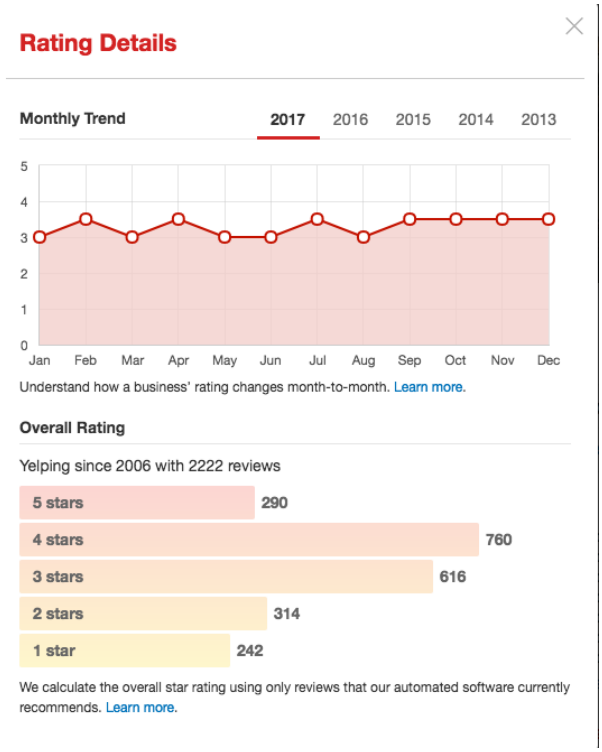
UIUC CS410

Synopsis

User reviews on the internet are becoming increasingly important for both buyers and sellers of goods and services. Reviews can now be more important for some business than traditional advertising. For example, a recent paper from the Harvard Business School concluded that an increase by a single ‘star’ on a ratings system will equate to approximately a 9% increase in overall revenues (http://www.hbs.edu/faculty/Pages/item.aspx?num=41233 (http://www.hbs.edu/faculty/Pages/item.aspx?num=41233)).

This is especially important for smaller, independent businesses. Franchise or chain businesses often see the primary value of the franchise in the brand popularity. Smaller or independent businesses can compete effectively if they have better ratings than the franchised businesses. Reviews are also important for users who can often be overwhelmed by too many options. Filtering by selected criteria allows users to quickly narrow down the choices they might be interested in.

There have been many data analytics projects that create models to predict the rating of a review, which is often useful. It is difficult however for users to quickly evaluate a business that has mixed reviews. For example, the business below was chosen at random. We can see that most of the reviews center around three or four stars, but several hundred reviews are in the one, two, or even five star rating.



It would be natural for the user to assume the quality of the product for this business is between three and four stars. However, it is also quitely likely that useful information exists in the one, two, and five star ratings. With over two thousand reviews, a user would struggle to pick out the key comments that are useful regardless of the

rating.

To further illustrate this concept, consider the following review examples that would likely not be useful to a customer:

- (Five Stars) - This place was great!
- (Three Stars) - It was ok
- (Two Stars) - I hated it

Now consider the following reviews which are at different ratings, but all would be considered useful for a customer searching for a restaurant.

- (Five Stars) - Choose the kale salad. It is the best!
- (Three Stars) - I went early on a Friday and still had to wait an hour because I didn't have a reservation
- (One Star) - They charge $20 for parking! How ridiculous!

The goal of this project is to build a predictive model which will indicate the usefulness of the review.

About the data

The dataset used for this project is provided by Yelp.com. Yelp was chosen as they are one of the most popular review sites online, but more importantly because they publish a large sample of their dataset online for educational use. The full dataset can be accessed at:

https://www.yelp.com/dataset (https://www.yelp.com/dataset)

** Please note that the R markdown used in this analysis caches many code segments that would take significant time to process. The cache for these segments has been included in the repository. If you choose to run the analysis without cache then it may take 10+ hours to complete **

Setting up the environment

In this section we will initialize the environment by clearing anything in memory, setting up the intial variables and loading the packages we will need

```
# Clear the environment
rm(list=ls())

# Set the working directory
setwd('/Users/robrupprath/Dropbox/Coursera/UIUC/CS410/project')

# Load the necessary libraries, using pacman as a package manager
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(openxlsx, sqldf, jsonlite, readr, dplyr, stringr, sqldf, sentimentr, ggpl
ot2, dplyr, reshape, textshape, rpart, randomForest, caret, scales)

# Variables
observationNumber <- 1000000      # Number of reviews to load
```

Reading the data

The entire dataset contains 4.7 million records, of which we will analyze a smaller subset to optimize processing time. We are making the assumption that the reviews in the Yelp data have been ordered randomly and that utilizing the first million records will not result in a selection bias. If given greater processing power and memory, one could conduct the analysis on the entire data set.

```
paste('For this analysis we will utilize the first ', observationNumber, ' results.', se
p = '')
```

```
## [1] "For this analysis we will utilize the first 1e+06 results."
```

```
# Read the data
reviewData <- read_lines(file = './dataset/review.json', n_max = observationNumber, prog
ress = FALSE)
reviewData <- str_c("[", str_c(reviewData, collapse = ", "), "]")

reviewData <- fromJSON(reviewData) %>%
  flatten() %>%
  tbl_df()
```

Initial Data Filters

Yelp utilized observed learning to tag reviews as useful. For example, each user can declare any particular review as useful. There is not a method to declare a review as not useful. From this we can conclude that a review that has a usefulness score of 0 may still actually be useful. For example, it could be a brand new review, or it could be a review that has not been observed or rated in the past.

We will subset the dataset by date range in order to exclude data that might be considered too new to evaluate, or data that is too old and perhaps was completed prior to the common use of Yelp as a recommendation system. The Yelp dataset begins in 2005 and ends in 2017. For our evaluation we will utilize reviews placed between 2012 and 2016.

```
reviewData$date <- as.Date(reviewData$date)
reviewData <- subset(reviewData, date >= '2012-01-01')
reviewData <- subset(reviewData, date <= '2017-01-01')

paste('After subsetting for the applied date range we have ', nrow(reviewData), ' observ
ations', sep = '')
```

```
## [1] "After subsetting for the applied date range we have 718828 observations"
```

Data Manipulation

In this section, we will manipulate the data to make a few additional observations:

- Number of words in the text review
- Number of reviews completed by the reviewer

```
# Add a column showing the length of the review
reviewData$numberOfWords <- sapply(gregexpr("\\W+", reviewData$text), length) + 1

# Add a column showing the number of reviews the user has completed
userReviewCount <- sqldf('
  select user_id, count(user_id) as count
  from reviewData
  group by user_id'
)

fnFindReviewCount <- function(user_id){
  reviewCount <- userReviewCount$count[userReviewCount$user_id == user_id]
  return(reviewCount)
}

reviewData$numberOfReviews <- sapply(reviewData$user_id, function(x) fnFindReviewCount
(x))

# Set data classes
reviewData$numberOfWords <- as.numeric(reviewData$numberOfWords)
reviewData$numberOfReviews <- as.numeric(reviewData$numberOfReviews)
reviewData$text <- as.character(reviewData$text)

# Create a new column for the numberOfReviewsDescription
reviewData$numberOfReviewsDescription <- NA
reviewData$numberOfReviewsDescription[reviewData$numberOfReviews == 1] <- '1 - Single'
reviewData$numberOfReviewsDescription[reviewData$numberOfReviews == 2] <- '2 - Two'
reviewData$numberOfReviewsDescription[reviewData$numberOfReviews > 2] <- '3 - More than
 two'

# Remove a few columns we no longer need to tidy up the dataset
reviewData$review_id <- NULL
reviewData$business_id <- NULL
reviewData$funny <- NULL
reviewData$cool <- NULL

# Clean up
rm(userReviewCount, fnFindReviewCount)
```

The number of words may be a good identifier on the overall usefulness of a review. For example, do users percieve a review which is too short or too long as not being useful? In order for this to be meaningful we need to catagorize the reviews based on review length. We will break the review length into four groups based on the average number of words:

- Short
- Medium
- Long
- Very Long

```
# Separate the review length into four groups
quantileReport <- as.data.frame(quantile(reviewData$numberOfWords))
reviewData$reviewLength <- NA
reviewData$reviewLength[reviewData$numberOfWords < quantileReport[2,1]] <- '1 - Short'
reviewData$reviewLength[reviewData$numberOfWords >= quantileReport[2,1] & reviewData$num
berOfWords < quantileReport[3,1]] <- '2 - Medium'
reviewData$reviewLength[reviewData$numberOfWords >= quantileReport[3,1] & reviewData$num
berOfWords < quantileReport[4,1]] <- '3 - Long'
reviewData$reviewLength[reviewData$numberOfWords >= quantileReport[4,1]] <- '4 - Very Lo
ng'

# Clean up
rm(quantileReport)
```

Initial data observations

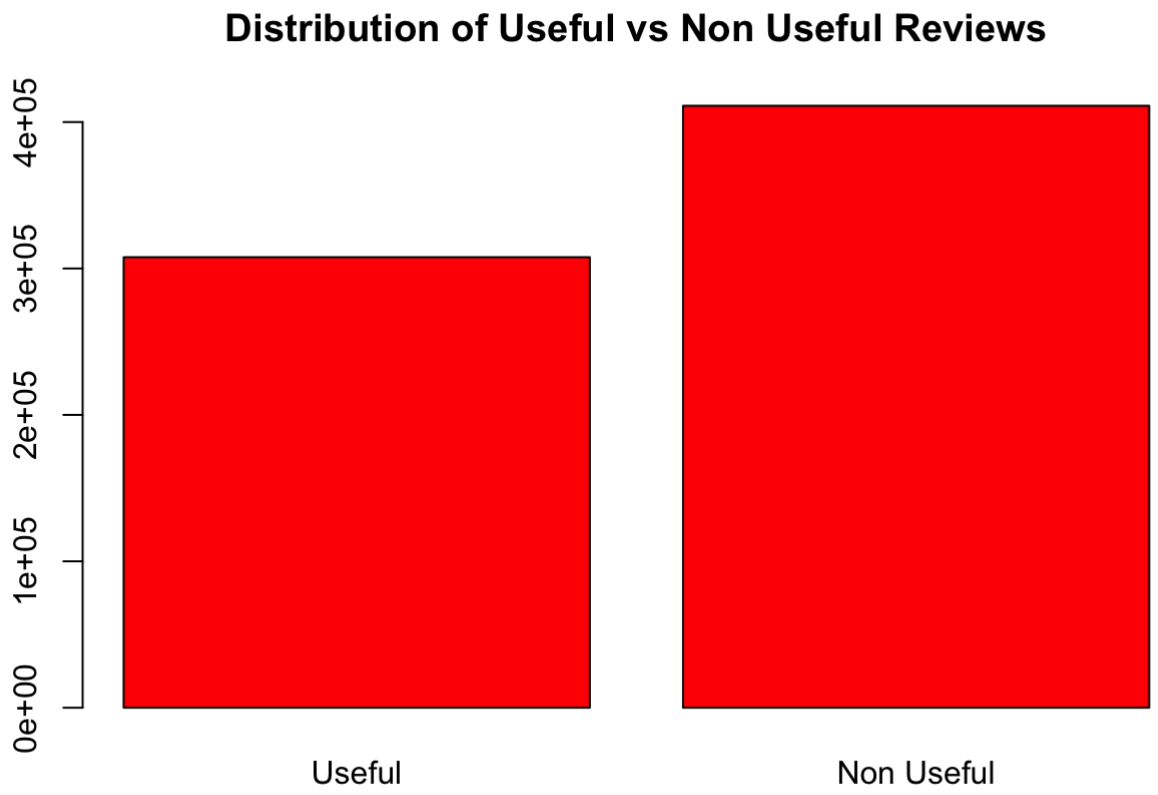
We now have some additional data points and will look at some summarizations prior to creating our model. Remember that the observed learning model that Yelp chose to implement does not permit for marking a review as not useful (negative usefullness scoring) and only allows users to tag a review as useful (positive usefulness scoring). We must remember that a review with a useful score of 0 is not necessarily a non-useful review.

Given this distribution we will add a column that indicates whether the review was useful or not based on whether the review has a score greater than zero.

```
usefulReviews <- length(nrow(reviewData)[reviewData$useful > 0])
nonUsefulReviews <- length(nrow(reviewData)[reviewData$useful == 0])

paste('We have a reasonable distribution of reviews tagged as useful (', usefulReviews,
') and not useful (', nonUsefulReviews, ')', sep = '')
```

```
## [1] "We have a reasonable distribution of reviews tagged as useful (307705) and not u
seful (411123)"
```

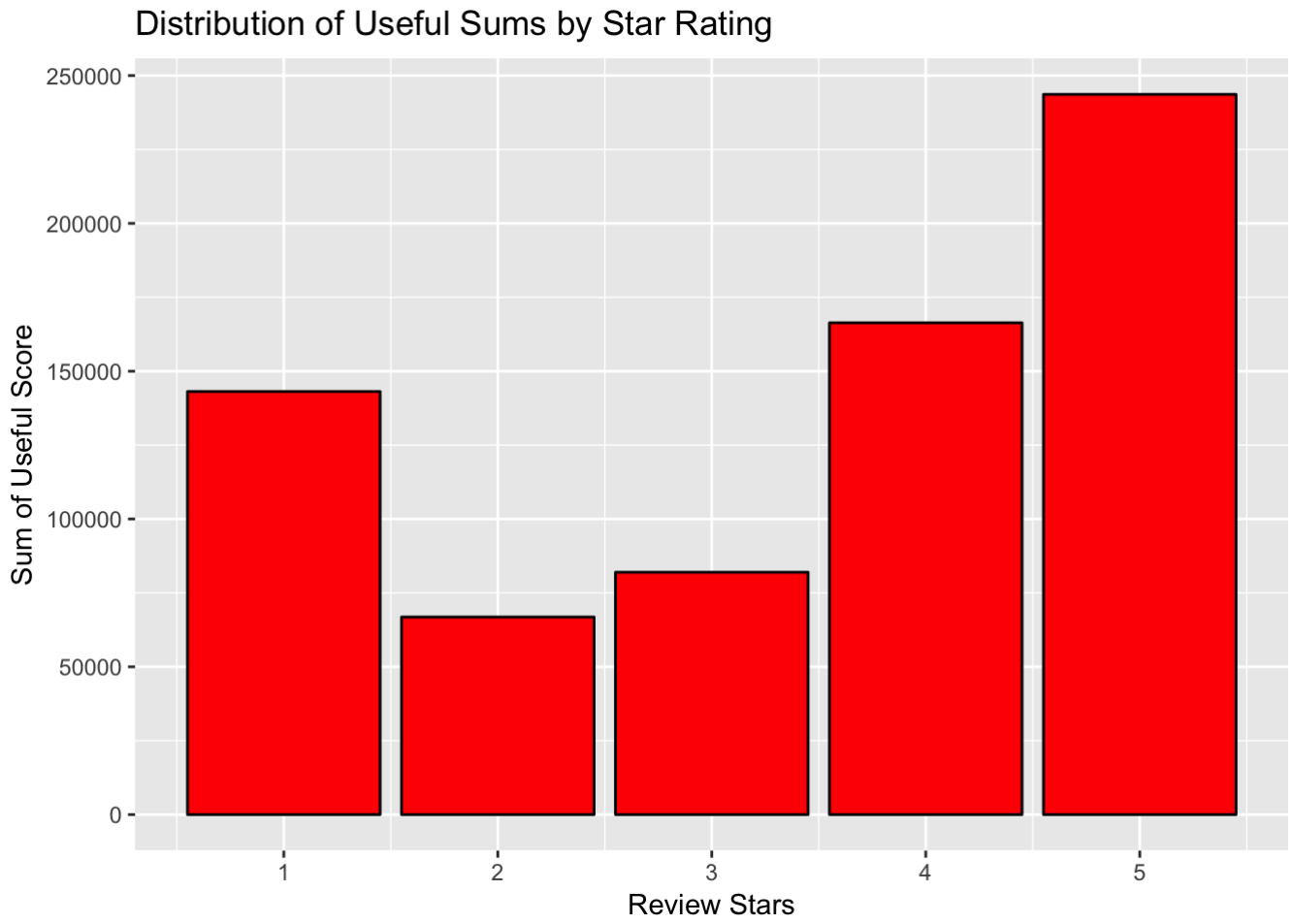
```
# Basic bar plot
tmpTable <- c(usefulReviews, nonUsefulReviews)
barplot(tmpTable, col = 'red', names.arg = c('Useful', 'Non Useful'), main = 'Distributi
on of Useful vs Non Useful Reviews')
```




```
# Clean up
rm(tmpTable, usefulReviews, nonUsefulReviews)

# Add useful column
reviewData$isUseful <- NA
reviewData$isUseful[reviewData$useful == 0] <- '1 - Not Useful'
reviewData$isUseful[reviewData$useful > 0] <- '2 - Useful'
```

We can also see that while there is some skewing in the distribution of reviews by the star rating, there is clearly a significant amount of useful data across all star ratings.

```
# Bar plot of review / star combinations
reviewStarCombinationData <- sqldf('
                                   select stars, sum(useful) as useful
                                   from reviewData
                                   group by stars
                                   order by stars asc
                                   ')

ggplot(data = reviewStarCombinationData, aes(x = stars, y = useful)) +
    geom_bar(color = 'black', fill = 'red', stat = 'identity') +
    guides(fill=FALSE) +
    xlab('Review Stars') + ylab('Sum of Useful Score') +
    ggtitle('Distribution of Useful Sums by Star Rating')
```




```
knitr::kable(reviewStarCombinationData)
```

stars	useful
1	143129
2	66824
3	81978
4	166375
5	243646

```
# Clean up
rm(reviewStarCombinationData)
```

Sentiment analysis

We will now conduct a basic sentiment analysis for each text review. The sentiment analysis is derived using the sentimentr package (https://cran.r-project.org/web/packages/sentimentr/README.html (https://cran.r-project.org/web/packages/sentimentr/README.html)). This package was chosen because it utilizes valence shifters (accelerators and deccelerators) in evaluating the text. For example, consider the following text sentiments:

- I like it (Baseline)

- I really like it (Accelerated)
- I barely like it (Decellerated)

We will also quantile the sentiments into four groups:

- Low
- Medium
- High
- Very High

```
# Create a new data frame for the sentiment
sentiments <- sentiment(reviewData$text)
```

```
## Warning in split_warn(text.var, "sentiment", ...): Each time `sentiment`
## is run it has to do sentence boundary disambiguation when a raw `character`
## vector is passed to `text.var`. This may be costly of time and memory. It
## is highly recommended that the user first runs the raw `character` vector
## through the `get_sentences` function.
```

```
sentiments <- sqldf('
  select element_id, avg(sentiment) as sentiment
  from sentiments
  group by element_id
  order by element_id asc
                  ')
sentiments$element_id <- NULL

# Join the sentiments to the reviewData table
reviewData <- cbind(reviewData, sentiments)

# Break the sentiment up into four groups
quantileReport <- as.data.frame(quantile(reviewData$sentiment))
reviewData$sentimentDescription <- NA
reviewData$sentimentDescription[reviewData$sentiment < quantileReport[2,1]] <- '1 - Low'
reviewData$sentimentDescription[reviewData$sentiment >= quantileReport[2,1] & reviewData
$sentiment < quantileReport[3,1]] <- '2 - Medium'
reviewData$sentimentDescription[reviewData$sentiment >= quantileReport[3,1] & reviewData
$sentiment < quantileReport[4,1]] <- '3 - High'
reviewData$sentimentDescription[reviewData$sentiment >= quantileReport[4,1]] <- '4 - Ver
y High'

# Clean up
rm(sentiments, quantileReport)
```

Summry of the results

The three variables derivied from the original data set lead us to believe that we can establish some level of usefulness based on available information. In summary:

- Higher sentiment in the text of the review will lead to an increased usefulness score
- Users who write multiple reviews tend to have a higher usefulness score than users who write one or two reviews

- Longer reviews tend to have a higher usefulness score than shorter reviews, and very short reviews aren't often considered useful

Sentiment Level

The interesting observation here is that sentiment level does have an impact on the usefulness score, but only when the sentiment is high or very high.

```
results <- sqldf('select sentimentDescription, isUseful, count(isUseful) as result
                 from reviewData
                 group by sentimentDescription, isUseful
                 order by sentimentDescription asc, isUseful asc')

knitr::kable(results)
```

sentimentDescription	isUseful	result
1 - Low	1 - Not Useful	92127
1 - Low	2 - Useful	87578
2 - Medium	1 - Not Useful	94403
2 - Medium	2 - Useful	85306
3 - High	1 - Not Useful	104208
3 - High	2 - Useful	75499
4 - Very High	1 - Not Useful	120385
4 - Very High	2 - Useful	59322

Review Volume from the user

Another interesting point can be made in looking at the number of reviews a user has completed. Users that have completed more than two reviews have a much higher proportion of usefulness than users who have completed a single or only two reviews.

```
results <- sqldf('select numberOfReviewsDescription, isUseful, count(isUseful) as result
                 from reviewData
                 group by numberOfReviewsDescription, isUseful
                 order by numberOfReviewsDescription asc, isUseful asc')

knitr::kable(results)
```

numberOfReviewsDescription	isUseful	result
1 - Single	1 - Not Useful	163028
1 - Single	2 - Useful	83277
2 - Two	1 - Not Useful	70263
2 - Two	2 - Useful	41929

numberOfReviewsDescription	isUseful	result
3 - More than two	1 - Not Useful	177832
3 - More than two	2 - Useful	182499

Review Length

The length of a review also contributed significantly to whether or not a review was considered useful. Short and medium length reviews were much more likely not to be useful. Long reviews were about at nearly the same level. However very long reviews seemed to score much higher than

```
results <- sqldf('select reviewLength, isUseful, count(isUseful) as result
                 from reviewData
                 group by reviewLength, isUseful
                 order by reviewLength asc, isUseful asc')

knitr::kable(results)
```

reviewLength	isUseful	result
1 - Short	1 - Not Useful	131540
1 - Short	2 - Useful	47338
2 - Medium	1 - Not Useful	115655
2 - Medium	2 - Useful	63261
3 - Long	1 - Not Useful	96441
3 - Long	2 - Useful	83727
4 - Very Long	1 - Not Useful	67487
4 - Very Long	2 - Useful	113379

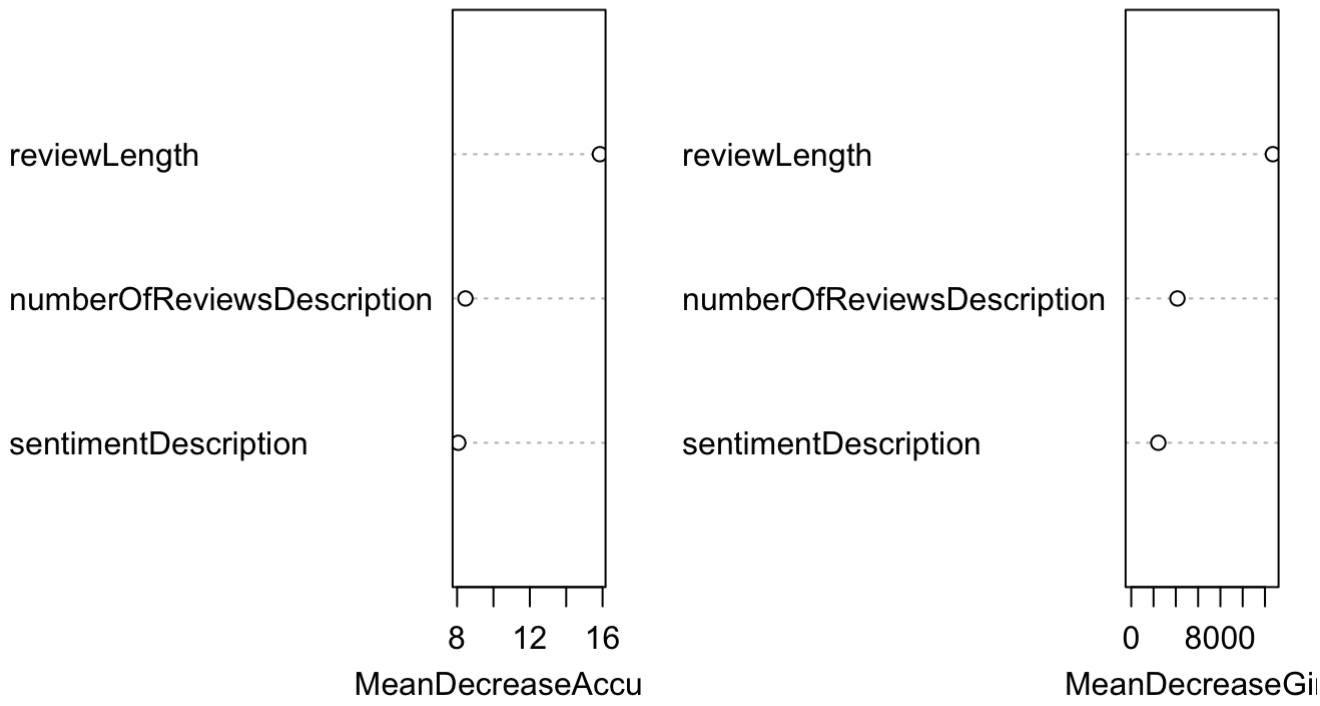
Generating a model and prediction

Based on the graphs above, we can see that there is some level of correlation between the variables we were able to infer and the usefulness score of the review. However, the question remains as to whether we can build a model that can accurately predict usefulness for users.

```
# Set useful as binary for the model generation
reviewData$usefulBinary[reviewData$isUseful == '1 - Not Useful'] <- FALSE
reviewData$usefulBinary[reviewData$isUseful == '2 - Useful'] <- TRUE

# Break the dataset into training and test groups
set.seed(410)
trainingPercent  <- 0.70
testPercent <- 0.30
reviewData$id <- 1:nrow(reviewData)
trainingData <- reviewData %>% sample_frac(.75)
testData  <- anti_join(reviewData, trainingData, by = 'id')

# Set classes for the training and test sets
trainingData$usefulBinary <- as.factor(trainingData$usefulBinary)
trainingData$numberOfReviewsDescription <- as.factor(trainingData$numberOfReviewsDescrip
tion)
trainingData$reviewLength <- as.factor(trainingData$reviewLength)
trainingData$sentimentDescription <- as.factor(trainingData$sentimentDescription)

testData$usefulBinary <- as.factor(testData$usefulBinary)
testData$numberOfReviewsDescription <- as.factor(testData$numberOfReviewsDescription)
testData$reviewLength <- as.factor(testData$reviewLength)
testData$sentimentDescription <- as.factor(testData$sentimentDescription)

# Fit a model using random forest
model <- randomForest(usefulBinary ~ numberOfReviewsDescription + reviewLength + sentime
ntDescription,
                      data = trainingData, ntree = 50, importance = TRUE)
```

By visually examining the model, we can see that the length of the review clearly has the most impact:

```
varImpPlot(model, sort = TRUE, main = 'Variable Impact', n.var = 3)
```



Making a prediction using our model

```
# Make our prediction
testData$prediction <- predict(model, testData, na.action = na.pass)

precision <- length(testData$user_id[testData$usefulBinary == TRUE & testData$prediction
 == TRUE]) /
  (length(testData$user_id[testData$usefulBinary == TRUE & testData$prediction == TRUE])
 +
     length(testData$user_id[testData$usefulBinary == FALSE & testData$prediction == TRU
E]))

recall <- length(testData$user_id[testData$usefulBinary == TRUE & testData$prediction ==
 TRUE]) /
  (length(testData$user_id[testData$usefulBinary == TRUE & testData$prediction == TRUE])
 +
     length(testData$user_id[testData$usefulBinary == TRUE & testData$prediction == FALS
E]))

print(paste('The precision of our model was: ', percent(precision)))
```

```
## [1] "The precision of our model was:  60.6%"
```

```
print(paste('The recall of our model was: ', percent(recall)))
```

```
## [1] "The recall of our model was:  47.7%"
```

The model produces a reasonable precision with a low recall. I believe this is acceptable given the application is recommending useful reviews. For example, if you search for a business on Yelp, you want the initial review results to be higher in accuracy. If a few useful reviews are missed, but the ones displayed are mostly helpful then it would likely be acceptable to the user.

Conclusion and additional thoughts

In the end, we were able to produce a model that can make some level of reasonable recommendation to users on which reviews they might find helpful. How might this model be improved? I think the answer likely lies in the ability to derive additional insights from the data. We added three variables to consider, but including others may have provided for a more accurate model. Some considerations include:

- The number of reviews a user produced may be artificially low, and the model may be more accurate than originally thought. Remember that we only sampled one million rows out of the more than seven million that were available. If given more time and processing power, we would have likely seen a higher number of users who had multiple review scores.
- Additional analysis of the review text. We looked at the sentiment and the length of the review, but I think it would be helpful to do additional text analysis. For example, we might have removed reviews that had non-english words (the study was done based on English speaking users only.) We could have reviewed the punctuation and grammar of the reviews to see if that had an impact on the usefulness score. During my research, I discovered the koRpus package (https://cran.r-project.org/web/packages/koRpus/vignettes/koRpus_vignette.pdf (https://cran.r-project.org/web/packages/koRpus/vignettes/koRpus_vignette.pdf)) which helps evaluate some of these perspectives.
- Additional data about the reviews. For the right reasons, Yelp chose to obfuscate details about the business and the user, and given the large size of the dataset, I chose to only load the review information. Examining additional details on the user such as the ammount of time they have been a user may have been interesting.