Group 1:

Allison Annick, Sara Cliffel, Kevin Hager, Sahil Nanavaty, Sana Khan, Victoria Shu

**1. Introduction**

**Dataset:** Goodreads Books Reviews

**Data overview:** This dataset contains more than 1.3M book reviews of approximately 25,475 books and 18,892 users where each book/user has at least one associated text-based review and numeric rating.

**Data Mining Problem**: User-generated reviews are a good way to judge the quality of any product, whether it's books, clothes, electronics, or any other purchasable good. Many of today’s transactions have transitioned from retail to ecommerce, so looking towards someone else’s prior experience with a product has become essential in our understanding of the product as a whole. Reader feedback, whether positive or negative, five stars or one star, will encourage the product owner to make improvements, allow the customer to make informed decisions regarding their time and money, and enable all parties involved to garner a better understanding of the product in question. Through the reviews catalogued in this dataset, we will demonstrate the importance of text-based reviews and explore the nuances of using a one to five rating system when ranking books.

**Attribute Information:**

* **user\_id** - Id of user
* **book\_id** - Id of Book
* **review\_id** - Id of review
* **rating** - rating from 0 to 5
* **review\_text** - review text
* **date**\_**added** - date added
* **date\_updated -** date updated
* **read\_at** - read at
* **started\_at -** started at
* **n\_votes** - no. of votes
* **n\_comments** - no. of comment

**Guiding Questions:**

* Are ratings, votes, and comment numbers an effective way to communicate the quality of a book?
* Can text-based reviews offer a medium in which to more effectively communicate the quality of a book?

**Table of Contents:**

1. Introduction
2. Overall Preprocessing
3. Association Rule Mining
4. K-Means Clustering
5. Decision Tree
6. Sentiment Analysis
7. Failed Plots
8. Data Critique
9. Conclusion
10. Appendices
    1. Appendix A – Association Rule Mining Code
    2. Appendix B – K-Means Clustering Code
    3. Appendix C – Decision Tree Code
    4. Appendix D – Sentiment Analysis Code
    5. Appendix E – Failed Plots Code

**2. Overall Preprocessing**

The first step in preprocessing this data was to get a summary of the data to understand the datatypes and column names associated with the data. After understanding a general outline of the data being used, the group confirmed that the data did not have any NAs or Null values that could inhibit correct processing of the data.

When assessing the data, the following columns were assessed to be relevant and able to be binned: user\_id, book\_id, rating, n\_votes, n\_comments. The time data included in this dataset (date\_added, date\_updated, read\_at, started\_at) was assessed to be unhelpful in answering the guiding questions generated for this project. This is because the time at which the book was read and the date at which the review was written are irrelevant to determining the quality of the book.

Additional preprocessing, specific to each process or algorithm used, is outlined in the section to which it pertains.

**3. Association Rule Mining**

The goal of using Association Rule Mining to assess this dataset was to answer two of the Guiding Questions generated for this project:

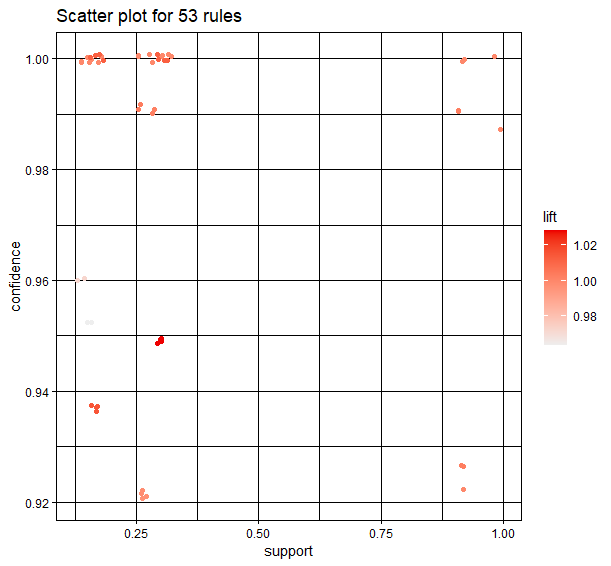
1. Are ratings, votes, and comment numbers an effective way to communicate the quality of a book?
2. Is the user who is authoring the review and rating important to the understanding of the overall book review?

The preprocessing for this rule mining began with determining what columns could be rule mined. As was explained above, the time data was removed from the dataset for this process. Additionally, the review\_text column was removed for this process for two reasons. First, including the review\_text was unnecessary to answer the questions stated above, and, secondly, the text in the review could not be easily discretized into digestible data for the association rule mining process.

The list of 18 rules below was generated using the entire dataset with a minimum confidence value of 0.9 and a minimum support value of 0.06:

|  |
| --- |
| lhs rhs  [1] {rating=2} => {n\_comments=between 0 and 3}  [2] {rating=3} => {n\_comments=between 0 and 3}  [3] {rating=5} => {n\_comments=between 0 and 3}  [4] {rating=4} => {n\_comments=between 0 and 3}  [5] {rating=3} => {n\_votes=between 0 and 10}  [6] {rating=5} => {n\_votes=between 0 and 10}  [7] {rating=2} => {n\_votes=between 0 and 10}  [8] {rating=4} => {n\_votes=between 0 and 10}  [9] {rating=2, n\_comments=between 0 and 3} => {n\_votes=between 0 and 10}  [10] {rating=4, n\_comments=between 0 and 3} => {n\_votes=between 0 and 10}  [11] {rating=5, n\_comments=between 0 and 3} => {n\_votes=between 0 and 10}  [12] {rating=3, n\_comments=between 0 and 3} => {n\_votes=between 0 and 10}  [13] {rating=2, n\_votes=between 0 and 10} => {n\_comments=between 0 and 3}  [14] {rating=3, n\_votes=between 0 and 10} => {n\_comments=between 0 and 3}  [15] {rating=5, n\_votes=between 0 and 10} => {n\_comments=between 0 and 3}  [16] {rating=4, n\_votes=between 0 and 10} => {n\_comments=between 0 and 3}  [17] {n\_comments=between 0 and 3} => {n\_votes=between 0 and 10}  [18] {n\_votes=between 0 and 10} => {n\_comments=between 0 and 3} |

*Code for association rule mining rules in Appendix A - Rules.*



*Code for association rule mining plot in Appendix A – Plot.*

The graph created from the rules is a demonstration of the similarity between the 18 rules. The viewer can see all the data points are grouped together which is indicative of the minimal variation in the rules.

The similarity in the set of 18 rules generated above is reflective of the small range in the ratings, comments, and votes sections of the data. The votes ranged from 0 to 3222 votes. However, after binning the vote numbers, 845448 reviews fell into the “0-10 votes” bin. The comments ranged from 0 to 1335 comments. Similarly, after binning the comment numbers, 840609 reviews fell into the “0-3 comments” bin. Because there is so little variation in the majority of the data, the 1-5 rating system is responsible for differentiating between the book reviews. However, the 1-5 rating system offers only 5 separate bins in which over 900,000 book reviews must be classified. This does not offer enough variation to truly distinguish between quality books and inferior books.

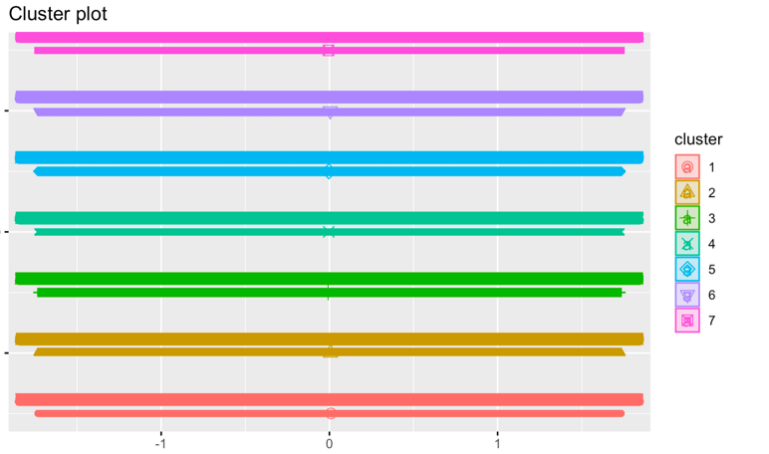
Revisiting the guiding questions referenced above, the rules generated for this project would argue that ratings, votes, and comment numbers are not an effective way to communicate the quality of a book due to the inadequate range of the distinguishing variables. The lack of inclusion of any user IDs or book IDs in the generated rules indicates that there is not enough consistency in the rating of any user or book to warrant a rule. As such, the conclusion can be drawn that the user authoring the rule is not essential to understanding the review.

**4. K-Means Clustering**

The goal of using K-Means clustering to assess this dataset was to answer two of the Guiding Questions generated for this project:

1. Are ratings, votes, and comment numbers an effective way to communicate the quality of a book?

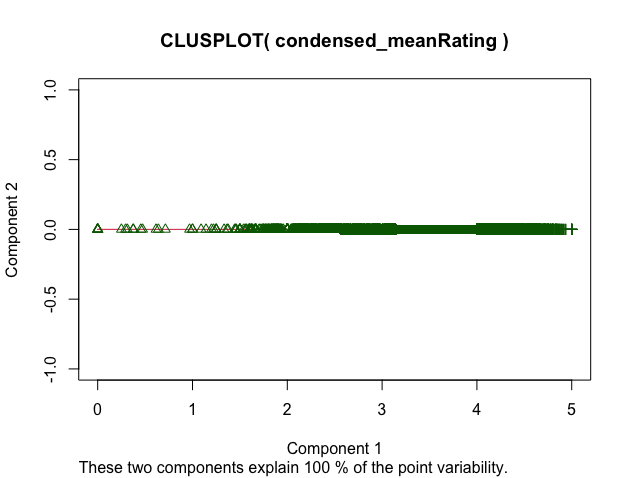
The preprocessing for K-means clustering began with determining what columns were relevant to answering these guiding questions. As was explained above, the time data was removed from the dataset for this process. The group did an initial k-means cluster on all values in the data set excluding the date/time columns. Using all the data provided inclusive results as seen in the image below. This shows that we need to narrow down what variables are used to get any valuable information out of using k-means for clustering ratings.



*Code for this plot in Appendix B – Plot 1*

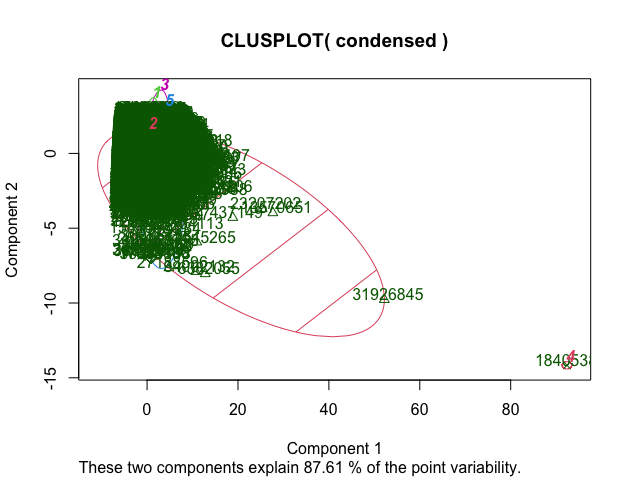
In preprocessing, it was observed that certain books were rated several times in the dataset. In order to generate clusters indexed by unique book IDs, the group averaged the ratings of all the books that had multiple reviews. The multiple reviews were condensed into a single data point with the rating of the now unique book\_id being the generated average. After condensing the book\_ids, the row value was changed to the book\_id so that the plot indexed the cluster based on the book\_id.

The first cluster plot generated used only the ratings. The almost solid line generated from the book\_ids is a demonstration that with such a large dataset, it is ineffective to differentiate book reviews by a 1 to 5 rating. The scale is simply too small for a specific conclusion to be drawn from such a large dataset.



*Code for this plot in Appendix B – Plot 2.*

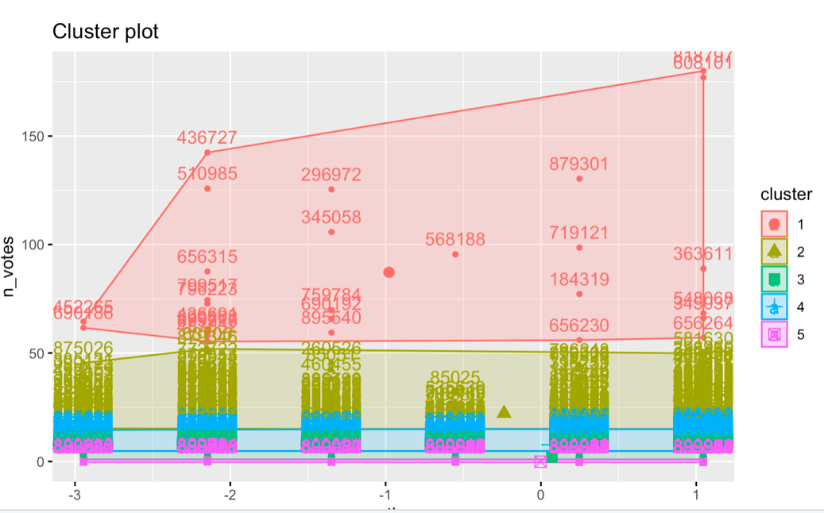
The second cluster plot used similar preprocessing to the first plot. The books with more than one review were condensed into a single review with their rating, comment number, and vote number averaged from all the reviews for that specific book\_id. The cluster plot once again confirms our findings that the numeric data gathered in this dataset, the rating, comment number, and vote number, are insufficient in differentiating reviews in a dataset of this size.



*Code for this plot in Appendix B – Plot 3.*

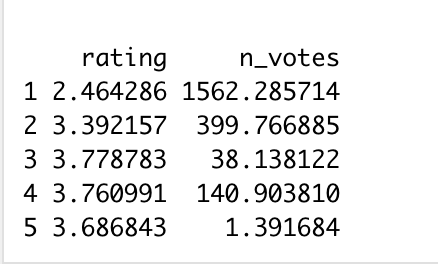
Because 845,448 reviews had less than 10 votes and 840,609 reviews had less than 3 comments, the 1-5 rating system is really the only measure from which to differentiate the data. However, as demonstrated by all the clusters overlapping, aside from a few outliers, the small rating scale of 1 to 5 cannot make up for the similarity of the voting and comment numbers in order to give the customer the ability to distinguish between book quality.

After plotting using comments, votes, and ratings, the group decided to attempt the cluster plots using only two of the attributes to see if more focused data could offer more interpretable plots. The group narrowed down the data to include only ratings and number of votes which produced this plot:

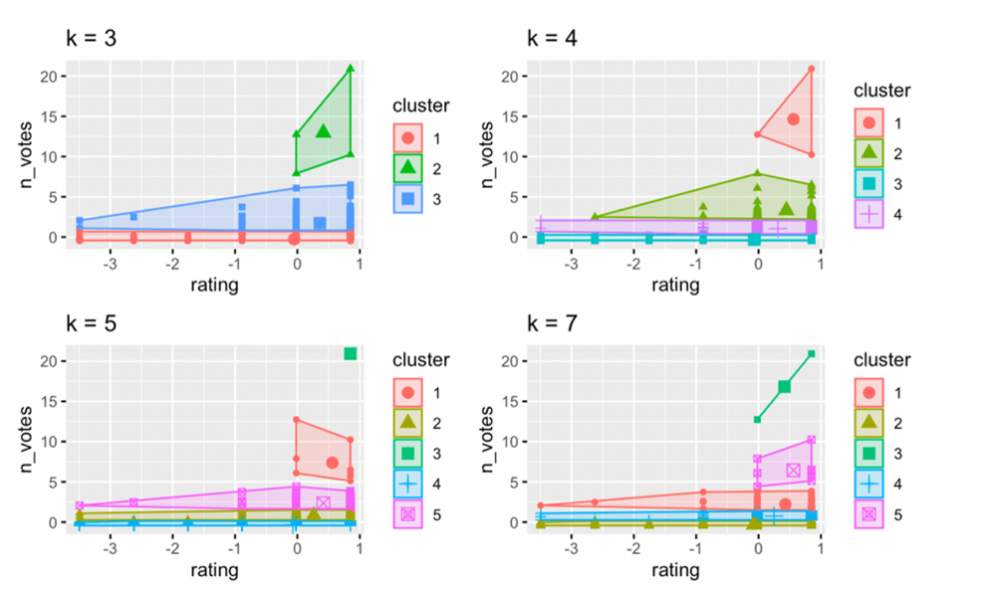


*Code for this plot in Appendix B – Plot 4.*

The graph above shows a few outliers, but it is rather difficult to make any additional interpretations. This is because there are so many rows in the data set, over 900,000 and our data is heavily skewed towards higher ratings. Plotting the cluster means shows the following data for ratings and number of votes:

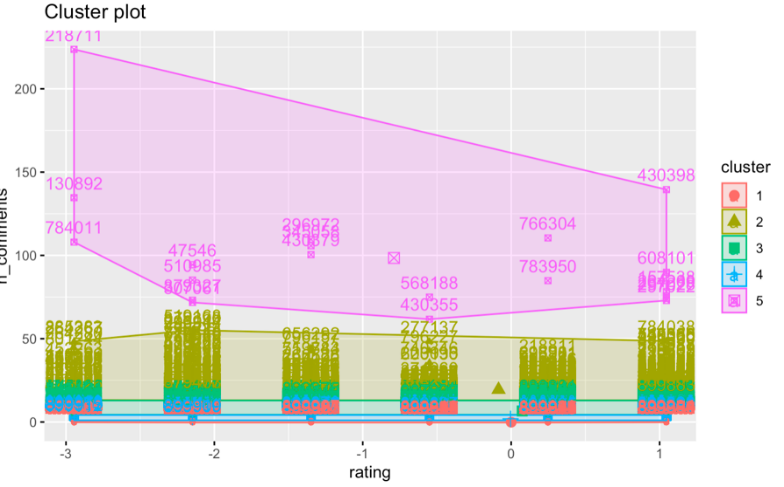


Plotting with multiple clusters shows that there are clear outliers in our data.



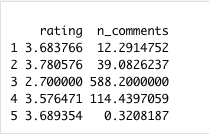
*Code for this plot in Appendix B – Plot 5.*

Plotting with ratings and the number of comments also demonstrated outliers, but again, due to the huge amount of data and very small scale, it was difficult to make any meaningful interpretations out of the generated plot:

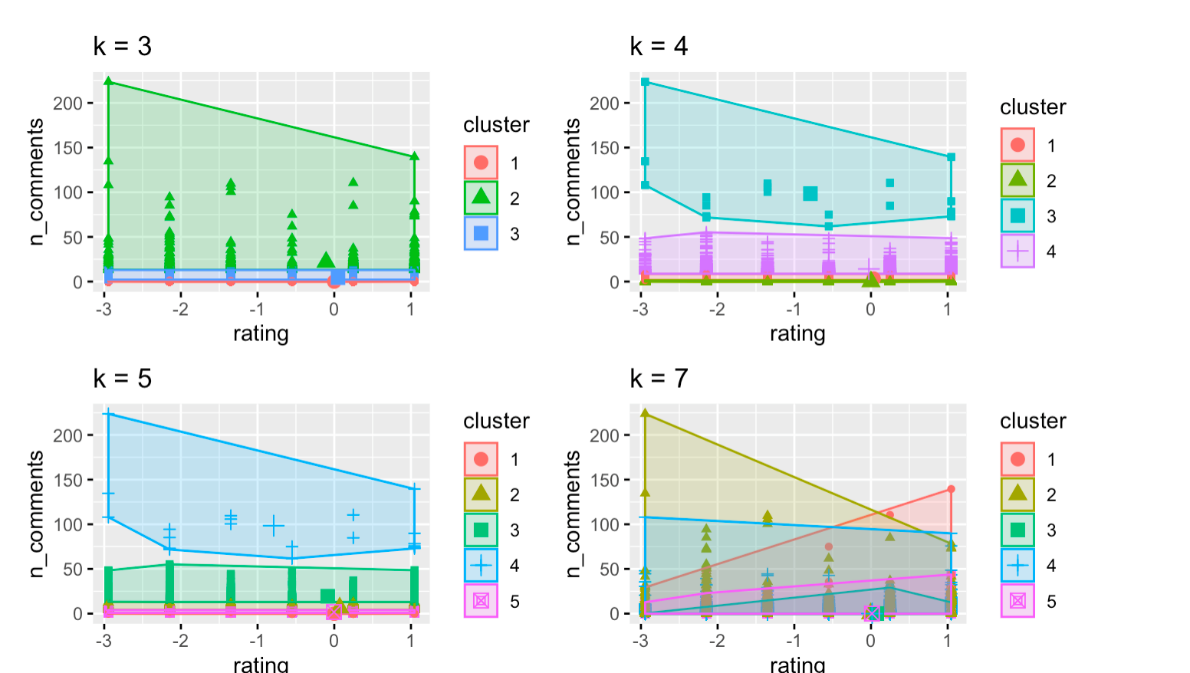


*Code for this plot in Appendix B – Plot 6.*

Plotting the cluster means shows the following data for ratings and the number of comments:



We decided to plot the k-means model for number of comments and ratings with five clusters. This data is similar to the graph above and is difficult to interpret due to the number of rows in the data set, but it shows there are some outliers compared to most of the data.

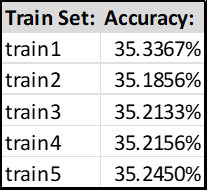


*Code for this plot in Appendix B – Plot 7.*

**5. Decision Tree**

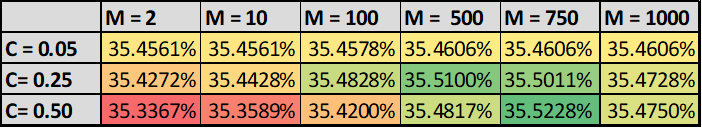
In an effort to help answer our guiding research questions, we attempted to create multiple predictive models to predict the rating of a book. In doing so, we chose to implement the decision tree model using the J48 option in the RWeka package. Our main predictors were the number of votes a book had and the number of comments on a book.

In the process of tuning our model, we experimented with several different parameters. To begin, we simply created a decision tree with five different, smaller, subsets of our original training data as we had size constraints. For each subset, we had randomly selected a total of 180,000 rows, or 20% of our larger train set. To determine which subset was the best, we initially ran each through their own decision tree model and compared their accuracies. Of our subsets, the first, “train1”, yielded the highest accuracy. The results of all five train sets can be seen in the table below.

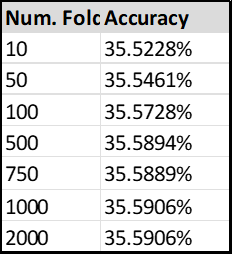


In addition to trying different subsets of the training data in order to improve our model accuracy, we also looked at the number of folds in our model, the level of confidence, the minimum number of instances, and adding other attributes into the original equation. Unfortunately, adding more attributes made the model worse so further exploration on that front was halted.

Before we began adjusting our number of cross-validation folds, we began by changing our minimum number of instances (M) and our confidence level (C). We started with a small c-value, setting it to only 0.05, and, over the course of adjusting, we ended up settling at 0.50. We went about adjusting the M-value in the same way, starting at the default 2, and increasing it up to 1000. The change in accuracy can be seen in the table below.

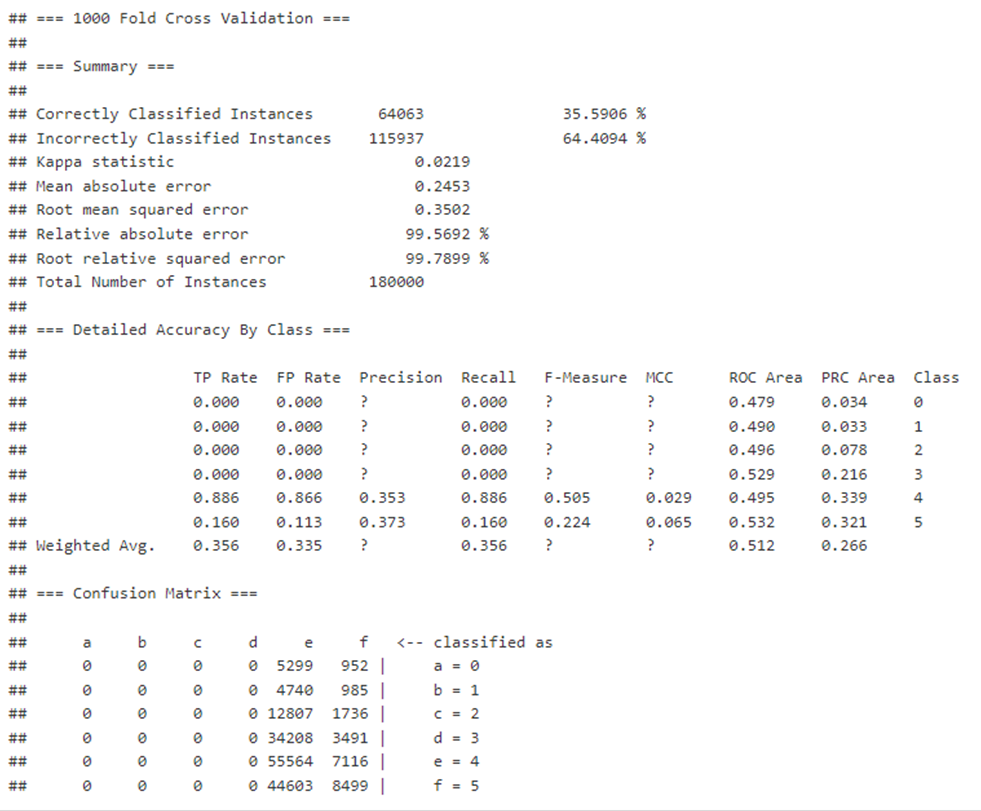


As seen in the table above, our best accuracy came from the parameters C=0.50 and M=750. This became the foundation for us to continue tuning by adjusting cross-fold validation. In changing the number of folds in our tree, we ranged in testing from 10 to 2000 folds.



To reiterate, our final model’s parameters were tuned so that our confidence level would equal 0.05, our minimum number of iterations was set to 750, and we chose to use 1000 folds for the cross-validation settings. The output of our implemented model was recorded, and as our dataset has a competition, we submitted it to Kaggle, which assigned us a score of 0.3547, taking the 22nd place on the leaderboard.

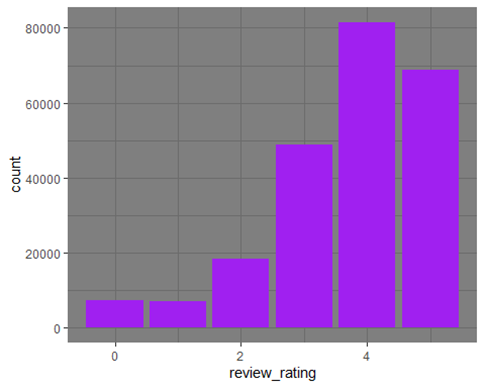
While this isn’t a bad standing necessarily, it was not as high of results we were hoping for. We can attribute some of this to factors like the uneven spread of book ratings. Of our total dataset, only 14% of ratings fell into the 0, 1, and 2 categories. This is bound to skew our predictive models, and if you look at the output confidence matrix in the results below, you will see clearly that these ratings were unlikely to have been accurately predicted.

*Code for this model is in Appendix C.*

**6. Sentiment Analysis**

When analyzing data that deals with reviews it is very beneficial to use sentiment analysis on the reviews to be able to draw conclusions. The benefit of sentiment analysis is that it helps reveal the emotion behind the reviews. By combining what we have learned in this class, along with learnings from IST 687, plus some internet resources <https://uc-r.github.io/sentiment_analysis> & <https://jtr13.github.io/cc21/sentiment-analysis-and-wordcloud.html>, we were able to do a detailed dive into the book reviews and glean very beneficial information. For this sentiment analysis we used the following packages: dplyr, tidyr, stringr, tidytext, ggplot2, wordcloud, and textdata.

First, we were able to determine that the average rating in the data was a 3.71/5 and the median rating was 4/5.



*Code for this plot in Appendix D – Plot 1 A.*

After removing rating of “0”:

Chart, histogram

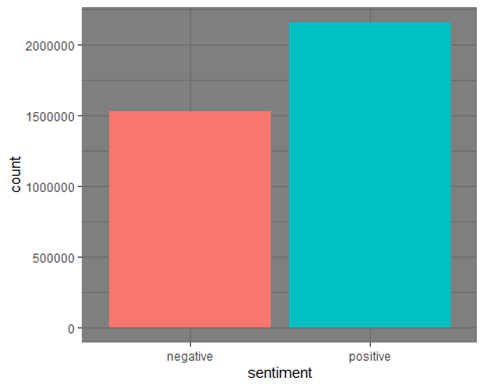
Description automatically generated

*Code for this plot in Appendix D – Plot 1 B.*

By using the regular expressions, we also determined that the mean number of characters in a review was 1029, mean number of words in a review was 187, and the mean number of sentences in a review was 12.57.

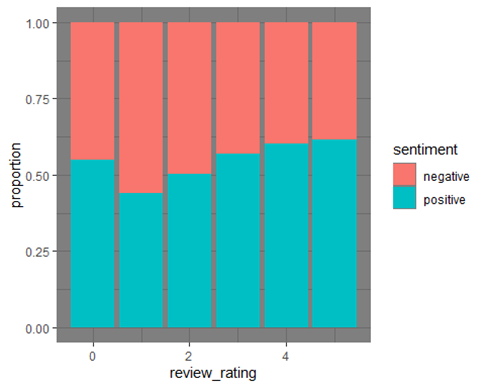
Now, getting into the sentiment analysis, there are a few different lexicons that are available in R to use for this analysis. We decided to use both the Bing lexicon and the NRC lexicon. The Bing lexicon classifies words as being either positive or negative, while the NRC lexicon categorizes the words by emotion by putting them into categories like positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

When we used the Bing sentiment analysis, we found that there were over 500,000 more positive words used in the reviews than negative.



*Code for this plot in Appendix D – Plot 2.*

When looking at each review, it is clear that as the rating increases from 1 to 5, the positive sentiment is higher. The “0” ratings were removed next because there seemed to be confusion among the users writing the reviews since books with a “0” were basically split 50/50 for sentiment analysis.



*Code for this plot in Appendix D – Plot 3 A.*

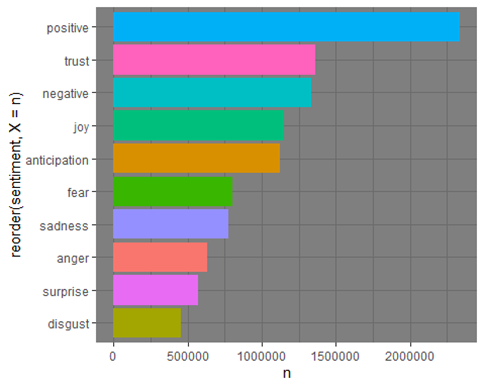
After removing ratings of 0:

Chart, bar chart, histogram

Description automatically generated

*Code for this plot in Appendix D – Plot 3 B.*

When we use the NRC lexicon, which splits the words into categories based on an associated emotion, we get the following results. Positive, trust, negative, joy, and anticipation are the top 5 categories with positive being by far the largest, doubling the next closest category.



*Code for this plot in Appendix D – Plot 4.*

When we do a simple word cloud looking at the top 300 words, we are given the following results:

Text

Description automatically generated

*Code for this plot in Appendix D – Plot 5.*

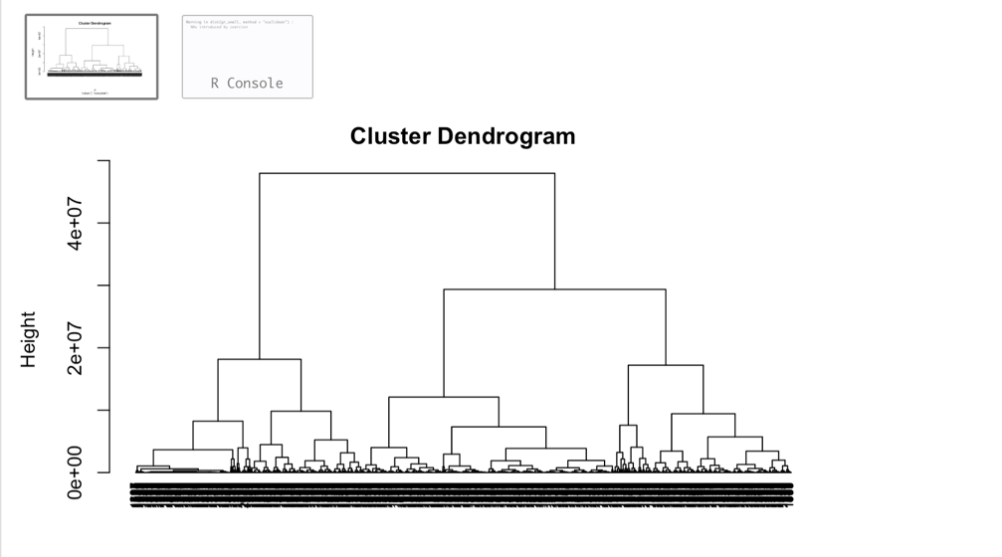
However, if we want a more detailed word cloud, we can use the Bing lexicon again and display both the positive and the negative words in the same output. These results are pictured below:

Text

Description automatically generated

*Code for this plot in Appendix D – Plot 6.*

**7. Failed Plots**



*Code for this plot in Appendix E – Plot 1.*

The group attempted to make a cluster Dendrogram, which ran successfully, but is difficult to make any useful interpretations from the resulting graph. This is because the amount of data used was too large. Even when limiting the data, the Dendrogram was difficult to interpret because of the similarity of most of the data.

**8. Data Critique**

With additional attributes, a more comprehensive analysis using the same data mining techniques demonstrated in the report could have been performed. Having access to attributes such as book titles, genres, and authors would have given us more options to consider with regards to classification. A greater range of attributes would have allowed different ways of distinguishing between books since the group demonstrated above that the comments, votes, and rating range was not adequate for this task.

This data could be paired with other review data in order to detect similarities and differences between the two datasets. For example, having access to professional reviews of each book would have allowed us to detect and compare differences with the user-generated reviews.

**9. Conclusion**

The following conclusion will demonstrate findings from the Goodreads Book Review data using the guiding questions generated at the beginning of the project:

1. Are ratings, votes, and comment numbers an effective way to communicate the quality of a book?

No, ratings, votes, and comment numbers are not an effective way to communicate the quality of a book. The dataset was extremely large (over 900,000 rows) with comparatively few attributes. The best attributes from which to generate association rules and clusters is the number of votes, number of comments, and rating scale. However, because over 93% of the data had fewer than 3 comments and fewer than 10 votes, the responsibility for differentiating the data fell largely on the 1 to 5 rating scale. The 1 to 5 rating scale, though, is too small a scale from which to distinguish book quality when using a dataset of over 900,000 rows.

1. Can text-based reviews offer a medium in which to more effectively communicate the quality of a book?

Yes, text-based reviews offer a medium in which to more effectively communicate the quality of a book. After conducting sentiment analysis on the text-based reviews in the dataset, there were clear patterns regarding positive and negative word association that the group was able to demonstrate using a number of different techniques.

Finally, the group assessed that with a greater number of attributes, more effective clustering could have been conducted on the dataset. Ratings, votes, and comments may be an effective way to communicate the quality of a book in a much smaller dataset. However, the larger the dataset, the less effective these measures can be in communicating the quality of a book. Text-based reviews are an effective way to communicate the quality of a book regardless of the size of the dataset, and sentiment analysis can offer a method to identify patterns in the text.

**Appendix A – Association Rule Mining Code**

**Rules:**

#load necessary libraries

library(RWeka)

library(arules)

library(arulesViz)

library(datasets)

library(readr)

library(tidyr)

#import data

originalData <- read.csv("/Users/aannick/Documents/Grad School/IST707/Project/goodreads\_train.csv")

#PREPROCESSING FOR ASSOCIATION RULE MINING

#Remove date\_added, date\_updated, read\_at, started\_at, review\_id, review\_text

preprocessedData <- originalData[,c("user\_id","book\_id","rating","n\_votes","n\_comments")]

#Check if there are entries with NAs

colSums(is.na(preprocessedData))

#no NAs present in data.

#Check data types

str(preprocessedData)

#Count unique values to determine the population present in the data

table(preprocessedData$user\_id)

table(preprocessedData$book\_id)

#The data contains repeat users reviewing the data, repeat books being reviewed, but every review is unique.

#Discretize rating. Rating is already in 6 bins: 0,1,2,3,4,5. Do we need to turn these into factors for the purpose of association rule mining.

#preprocessedData$rating <- as.factor(preprocessedData$rating)

#str(preprocessedData)

#Discretize voting number

#find minimum and maximum number of votes in order to determine necessary bins

min(originalData$n\_votes)

max(originalData$n\_votes)

table(preprocessedData$n\_votes)

#change all values less than or equal to 0 to 1 so they go in the 0-50 bin

preprocessedData$n\_votes[preprocessedData$n\_votes<=0] <- 1

#initially tried: preprocessedData$n\_votes <- cut(preprocessedData$n\_votes, breaks = c(0,50,100,150,200,300,400,500,1000,2000,Inf),labels=c("between 0 and 50","between 50 and 100","between 100 and 150","between 150 and 200","between 200 and 300","between 300 and 400", "between 400 and 500","between 500 and 1000", "between 1000 and 2000","more than 2000"))

#the above attempt did not differentiate the votes enough, as 0-50 still had 890,000 of the 900,000 total rows.

preprocessedData$n\_votes <- cut(preprocessedData$n\_votes, breaks = c(0,10,20,30,40,50,60,70,80,300,Inf),labels=c("between 0 and 10","between 10 and 20","between 20 and 30","between 30 and 40","between 40 and 50","between 50 and 60","between 60 and 70", "between 70 and 80","between 80 and 300","more than 300"))

table(preprocessedData$n\_votes)

#Discretize comment number

#find minimum and maximum number of votes in order to determine necessary bins

min(originalData$n\_comments)

max(originalData$n\_comments)

table(preprocessedData$n\_comments)

#change all values less than or equal to 0 to 1 so they go in the 0-20 bin

preprocessedData$n\_comments[preprocessedData$n\_comments<=0] <- 1

#initially tried: preprocessedData$n\_comments <- cut(preprocessedData$n\_comments, breaks = c(0,20,30,40,50,100,200,300,400,Inf),labels=c("between 0 and 20","between 20 and 30","between 30 and 40","between 40 and 50","between 50 and 100","between 100 and 200", "between 200 and 300","between 300 and 400","more than 400"))

#the above attempt did not differentiate the votes enough

preprocessedData$n\_comments <- cut(preprocessedData$n\_comments, breaks = c(0,3,6,9,12,15,18,21,50,Inf),labels=c("between 0 and 3","between 3 and 6","between 6 and 9","between 9 and 12","between 12 and 15","between 15 and 18", "between 18 and 21","between 21 and 50","more than 50"))

#values less than 0 and equal to 0 were changed to NA

colSums(is.na(preprocessedData))

#changing columns 1,2,3 to factors

preprocessedData$user\_id <- as.factor(preprocessedData$user\_id)

preprocessedData$book\_id <- as.factor(preprocessedData$book\_id)

preprocessedData$rating <- as.factor(preprocessedData$rating)

#table to figure the high frequency user\_id/book\_id

#https://www.geeksforgeeks.org/find-the-index-of-the-maximum-value-in-r-dataframe/

which.max(table(preprocessedData$user\_id))

#highest frequency user\_id: aca760854b57ce2ec981df32e46dc96c

which.max(table(preprocessedData$book\_id))

#highest frequency book\_id: 11870085

################

#can create a smaller data set with unique user\_ids or book\_ids to generate rulesets specific to a user or book

smallerUserData <- preprocessedData[preprocessedData$user\_id == "aca760854b57ce2ec981df32e46dc96c",]

smallerBookData <- preprocessedData[preprocessedData$book\_id == "11870085"]

###############

#INITIAL RULE DISCOVERY

#credit for below code: SU Wk. 3 Course Guidance: "Convert Record Data to Transactions for Association Rule Mining in R

#adjusted support and confidence value until arriving at a list of 30 rules with high support and confidence.

#can replace value: "smallerUserData" with different datasets

myRules <- apriori(preprocessedData, parameter = list(minlen = 2,conf = 0.9,supp = 0.06))

#credit: https://stackoverflow.com/questions/32106764/sorting-rules-by-lift-and-confidence

#sort rules by lift, and then use the top 20 for the strong rule list

myRules <- sort(myRules, decreasing=TRUE,by="lift")

inspect(myRules)

#setting book\_id as RHS value

#can replace value: "smallerUserData" with different datasets

LHSRules <- apriori(preprocessedData, parameter=list(minlen = 2), appearance = list(lhs=c("rating=5")))

inspect(LHSRules)

**Plot:**

plot(myRules)

**Appendix B – K-Means Clustering Code:**

**Plot 1:**

K -means plot:

model\_r = kmeans(small\_trainset, centers = 7, nstart = 25)

model\_r$centers

cluster\_assignment <- data.frame(small\_gr\_train\_copy,model\_r$cluster)

View(cluster\_assignment)

fviz\_cluster(model\_r, data = cluster\_assignment)

**Plot 2:**

#load necessary libraries

library(RWeka)

library(cluster)

library(arules)

library(arulesViz)

library(datasets)

library(readr)

library(tidyr)

#import data

originalData <- read.csv("/Users/aannick/Documents/Grad School/IST707/Project/goodreads\_train.csv")

#PREPROCESSING FOR CLUSTERING

#Check if there are entries with NAs

colSums(is.na(preprocessedData))

#no NAs present in data.

#Check data types

str(preprocessedData)

########################CLUSTER WITH ONLY RATING################################

clusterRating <- originalData[,c("book\_id","rating")]

condensed\_meanRating <- aggregate(clusterRating$rating,by=list(clusterRating$book\_id),data=clusterRating,FUN=mean)

#change the book\_id to the row values

rownames(condensed\_meanRating) <- condensed\_meanRating$Group.1

#get rid of the book\_id column cause it's now the row label

condensed\_meanRating$Group.1<-NULL

colnames(condensed\_meanRating) <- c("rating")

#use the kMeans algorithm in R

model\_r\_bookDataRating <- kmeans(condensed\_meanRating,5)

#plot kMeans algorithm

clusplot(condensed\_meanRating,model\_r\_bookDataRating$cluster,color=TRUE,shade=TRUE,labels=2,lines=0)

########################END CLUSTER WITH ONLY RATING################################

**Plot 3:**

#load necessary libraries

library(RWeka)

library(cluster)

library(arules)

library(arulesViz)

library(datasets)

library(readr)

library(tidyr)

#import data

originalData <- read.csv("/Users/aannick/Documents/Grad School/IST707/Project/goodreads\_train.csv")

#PREPROCESSING FOR CLUSTERING

#Check if there are entries with NAs

colSums(is.na(preprocessedData))

#no NAs present in data.

#Check data types

str(preprocessedData)

########################START CLUSTER WITH RATING, COMMENTS, VOTES##################

#Remove date\_added, date\_updated, read\_at, started\_at, review\_id, review\_text

clusterRatingVotesComments <- originalData[,c("book\_id","rating","n\_votes","n\_comments")]

#keep only unique book\_id values, but average the rating of all the instances of that particular book\_id from the big dataset

#source: https://r-coder.com/aggregate-r/

condensed\_meanRating <- aggregate(clusterRatingVotesComments$rating,by=list(clusterRatingVotesComments$book\_id),data=clusterRatingVotesComments,FUN=mean)

condensed\_meanVotes <- aggregate(clusterRatingVotesComments$n\_votes,by=list(clusterRatingVotesComments$book\_id),data=clusterRatingVotesComments,FUN=mean)

condensed\_meanComments <- aggregate(clusterRatingVotesComments$n\_comments,by=list(clusterRatingVotesComments$book\_id),data=clusterRatingVotesComments,FUN=mean)

condensed <- merge(condensed\_meanRating,condensed\_meanVotes,by="Group.1")

colnames(condensed) <- c("Group.1","rating","votes")

condensed <- merge(condensed,condensed\_meanComments,by="Group.1")

colnames(condensed) <- c("Group.1","rating","votes","comments")

#change the book\_id to the row values

rownames(condensed) <- condensed$Group.1

#get rid of the book\_id column cause it's now the row label

condensed$Group.1<-NULL

#testing condensed\_mean code

#newVector <- preprocessedData[preprocessedData$book\_id == 2, ]

#mean(newVector$rating)

#use the kMeans algorithm in R

model\_r\_bookDataAll <- kmeans(condensed,5)

#plot kMeans algorithm

clusplot(model\_r\_bookDataAll$cluster,color=TRUE,shade=TRUE,labels=2,lines=0)

########################END CLUSTER WITH RATING, COMMENTS, VOTES##################

**Plot 4:**

gr\_train <- read.csv("/Users/gozi/Downloads/goodreads-books-reviews-290312/goodreads\_train.csv")

trial = gr\_train[,c('rating', 'n\_votes')]

model\_r = kmeans(trial, centers = 5, nstart = 25)

model\_r$centers

cluster\_assignment <- data.frame(trial,model\_r$cluster)

#View(cluster\_assignment)

fviz\_cluster(model\_r, data = trial)

**Plot 5:**

k3 <- kmeans(trial, centers = 3, nstart = 25)

k4 <- kmeans(trial, centers = 4, nstart = 25)

k5 <- kmeans(trial, centers = 5, nstart = 25)

# plots to compare

p1 <- fviz\_cluster(k3, geom = "point", data = trial) + ggtitle("k = 3")

p2 <- fviz\_cluster(k4, geom = "point", data = trial) + ggtitle("k = 4")

p3 <- fviz\_cluster(k5, geom = "point", data = trial) + ggtitle("k = 5")

p4 <- fviz\_cluster(model\_r, geom = "point", data = trial) + ggtitle("k = 7")

require(gridExtra)

grid.arrange(p1, p2, p3, p4, nrow = 2)

**Plot 6:**

trial2 = gr\_train[,c('rating', 'n\_comments')]

trial2

model\_r2 = kmeans(trial2, centers = 5, nstart = 25)

model\_r2

model\_r2$centers

fviz\_cluster(model\_r2, data = trial2)

**Plot 7:**

k31 <- kmeans(trial2, centers = 3, nstart = 25)

k41 <- kmeans(trial2, centers = 4, nstart = 25)

k51 <- kmeans(trial2, centers = 5, nstart = 25)

# plots to compare

p11 <- fviz\_cluster(k31, geom = "point", data = trial2) + ggtitle("k = 3")

p21 <- fviz\_cluster(k41, geom = "point", data = trial2) + ggtitle("k = 4")

p31 <- fviz\_cluster(k51, geom = "point", data = trial2) + ggtitle("k = 5")

p41 <- fviz\_cluster(model\_r, geom = "point", data = trial2) + ggtitle("k = 7")

require(gridExtra)

grid.arrange(p11, p21, p31, p41, nrow = 2)

**Appendix C – Decision Tree Code:**

#Libraries:

suppressWarnings(suppressMessages(library(tidyverse)))

suppressWarnings(suppressMessages(library(ggplot2)))

suppressWarnings(suppressMessages(library(rpart)))

suppressWarnings(suppressMessages(library(plyr)))

suppressWarnings(suppressMessages(library(dplyr)))

suppressWarnings(suppressMessages(library(rJava)))

suppressWarnings(suppressMessages(library(RWeka)))

suppressWarnings(suppressMessages(library(cluster)))

suppressWarnings(suppressMessages(library(caret)))

suppressWarnings(suppressMessages(library(e1071)))

suppressWarnings(suppressMessages(library(class)))

suppressWarnings(suppressMessages(library(arules)))

suppressWarnings(suppressMessages(library(quanteda)))

suppressWarnings(suppressMessages(library(quanteda.textplots)))

suppressWarnings(suppressMessages(library(quanteda.textstats)))

suppressWarnings(suppressMessages(library(readr)))

suppressWarnings(suppressMessages(library(arules)))

suppressWarnings(suppressMessages(library(arulesViz)))

suppressWarnings(suppressMessages(library(randomForest)))

#Start by importing the test and train data sets.

test <- read.csv("C:\\Users\\sarac\\OneDrive\\Documents\\Syracuse\\IST 707 Data Analytics--Applied Machine Learning\\Group Project\\goodreads\_test.csv")

train <- read.csv("C:\\Users\\sarac\\OneDrive\\Documents\\Syracuse\\IST 707 Data Analytics--Applied Machine Learning\\Group Project\\goodreads\_train.csv")

## \*\*Pre-processing\*\*

#Get rid of unnecessary attributes.

dropAttributes <- c("started\_at", "read\_at", "date\_updated", "date\_added")

test <- select(test, -dropAttributes) #dpylr package select() to delete columns by name

train <- select(train, -dropAttributes)

train$rating <- as.factor(train$rating)

#Because of the massive size of both of these sets they take a long time to process. We will be #cutting them each down and pasting their outputs together for the final results.

set.seed(123)

train1 <- sample(nrow(train),nrow(train)\*.20)

train1 <- train[train1,]

train2 <- sample(nrow(train),nrow(train)\*.20)

train2 <- train[train2,]

train3 <- sample(nrow(train),nrow(train)\*.20)

train3 <- train[train3,]

train4 <- sample(nrow(train),nrow(train)\*.20)

train4 <- train[train4,]

train5 <- sample(nrow(train),nrow(train)\*.20)

train5 <- train[train5,]

# \*\*Decision Tree Models:\*\*

#Run decision tree models on each subset of data.

#Model 1:

DTmodel1 <- J48(rating~ n\_votes + n\_comments , data = train1, control=Weka\_control(U=FALSE, M=2, C=0.5))

#Model 2:

DTmodel2 <- J48(rating~ n\_votes + n\_comments , data = train2, control=Weka\_control(U=FALSE, M=2, C=0.5))

#Model 3:

DTmodel3 <- J48(rating~ n\_votes + n\_comments , data = train3, control=Weka\_control(U=FALSE, M=2, C=0.5))

#Model 4:

DTmodel4 <- J48(rating~ n\_votes + n\_comments , data = train4, control=Weka\_control(U=FALSE, M=2, C=0.5))

#Model 5:

DTmodel5 <- J48(rating~ n\_votes + n\_comments , data = train5, control=Weka\_control(U=FALSE, M=2, C=0.5))

#Might have to run the following code if you have a r Java memory error:

#`options(java.parameters = "- Xmx1024m")`

#Run evaluations on the models:

options(java.parameters = "- Xmx1024m")

Evaluation1 <- evaluate\_Weka\_classifier(DTmodel1, numFolds = 10, seed = 1, class = TRUE)

Evaluation2 <- evaluate\_Weka\_classifier(DTmodel2, numFolds = 10, seed = 1, class = TRUE)

Evaluation3 <- evaluate\_Weka\_classifier(DTmodel3, numFolds = 10, seed = 1, class = TRUE)

Evaluation4 <- evaluate\_Weka\_classifier(DTmodel4, numFolds = 10, seed = 1, class = TRUE)

Evaluation5 <- evaluate\_Weka\_classifier(DTmodel5, numFolds = 10, seed = 1, class = TRUE)

#Examine our evaluations.

Evaluation1

Evaluation2

Evaluation3

Evaluation4

Evaluation5

# \*\*Tuning of the Model\*\*

#In the first part of our decision tree model, we created 5 randomly sampled models and tested #each one for the best accuracy. Now we will try to tune our best model, "DTModel1", to give us #the highest possible accuracy we can manage.

### \*\*Tuning Minimum Number of Instances(M) and Confidence Level(C)\*\*

#To begin, adjust the minimum number of instances, then adjust the confidence. Record the #accuracy below.

options(java.parameters = "- Xmx1024m")

DTmodel1 <- J48(rating~ n\_votes + n\_comments , data = train1, control=Weka\_control(U=FALSE, M=750, C=0.50))

Evaluation1 <- evaluate\_Weka\_classifier(DTmodel1, numFolds = 10, seed = 1, class = TRUE)

Evaluation1

# \*\*CHANGE M WHEN C=0.50, NumFolds=10\*\*

#M=2: 35.3367 %

# M=3: 35.3483 %

# M=5: 35.3550 %

# M=10: 35.3589 %

# M=50: 35.3972 %

# M=100: 35.4200 %

# M=500: 35.4817 %

# \*M=750: 35.5228 %\*

# M=1000: 35.4750 %

# \*\*CHANGING M WHEN C=0.25, NumFolds=10\*\*

# M=2: 35.4272 %

# M=10: 35.4428 %

# M=50: 35.4772 %

# M=100: 35.4828 %

# \*M=500: 35.5100 %\*

# M=750: 35.5011 %

# M=1000: 35.4728 %

# \*\*CHANGING M WHEN C=0.05, NumFolds=10\*\*

# M=2: 35.4561 %

# M=5: 35.4561 %

# M=10: 35.4561 %

# M=100: 35.4578 %

# \*M=500: 35.4606 %\*

# M=750: 35.4606 %

# M=1000: 35.4606 %

### \*\*Tuning the Cross-Fold Validation\*\*

# Based on our findings in the previous section, it would seem as though our highest accuracy #came from having our parameters set to C=0.50 and M=750. We now want to see how #changing # the number of folds in our cross fold validation will effect the model even further.

options(java.parameters = "- Xmx1024m")

DTmodel1 <- J48(rating~ n\_votes + n\_comments , data = train1, control=Weka\_control(U=FALSE, M=750, C=0.50))

Evaluation1 <- evaluate\_Weka\_classifier(DTmodel1, numFolds = 1000, seed = 1, class = TRUE)

Evaluation1

# \*\*CHANGING NUMFOLDS WITH C=0.50, M=750\*\*

# numFolds=5: 35.4283 %

# numFolds=10: 35.5228 %

# numFolds=50: 35.5461 %

# numFolds=100: 35.5728 %

# numFolds=500: 35.5894 %

# numFolds=750: 35.5889 %

# \*numFolds=1000: 35.5906 %\*

# numFolds=2000: 35.5906 %

# It would appear as though our best model is one in which we implement 1000 cross fold #validation to our previous optimal parameters.

# \*\*Apply to test\*\*

#The final step is to apply our model to the test set ands submit to Kaggle to see how well we #did...

#Run the prediction

pred <- predict (DTmodel1, newdata = test, type = c("class"))

#create output labels

myLabels <- c("review\_id")

myLabelCol <- test[myLabels]

newDTPred <- cbind(myLabelCol, pred)

colnames(newDTPred) <- c("review\_id", "rating")

#output predictions to CSV file.

write.csv(newDTPred, file="C:\\Users\\sarac\\OneDrive\\Documents\\Syracuse\\IST 707 Data Analytics--Applied Machine Learning\\Group Project\\DTFinalModel.csv", row.names=FALSE)

# \*\*Kaggle Score: 0.35347\*\*

# This is slightly less than anticipated given our 35.5906% accuracy rate in the test data.

**Appendix D – Sentiment Analysis Code:**

**Plot 1 A & B:**

good\_reads = read.csv("/Users/kbhager/OneDrive - Syracuse University/IST 707 machine/goodreads\_train.csv",stringsAsFactors = F)

ggplot(data=good\_reads,aes(x=review\_rating))+  
 geom\_bar(fill='purple')+  
 theme\_dark()

**Plot 2:**

good\_reads%>%  
 summarize(mean\_character = mean(nchar(review\_text)), median\_character = median(nchar(review\_text)))

## mean\_character median\_character  
## 1 1047.123 601

good\_reads%>%  
 summarize(mean\_words = mean(str\_count(string = review\_text,pattern = '\\S+')), median\_words = median(str\_count(string = review\_text,pattern = '\\S+')))

## mean\_words median\_words  
## 1 189.9818 110

good\_reads%>%  
 summarize(mean\_sentences = mean(str\_count(string = review\_text,pattern = "[A-Za-z,;'\"\\s]+[^.!?]\*[.?!]")), median\_sentences = median(str\_count(string = review\_text,pattern = "[A-Za-z,;'\"\\s]+[^.!?]\*[.?!]")))

## mean\_sentences median\_sentences  
## 1 12.76781 7

good\_reads %>%  
 select(id,review\_text) %>%  
 unnest\_tokens(output = word,input=review\_text) %>%  
 group\_by(id) %>%  
 count() %>%  
 head()

## # A tibble: 6 x 2  
## # Groups: id [6]  
## id n  
## <int> <int>  
## 1 1 363  
## 2 2 14  
## 3 3 466  
## 4 5 725  
## 5 6 256  
## 6 7 38

positive\_negative\_words = good\_reads%>%  
 select(id,review\_text)%>%  
 unnest\_tokens(output = word, input = review\_text)%>%  
 inner\_join(get\_sentiments('bing'))

## Joining, by = "word"

head(positive\_negative\_words)

## id word sentiment  
## 1 1 slow negative  
## 2 1 interesting positive  
## 3 1 love positive  
## 4 1 good positive  
## 5 1 fiction negative  
## 6 1 winner positive

ggplot(positive\_negative\_words,aes(x=sentiment,fill=sentiment))+  
 geom\_bar()+  
 guides(fill = F)+  
 theme\_dark()

**Plot 3 A & B:**

Sentiment\_of\_Reviews =   
 good\_reads %>%  
 select(id,review\_text,review\_rating)%>%  
 unnest\_tokens(output=word,input=review\_text)%>%  
 inner\_join(get\_sentiments('bing'))%>%  
 group\_by(review\_rating,sentiment)%>%  
 summarize(amount = n())%>%  
 mutate(proportion = amount/sum(amount))

## Joining, by = "word"  
## `summarise()` has grouped output by 'review\_rating'. You can override using the  
## `.groups` argument.

Sentiment\_of\_Reviews %>%  
 ggplot(aes(x=review\_rating,y=proportion,fill=sentiment))+  
 geom\_col()+  
 theme\_dark()

**Plot 4:**

good\_reads %>%  
 select(id,review\_text) %>%  
 unnest\_tokens(output = word, input = review\_text) %>%  
 inner\_join(get\_sentiments('nrc')) %>%  
 group\_by(sentiment) %>%  
 count()

## Joining, by = "word"

## # A tibble: 10 x 2  
## # Groups: sentiment [10]  
## sentiment n  
## <chr> <int>  
## 1 anger 626238  
## 2 anticipation 1111966  
## 3 disgust 450522  
## 4 fear 788766  
## 5 joy 1134028  
## 6 negative 1316504  
## 7 positive 2306915  
## 8 sadness 765937  
## 9 surprise 566342  
## 10 trust 1346562

good\_reads %>%  
 select(id,review\_text)%>%  
 unnest\_tokens(output = word, input = review\_text)%>%  
 inner\_join(get\_sentiments('nrc'))%>%  
 group\_by(sentiment)%>%  
 count()%>%  
 ggplot(aes(x=reorder(sentiment,X = n),y=n,fill=sentiment))+  
 geom\_col()+  
 guides(fill=F)+  
 coord\_flip()+  
 theme\_dark()

**Plot 5:**

wordcloudData1 =   
 good\_reads%>%  
 select(id,review\_text)%>%  
 unnest\_tokens(output=word,input=review\_text)%>%  
 anti\_join(stop\_words)%>%  
 group\_by(word)%>%  
 summarize(freq = n())%>%  
 arrange(desc(freq))%>%  
 ungroup()%>%  
 data.frame()

## Joining, by = "word"

set.seed(123)  
wordcloud(words = wordcloudData1$word,wordcloudData1$freq,scale=c(2,0.5),max.words = 300,colors=brewer.pal(8,"Dark2"))

**Plot 6:**

wordcloudData2 =   
 good\_reads%>%  
 select(id,review\_text)%>%  
 unnest\_tokens(output=word,input=review\_text)%>%  
 anti\_join(stop\_words)%>%  
 inner\_join(get\_sentiments('bing'))%>%  
 count(sentiment,word,sort=T)%>%  
 spread(key=sentiment,value = n,fill=0)%>%  
 data.frame()

## Joining, by = "word"  
## Joining, by = "word"

rownames(wordcloudData2) = wordcloudData2[,'word']  
wordcloudData2 = wordcloudData2[,c('positive','negative')]  
wcd2<-as.matrix(wordcloudData2)

set.seed(123)  
comparison.cloud(term.matrix = wcd2,scale = c(2,0.5),max.words = 250, rot.per=0,colors=brewer.pal(8,"Dark2"))

**Appendix E – Failed Plots Code:**

**Plot 1:**

#Remove date\_added, date\_updated, read\_at, started\_at, review\_id, review\_text

clusterRatingVotesComments <- gr\_train[,c("book\_id","rating","n\_votes","n\_comments")]

#keep only unique book\_id values, but average the rating of all the instances of that particular book\_id from the big dataset

#source: https://r-coder.com/aggregate-r/

condensed\_meanRating <- aggregate(clusterRatingVotesComments$rating,by=list(clusterRatingVotesComments$book\_id),data=clusterRatingVotesComments,FUN=mean)

condensed\_meanVotes <- aggregate(clusterRatingVotesComments$n\_votes,by=list(clusterRatingVotesComments$book\_id),data=clusterRatingVotesComments,FUN=mean)

condensed\_meanComments <- aggregate(clusterRatingVotesComments$n\_comments,by=list(clusterRatingVotesComments$book\_id),data=clusterRatingVotesComments,FUN=mean)

condensed <- merge(condensed\_meanRating,condensed\_meanVotes,by="Group.1")

colnames(condensed) <- c("Group.1","rating","votes")

condensed <- merge(condensed,condensed\_meanComments,by="Group.1")

colnames(condensed) <- c("Group.1","rating","votes","comments")

#change the book\_id to the row values

rownames(condensed) <- condensed$Group.1

#get rid of the book\_id column cause it's now the row label

condensed$Group.1<-NULL

#testing condensed\_mean code

#newVector <- preprocessedData[preprocessedData$book\_id == 2, ]

#mean(newVector$rating)

#use the kMeans algorithm in R

d <- dist(condensed\_meanRating, method = "euclidean")

# Hierarchical clustering using Complete Linkage

hc1 <- hclust(d, method = "complete" )

# Plot the obtained dendrogram

plot(hc1, cex = 0.6, hang = -1)