Jeremy Guinta (ID 604882679)
June 7, 2018

STAT 412 - Final Project
Modeling New York Times Comment Recommendations

Contents

Executive Summary

Attempting to predict user reaction, in the form of recommendations to New York Times article comments between January 2018 to May 2018, is a challenging endeavor. The feature building focused on identifying well written, timely comments made on popular articles, and then modeling the data using a Gradient Boosted Machine and Random Forest. For most of the data, an accurate portrayal of the number of recommendations for any comment is feasible. However, the prediction accuracy for comments with an extreme number of recommendations appears to be poor.

Data Relied Upon

There are two main sources of data used for this exercise, and no external data was used:

1. train_comments.csv
2. train_articles.csv

The train_comments.csv file contains 665,396 observations. The data contains information specific to a comment made on an article for New York Times articles published between 2018-01-04 and 2018-04-02.

The train_comments.csv looks like the following:

Variable Name	Variable Type	Short Description
approveDate	int	1519852022, 1518469135, 1518385379, 1517…
articleID	chr	“5a7101c110f40f00018be961”, “5a7101c110f…
articleWordCount	int	1322, 1322, 1322, 1322, 1322, 1322, 1322…
commentBody	chr	“I typically strongly dislike articles w…
commentID	dbl	26156416, 25930059, 25912292, 25864174, …
commentSequence	dbl	26156416, 25930059, 25912292, 25864174, …
commentTitle	chr	“”, “”, “”, “”, “<br…
commentType	chr	“comment”, “comment”, “comment”, “commen…
createDate	int	1519849555, 1518458548, 1518304930, 1517…
depth	int	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
editorsSelection	chr	“False”, “False”, “False”, “False”, “Fal…
inReplyTo	int	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
newDesk	chr	“Travel”, “Travel”, “Travel”, “Travel”, …
parentID	dbl	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
parentUserDisplayName	chr	, , , , , , , , , , …
permID	chr	“26156416”, “25930059”, “25912292”, “258…
picURL	chr	“https://graphics8.nytimes.com/images/ap…
printPage	int	5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
recommendations	int	1, 0, 1, 0, 12, 1, 0, 2, 6, 4, 3, 2, 0,
recommendedFlag	lgl	, , , , , , , , , , …
replyCount	int	0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
reportAbuseFlag	lgl	, , , , , , , , , , …
sectionName	chr	“Unknown”, “Unknown”, “Unknown”, “Unknow…
sharing	int	0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,

status	chr	“approved”, “approved”, “approved”, “app…
timespeople	int	0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
trusted	int	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
typeOfMaterial	chr	“News”, “News”, “News”, “News”, “News”, …
updateDate	int	1519852022, 1518469135, 1518385379, 1517…
userDisplayName	chr	“Emma Claire Lisk”, “Eyes Open”, “Mark P…
userID	dbl	83288014, 53167641, 44043675, 84748907, …
userLocation	chr	“Wilmington, NC”, “San Francisco”, “Chic…
userTitle	chr	“”, “”, “”, “”, “”, “”, “”, “”, “”, “”, …
userURL	lgl	, , , , , , , , , , …

The train_article.csv file contains 3,445 observations. The data contains information specific to a comment made on an article for New York Times articles published between 2018-01-04 and 2018-04-02.

The train_article.csv looks like the following:

Variable Name	Variable Type	Short Description
articleID	chr	“5a7101c110f40f00018be961”, “5a70fc1210f40f00…
articleWordCount	int	1322, 1308, 228, 1114, 777, 1501, 754, 842, 9…
byline	chr	“By SHANNON SIMS”, “By ALAN RAPPEPORT and THO…
documentType	chr	“article”, “article”, “article”, “article”, “…
headline	chr	“Rhythm of the Streets: â ˜Weâ ™re Warrior Wo…
keywords	chr	“[‘Bahia (Brazil)’, ‘Music’, ’Women and Girls…
multimedia	int	68, 68, 0, 61, 68, 68, 68, 65, 68, 66, 68, 68…
newDesk	chr	“Travel”, “Washington”, “Metro”, “Editorial”,…
printPage	int	5, 17, 16, 24, 0, 1, 0, 0, 22, 11, 2, 19, 5,
pubDate	chr	“2018-01-30T23:37:31Z”, “2018-01-30T23:13:14Z…
sectionName	chr	“Unknown”, “Politics”, “Unknown”, “Editorials…
snippet	chr	“Meet the all-female Brazilian drum group tha…
source	chr	“The New York Times”, “The New York Times”, “…
typeOfMaterial	chr	“News”, “News”, “News”, “Editorial”, “Op-Ed”,…
webURL	chr	“https://www.nytimes.com/2018/01/30/travel/br…

As will be discussed later, the datasets are used to create features for the modeling process. The information from the article data is combined with the information from the comments via the articleID field.

Data Exclusions

During the combination process to link information from the train_article.csv data to the train_comments.csv data via the articleID field, 240 articleIDs found in the train_comments.csv data were *not* found in the train_article.csv data. These articles were removed from the analysis. The total number of original train_comments.csv data was reduced from 665,396 to 640,904.

Data Splitting

The testing dataset provided along with the training dataset was not sufficient for feature building and modeling tuning because it was the final submission data. To develop the model and features the original training dataset was divided using a 70/30% split into a training set and a validation set. The final training set contained 448,633 observations. The validation set contained 192,271.

All feature engineering, exploratory data analysis, and model tuning are performed on the training set. The validation set was held out specifically to determine model performance

Features Generation and Purpose

Theory

Generally, articles that are discussing popular topics should get more views and page hits. As a result, these articles should get more comments and more comment recommendations. Furthermore, comments that are made first are likely to be viewed more often, and thus should receive more recommendations. Additionally, how the comment is written and the sentiment of the comment may influence the number of recommendations. With these factors in mind, features were developed out of the data to identify popular content with well written, timely comments.

Features

Article Features

1. *Keyword Rank* - This feature puts a numerical score on the keywords extracted from the *Keyword* field contained within the train_article.csv dataset. For example, the most popular keyword is “Trump, Donald J” and the second most popular is "United States Politics” This is not surprising as the news has been very focused on politics and the President’s dealings since his election. As another example, one of the least popular keywords is “Eyes and Eyesight.” The keyword rank is a composite score of all keywords tagged to the article. Therefore, if an article has many popular keywords then it will be ranked higher than an article with fewer popular keywords. The overall purpose of this feature is to identify popular articles based on the keywords that were tagged to the article.

2. *Keyword* – This feature extracts each of the keywords from the *Keyword* field contained within the train_article.csv dataset. Once the keywords are extracted, they are ordered by most popular to least popular keyword based on the numerical score of each keyword. Then the top three most popular keywords by the article are transformed into three categorical variables.

3. *Topic Analysis* - This feature puts each of the articles into a more defined bucket of categories. The keywords of each article were used to define each article into a broad topic. The following topics were defined:

topic
1. Politics
2. Economy
3. National
4. International
5. Entertainment
6. Science and Technology
7. Food
8. Lifestyle
9. Local
10. Other

3. *Time and Day of the Article* - This feature puts the date, time, the day of the week, and time of day categories for each article. The purpose of this feature is to divide up the data into groups under the assumption that articles published, for example, Monday morning) may get more attention than an article published for example, late Friday night.

4. *Sentiment of the Article* - This feature puts a numerical and categorical ranking on each article based on the *Snippet* field. The purpose is to rank the articles from “Very Negative” to “Very Positive.” However, due to the limited number of words of the *Snippet* field, this feature may not be useful.

<u>Comment Features</u>

1. *Number of Users Commenting on an Article* - This feature calculates the unique number of users based on the *userDisplayName* that comment on the article. The purpose is to identify popular articles that are being viewed on and commented with greater frequency than other articles.

2. *Rank Order of the Comments* - This feature puts a numeric ranking on each comment on the article based on the sequence order of the comment. The purpose is to divide the data into categories that rank comments closer to the top of the list separately from comments made towards the end.

3. *Time to Post* - This feature calculates the time it takes from the publish date to when the post occurs. This is similar to the rank order but differs as it is based on time. The purpose is to differentiate comments that are made close to when the article is published (and thus more likely to be seen) versus comment made hours or days later (and thus less likely to be seen).

4. *Sentiment Analysis* - This feature puts a numerical score on the sentiment of an article. The sentiment function from the sentimentr package (https://cran.r-project.org/web/packages/sentimentr/sentimentr.pdf) was used.

5. *Comment length* - This feature calculates the string length of the comment. The purpose is to separate the data based on long comments versus shorter comments.

6. *Written Grade Level* - This feature put a numerical grade level score on each comment (e.g. the comment was written at the 8th-grade level). The textstat_readability from the quanteda package (https://cran.r-project.org/web/packages/quanteda/quanteda.pdf) was used. Flesch Kincaid and Coleman Liau scores were calculated for each comment.

Other Features

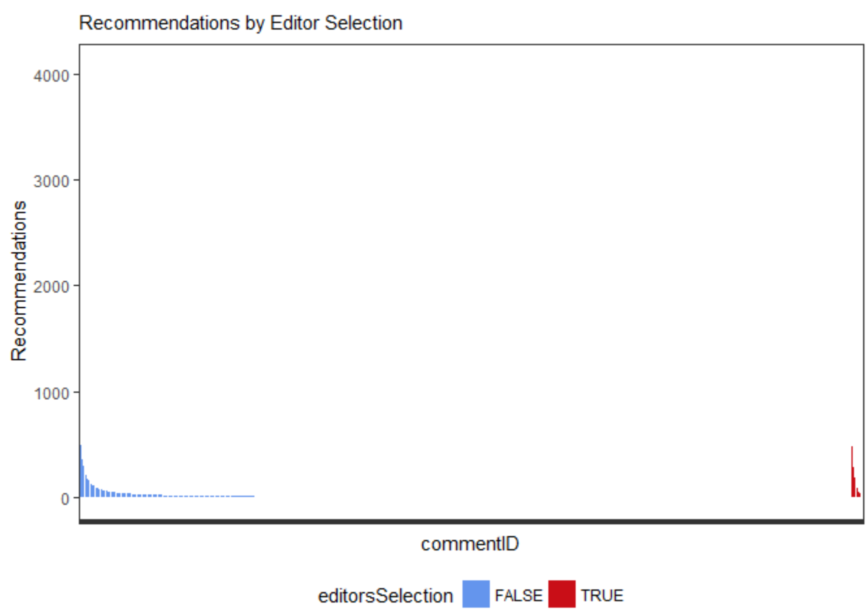
The data contained other features that were useful without any modification. The *editorsSelection*, *replyCount*, *newDesk*, and *articleWordCount* all appeared to be useful variables for analysis.

Analysis of Features

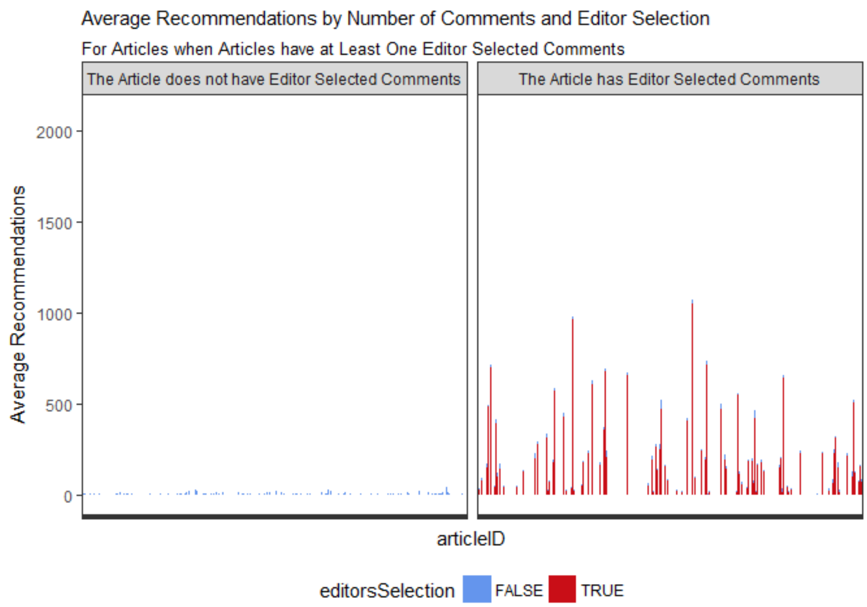
For purposes of the following graphical analysis, a summary metric, average recommendations, was developed. This metric computes the average number of recommendations across all comments for an article.

The first feature analyzed is the *editorSelection* variable. The following graphs compare the *editorSelection* variable for all comments and articles against total recommendations and average recommendations.

Graph 1 - All Recommendations by Editor Selection



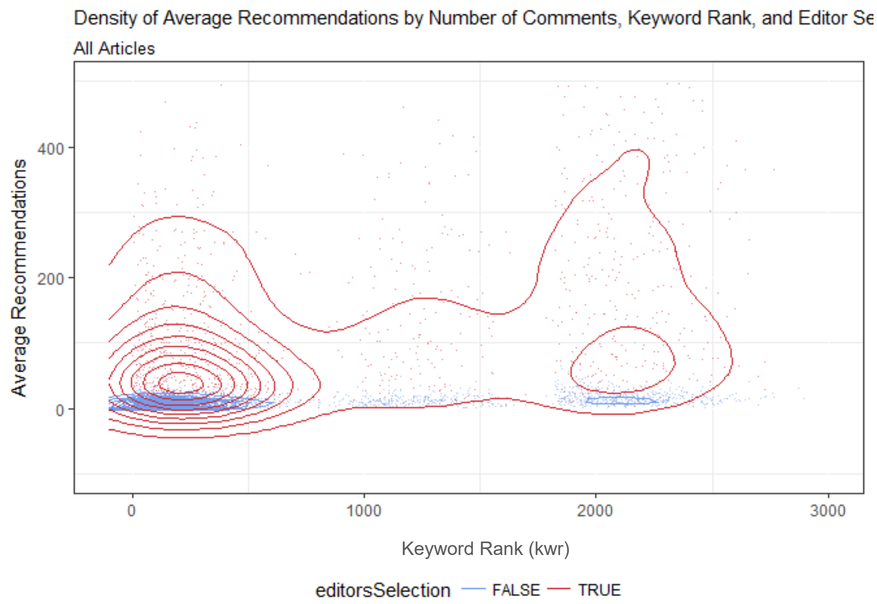
Graph 2 - Article Average Recommendations by Editor Selection



Both graphs above show that comments made on articles that have an editor's selection designation receive many more recommendations than articles that do not have an editor selection. The first graph also shows that many comments never receive or receive very few recommendations. It also shows that there are extreme outliers with the number of recommendations. Certain comments have thousands of recommendations while others have zero.

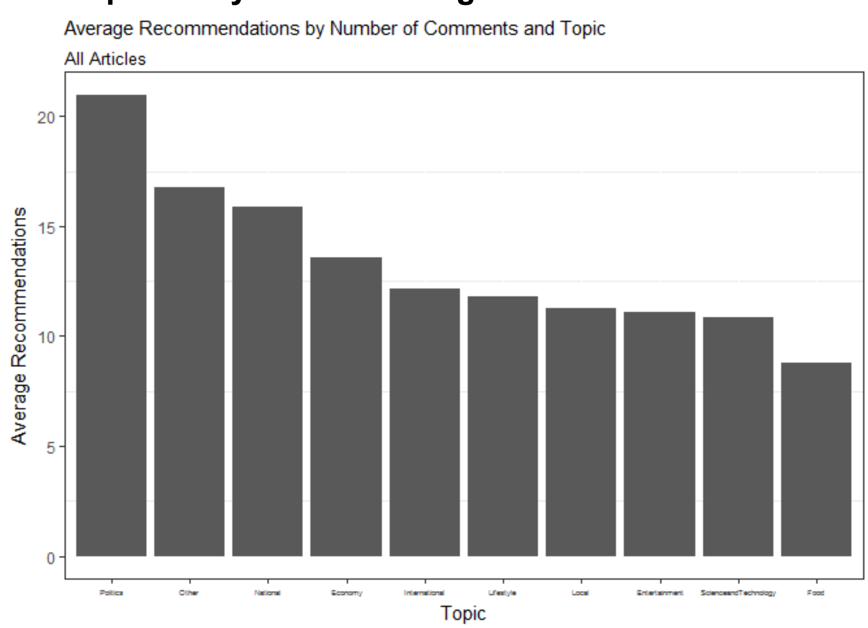
The next feature is keyword rank. The below graph is a density plot comparing the average number of recommendations by article against the keyword rank. The larger the keyword rank the better. As this graph suggests the clear majority of red (editor selected comments) have a significantly higher average number of recommendations than non-editor selected comments. This is true at all keyword rank values.

Graph 3 - Keyword Rank by Recommendations and Editor Selection



Graph 4 shows the results of topic analysis and average recommendations. This process took the keywords from each article and categorized them into general categories. The graph below shows that average number of recommendations by each topic.

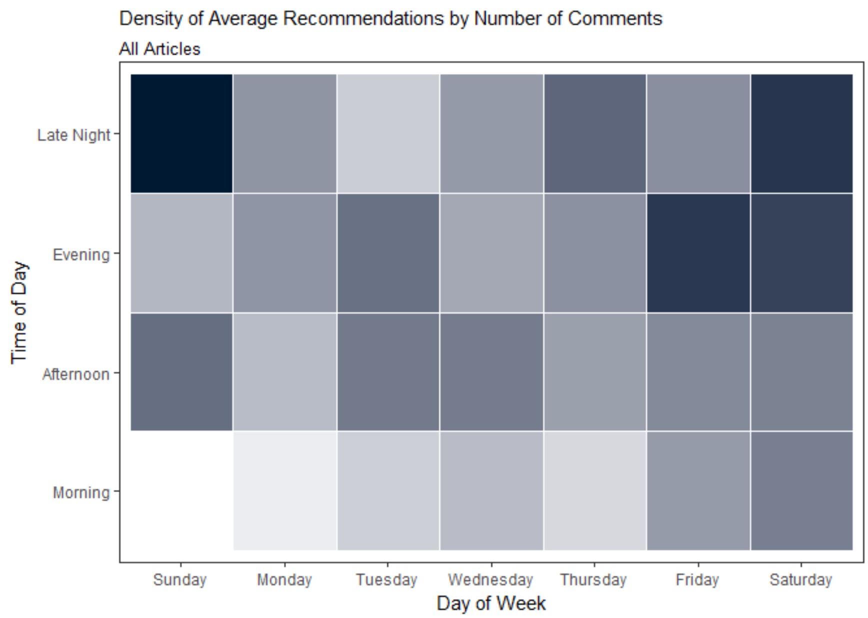
Graph 4 - Topic Analysis and Average Number of Recommendations



Jeremy Guinta (ID 604882679)
June 7, 2018

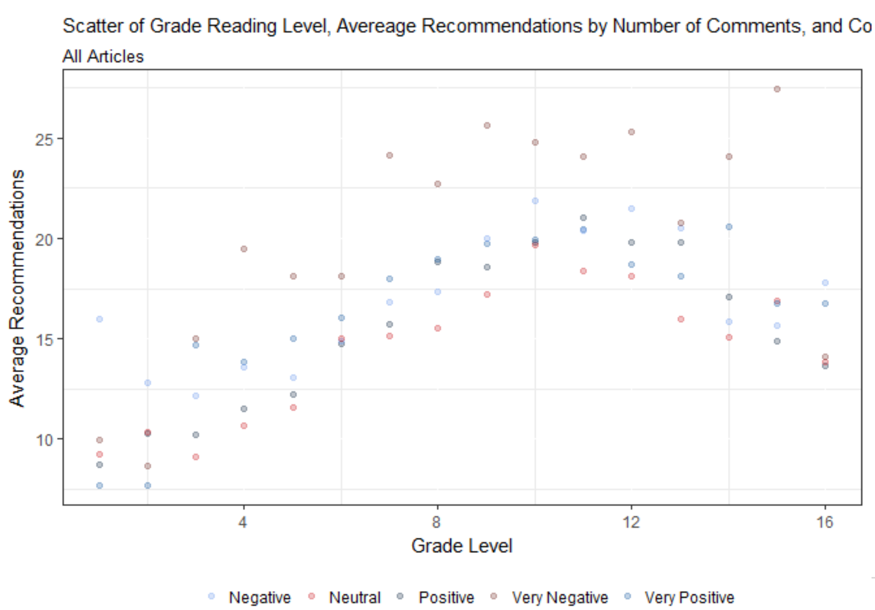
Graph 5 shows the density of average recommendations by the day of the week and the time of day category based on the publishing date of the article. Certain time periods and days of weeks when an article is published can impact the average number of recommendations that any comment for that article received.

Graph 5 - Time of Day and Day of Week and the Average Number of Recommendations



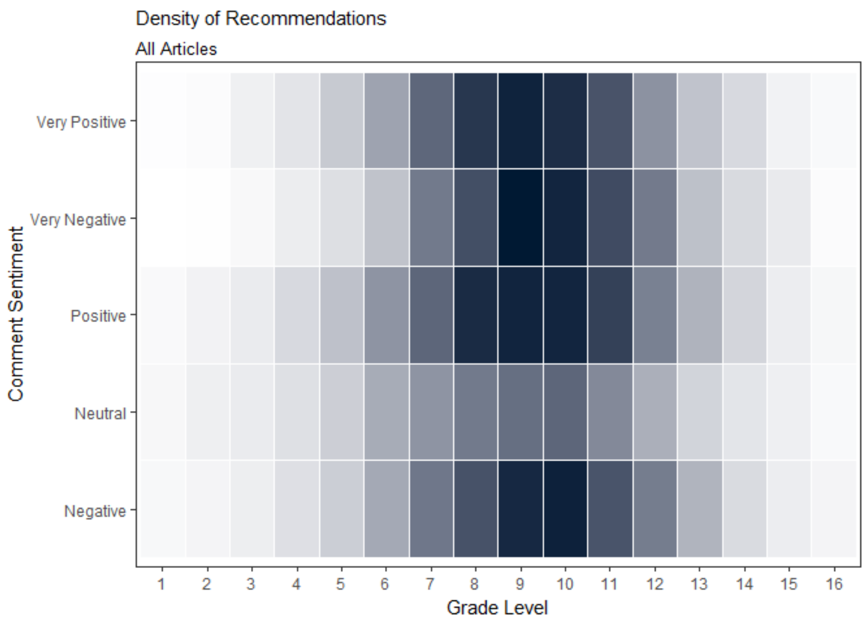
Graph 6 shows that average recommendation for comments across written grade level and sentiment of the comment. Although there is no clear pattern of the sentiment variable, there is a pattern base d on the written grade level of the comment. Comments in the 8 to 12 grade written level tend to score higher average recommendations than comments written at higher or lower grade levels.

Graph 6 - Grade Level, Sentiment Analysis and Recommendations



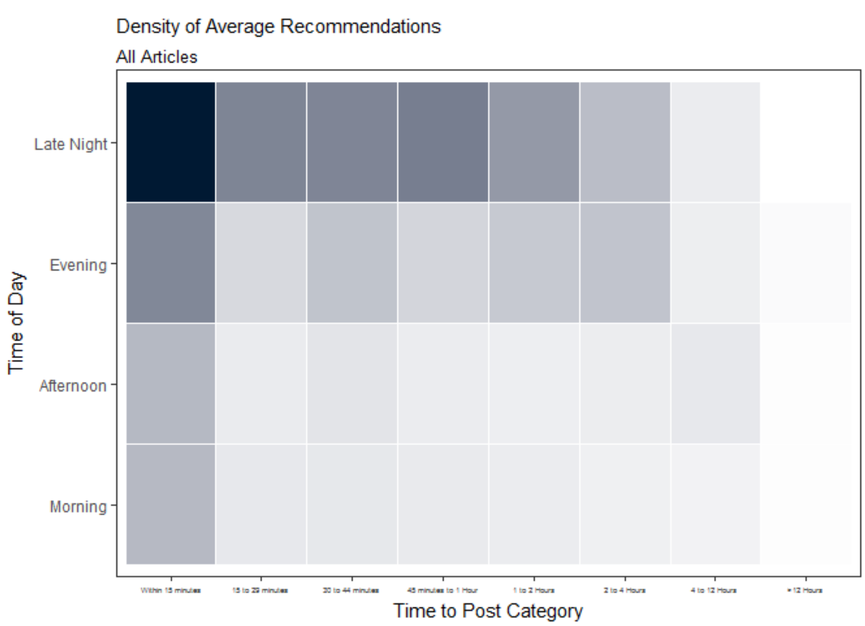
In another look at the same variables, Graph 7 below suggests that the comment sentiment is consistent across the average number of recommendations, but the written grade level is very important when determining the average number of recommendations.

Graph 7 - Grade Level, Sentiment Analysis and Recommendations



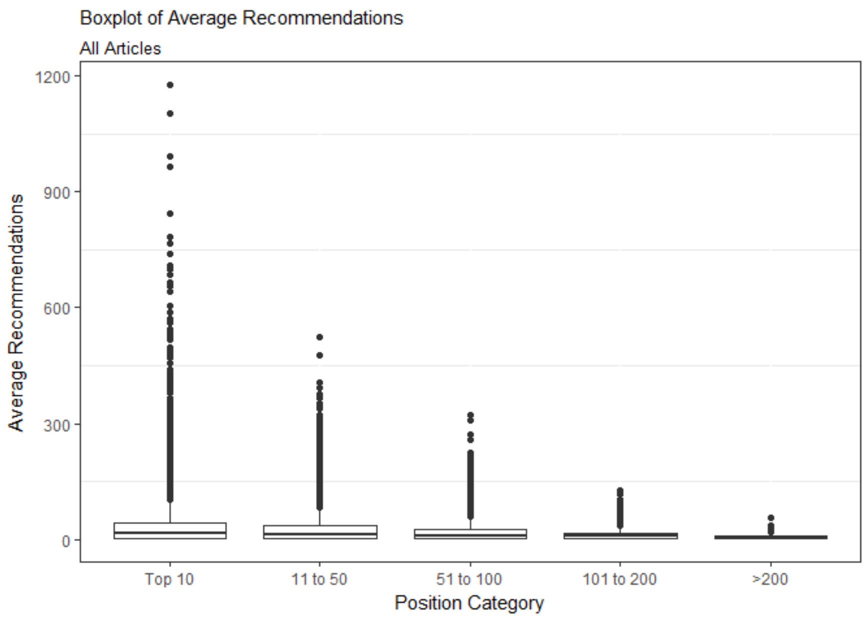
Graph 8 demonstrates a clear pattern that posts that are within 15 minutes of the publishing date and time tend to have higher average recommendations than other comments. More importantly, comments made hours after the publishing of the article tend to have lower average recommendations than comments made closer to the publishing date and time.

Graph 8 - Time of Day, Time to Post, and Recommendations



Graph 9 - Comment Position and Recommendations

This final graph shows that comments positioned towards the top of the order have higher than average recommendations than comments that are made later. It also shows that recommendation counts have extreme outliers.



Each of the proceeding analyses suggests that the initial theory is correct: Well written, timely comments on popular articles are going to have more recommendations.

Machine Learning Model

General Approach

The purpose of the model is to predict the number of recommendations based on features of the article and the comment. A Gradient Boosted Machine model and a Random Forest model were proposed, tuned, and tested to minimize prediction error using Mean Absolute Error (MAE) as the error metric. h2o was employed as the front-end to the models. h2o has a series of advantages over other packages that allow for many models to be developed at once.

1. h2o allows randomized grid searching with a wide variety of parameters.
2. h2o automatically standardized all dependent variables.
3. h2o allows for various distributions to be assumed into the model.
4. h2o automatically encodes all categorical variables.

All models were trained for four to eight hours using MAE as the stopping metric. Five-fold cross-validation was performed on each model run as well. The best model was determined by the model that produced the lowest MAE in the cross-validation process.

The following variables were modeled against the number of recommendations for each comment.

Variable	Description
articleWordCount	Word count of the article
commentType	Categorical variable describing the type of comment
depth	The depth of the comment
editorsSelection	Categorical variable TRUE/FALSE
newDesk	The department that published the article
replyCount	The number of replies a comment receives
sectionName	Newspaper section
timespeople	Unknown 0/1 indicator
trusted	Unknown 0/1 indicator
com_ord	Comment order by article
com_pos_cat	Categorical variable of Comment order
time_to_post	Time in minutes to post comment from publish date and time
time_to_post_cat	Categorical variable of time to post
comment_length	String length of entire comment
readFR	Flesch Kincaid grade written level
readCL	Coleman Liau grade written level
com_sent	Comment sentiment
com_cat	Categorical breakdown of comment sentiment
kw1	First keyword after the keyword reorganization
kw2	Second keyword after the keyword reorganization
kw3	Third keyword after the keyword reorganization
kwr1	First keyword rank after the keyword reorganization
kwr2	Second keyword rank after the keyword reorganization
kwr3	Third keyword rank after the keyword reorganization
timeofday	Categorical variable for the time of day of the article is published
dow	Day of the week of the article is published
topic	General topic based on keywords
specific	Specific topic based on keywords
kwr	Composite keyword ranking for the article
minkwr	Lowest ranking keyword for the article
snip_sent	Article snippet sentiment
snip_cat	Categorical breakdown of article sentiment

Gradient Boosted Machine

The selected GBM model has the following parameters.

Parameter	Value
ntrees	87 – Total number of trees
max_depth	20 – Maximum depth of each Tree
min_rows	5 – Minimum number of observations for a leaf
nbins_cats	64 – Total categorical bins

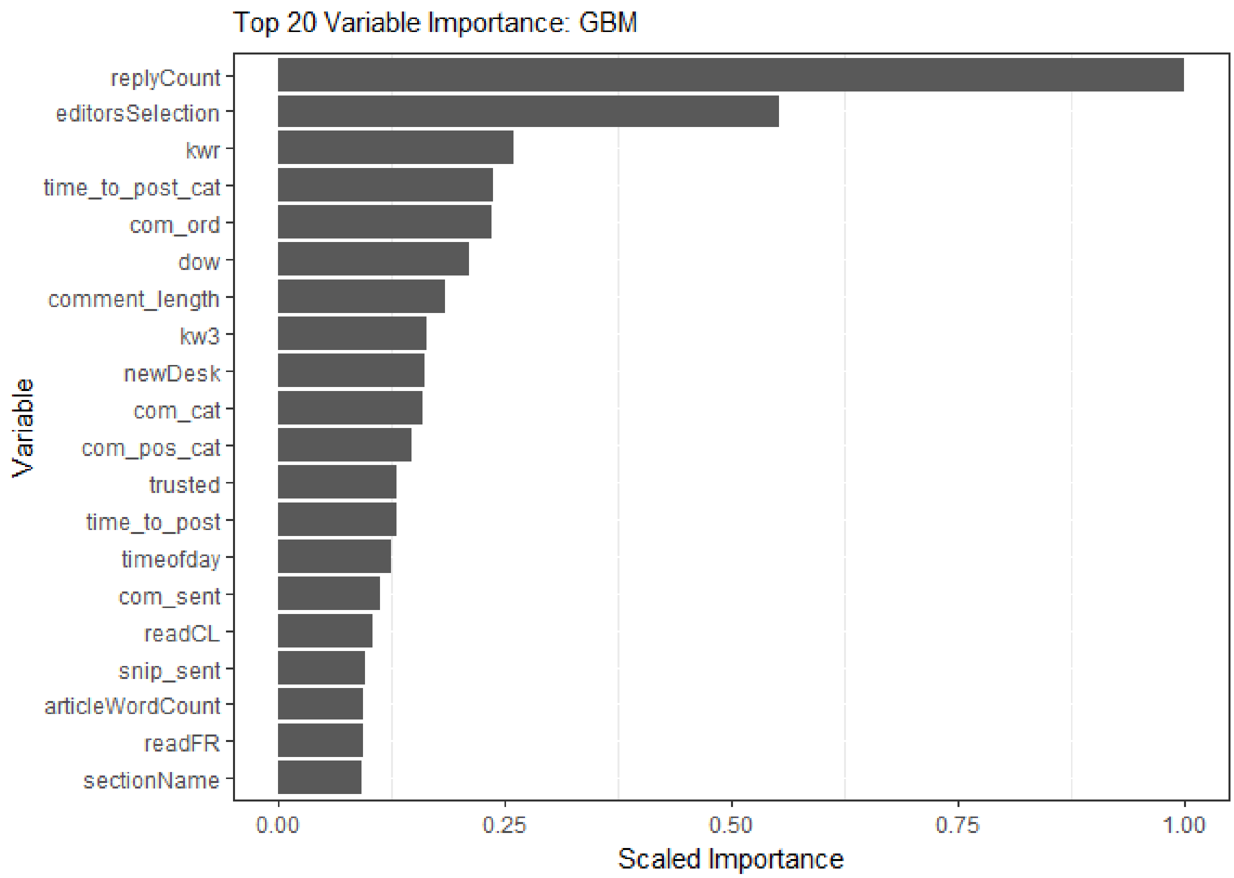
nbins	20 – Total numerical bins
stopping_metric	MAE – Mean Absolute Error
distribution	Poisson – Distribution of the loss function
sample_rate	0.99 – Row sampling rate.
col_sample_rate	0.6 – Column sampling rate.
col_sample_rate_per_tree	0.9 – Columns to sample per tree
learn_rate	0.09 – Learning rate of each iteration
learn_rate_annealing	1 – Adjustment to the learn rate

The GBM model with the best cross-validated MAE is selected as the "best" model. For this model the overall cross-validated training MAE is:

MAE Category	MAE
mean	13.18
cv 1	13.01
cv 2	13.20
cv 3	13.52
cv 4	13.09
cv 5	13.10

Using this model and predicting recommendations and then calculating the MAE on the validation set results in a validation MAE of 13.19.

This model uses many variables. The variable importance metrics that GBM creates can be used to determine if any of the developed features are important.



The developed keyword rank (*kwr*) feature ranked third in importance after existing features *replyCount* and *editorSelection*. Other developed features *time_to_post_cat*, *com_ord*, *dow*, and *comment_length* all ranked in the top ten. Surprisingly, *com_sent* and *readFR* ranked in the bottom half of the top 20 importance variables.

Random Forest

The selected RF model has the following parameters.

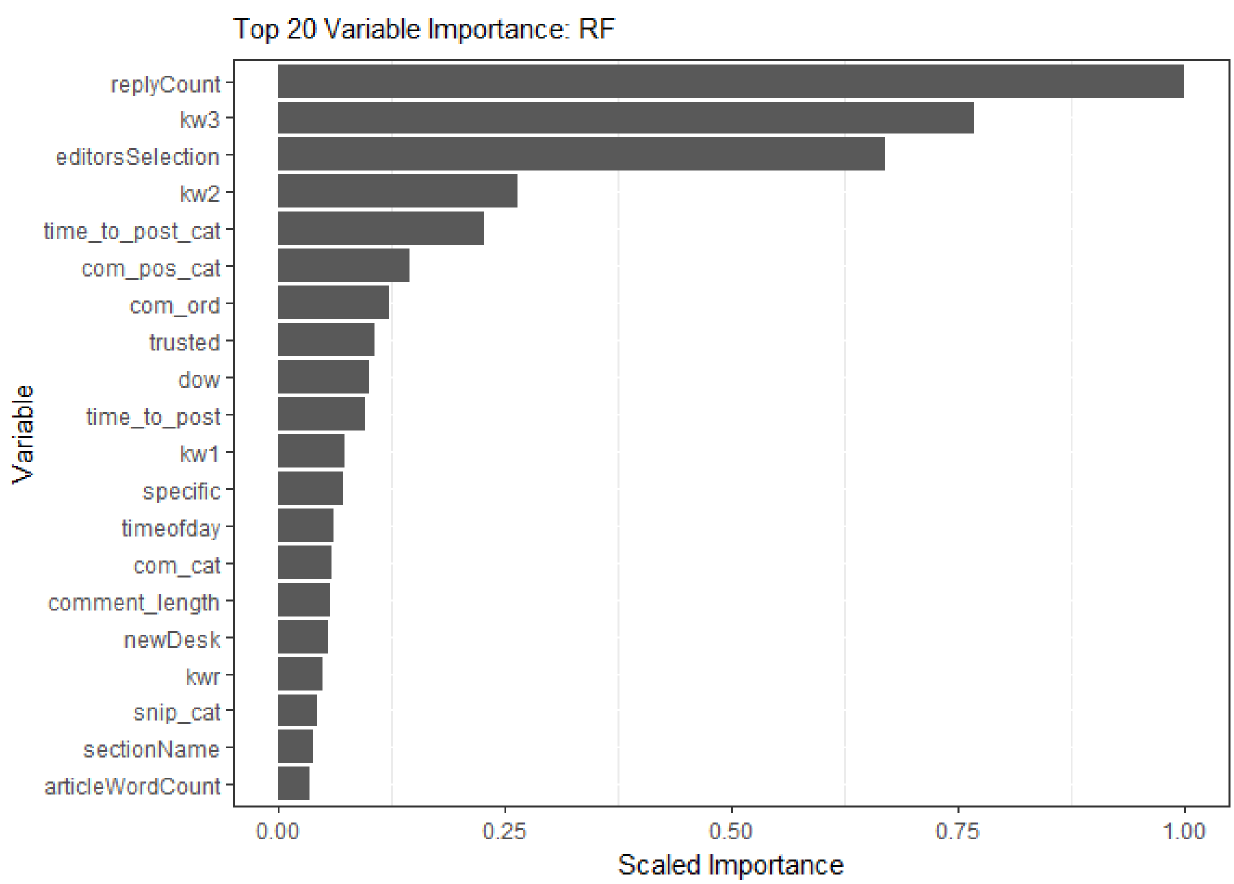
Parameter	Value
ntrees	65 – Total number of frees
max_depth	20 – Maximum depth of each tree
min_rows	10 – Minimum number of observations for a leaf
nbins_cats	1536 – Total categorical bins
nbins	20 – Total numerical bins
stopping_metric	MAE – Mean Absolute Error
distribution	Poisson – Distribution of the loss function
sample_rate	0.995 – Row sample rate
col_sample_rate_per_tree	0.8 – Column sample rate

The RF model with the best cross-validated MAE was selected as the "best" model. For this model the overall cross-validated training MAE is:

MAE Category	MAE
mean	14.48
cv 1	14.33
cv 2	14.47
cv 3	14.85
cv 4	14.30
cv 5	14.45

Using this model and predicting recommendations and then calculating the MAE on the validation set results in a validation MAE of 14.36.

This model uses many variables. The variable importance metrics that RF creates can be used to determine if any of the developed features are important.



Other than *replyCount* and *editorsSelection* ranking very high, the RF importance plot shows dramatically different variables as being important. *kw2* and *kw3* rank much higher in the RF model than in the GBM model. Other developed features *time_to_post*, *time_to_post_cat*, *com_ord*, and *dow* all ranked in the top ten. Surprisingly, *com_sent* and *readFR* do not rank in the top 20. Additionally, *com_cat* and *kwr* rank towards the bottom of the top 20.

Other Models

There were other models considered but ultimately ruled out as independent models for consideration. Linear Regression and Neural Net models were trained using the same variables previously described. The overall MAE for cross-validated training, validation, and Kaggle submission for each of these models were higher than the GBM and RF models selected.

Additionally, an autoML process was also conducted. This process was automated through h2o and runs a Gradient Boosted Machine, Random Forest, Deep Random Forest, Linear Regression, and Neural Net using randomized grid search with randomized tuning parameters. The process then ensembles all models into a single super learner. Surprisingly the results of this process were not better than the stand-alone GBM or RF models ultimately selected.

Model Error

GBM

Most of the error in the GBM model is driven by comments with extreme numbers of recommendations. Using the validation set, the number of comments is categorized, and then the MAE is calculated within each number of comments category. The results below show that error is being driven by a relatively small percentage of comment (as a percentage of all comments analyzed). The Greater than or equal to 200 category makes up 1.6% of the data, but 43.1% of the absolute error from the model.

Recommendation Category	MAE	AE	# of Comments	% of Comments	% of Error
Less than 10	3.34	475,963	142,660	74.2%	18.8%
Between 10 and 19	8.34	195,172	23,388	12.2%	7.7%
Between 20 and 49	18.64	284,918	15,284	7.9%	11.2%
Between 50 and 99	44.33	224,908	5,073	2.6%	8.9%
Between 100 and 199	93.42	262,799	2,813	1.5%	10.4%
Greater than or equal to 200	349.31	1,094,051	3,132	1.6%	43.1%

RF

Just like the GBM model, most of the error in the RF model is driven by comments with extreme numbers of recommendations. Using the validation set, the number of comments is categorized, and then the MAE is calculated within each number of comments category. The results below show that error is being driven by a relatively small percentage of comment (as a percentage of all comments analyzed). The Greater than or equal to 200 category makes up 1.6% of the data, but 32.7% of the absolute error from the model.

Recommendation Category	MAE	AE	# of Comments	% of Comments	% of Error
Less than 10	5.44	775,667	142,660	74.2%	28.1%
Between 10 and 19	11.37	265,812	23,388	12.2%	9.6%
Between 20 and 49	22.15	338,584	15,284	7.9%	12.3%
Between 50 and 99	46.24	234,575	5,073	2.6%	8.5%
Between 100 and 199	86.92	244,507	2,813	1.5%	8.8%
Greater than or equal to 200	288.56	903,771	3,132	1.6%	32.7%

Summary and Conclusion

The two models performed well in terms of the training and validation MAE, and several developed features demonstrate high relative importance in each of the models conducted. However, both models performed poorly when predicting the number of recommendations for comments when the actual value of recommendations was extremely high. The Kaggle scored MAE for both models was relatively close to the training and validation MAE. This suggests that the models were not overfitted.

The final submitted Kaggle competition models were:

1. GBM model. This model had a public Kaggle score of 14.59.
2. A simple ensemble of the GBM and the RF models. This model had a public Kaggle score of 14.95.

Appendix A – Programming Code

1. 000_mk_function.r – Loads user created functions
2. 001a_mk_data1.r – Data building and feature engineering
3. 001b_an_data1.r – Exploratory data analysis
4. 002_an_model.r – Initial Modeling: Linear Regression, Gradient Boosting Machine, Random Forest
5. 003_an_model.r – Initial Modeling: Neural Net, h2o AutoML
6. 004_an_model.r – Final Modeling: Gradient Boosting Machine, Random Forest
7. 005_an_model.r – Ensemble models
8. 006_an_lime.r – Lime explanation of select records
9. 007_an_report.Rmd – R Markdown that produces the first draft of this document

```r
########################/##############################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                  #
#FileName       -   001_mk_data.r                                                        #
#By             -   Jeremy Guinta (ID 604882679)                                         #
#                                                                                        #
#Last Update Date:  5/11/2017                                                            #
#                                                                                        #
#Purpose:       -   Build data and features                                              #
#               -   Proposed theory: Well written, timely comments on popular articles will have    #
#                   more recommendations.  Build features to capture these effects       #
#                                                                                        #
#               -   Article Features:                                                    #
#                   1. Key  Words - Parse key word string and reorg into a key word ranking. Rank   #
#                   each article by a key word rank that will separate popular articles away from   #
#                   less popular articles on a continuous scale.                         #
#                   2. Topic Analysis - Group articles in general and specific topics    #
#                   3. Date / Time / DOW / Time of Day of the article based on publish date #
#                   4. Sentiment Analysis on article snippet                             #
#                                                                                        #
#               -   Comment Features                                                     #
#                   1. Number of unique users posting on an article                      #
#                   2. Rank order of the comment                                         #
#                   3. Date / Time / Time to Post (from publish date) of a comment       #
#                   4. Sentiment Analysis (comment positive or negative)                 #
#                   5. Comment length                                                    #
#                   6. Reading Grade level of the comment                                #
#                                                                                        #
#               -   Useful Existing Features                                             #
#                   1. Editors Selection                                                 #
#                   2. replyCount                                                        #
#                   3. newDesk                                                           #
#                   4. articleWordCount                                                  #
#########################################################################################



#I. Setup -----------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")
```

```
    #Package Install
    require(grid)           #Plotting utilities
    require(gridExtra)      #Plotting utilities
    require(tidyverse)      #All things tidy
    require(data.table)     #Data table is better
    require(dtplyr)         #Make sure Data table and dplyr work together
    require(ggplot2)        #Graphing Utilities
    require(stringr)        #String Functions
    require(reshape2)       #Data Reshape
    require(GGally)         #Correlation
    require(sentimentr)     #Sentiment Analysis
    require(quanteda)       #Readability Scores
    require(lubridate)

    #Set Options
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
                axis.text.x = element_text(size= rel(1.0)),
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                              "#999999", "#000000",  "#800000", "#B23232")

    #Custom Functions
    source("./000_mk_functions.r")  #Loads Text to Columns

#II.  Data Loading ----------------------------------------------------------------------------
trn_art<-fread("./train_articles.csv")
trn_art[, type:="trn"]
trn_com<-fread("./train_comments.csv")
trn_com[, type:="trn"]
tst_art<-fread("./test_articles.csv")
tst_art[, type:="tst"]
tst_com<-fread("./test_comments.csv")
tst_com[, type:="tst"]

#III. Data Processing ---------------------------------------------------------------------------
#A. Combine the Data - For convenience of feature building

    #Articles
    art<-bind_rows(trn_art, tst_art)
    art[, .N, by=type]
```

```
       # type    N
    # 1:  trn 3445
    # 2:  tst 1324

    #Comments
    trn_com[, editorsSelection:=ifelse(editorsSelection=="0" | editorsSelection=="False", FALSE,
                                ifelse(editorsSelection=="1" | editorsSelection=="True", TRUE, NA))
        ]
    com<-bind_rows(trn_com, tst_com)
    com[, .N, by=type]
       # type      N
    # 1:  trn 665396
    # 2:  tst 264924

#B. Article Features

    #1. Keywords

        #a. Split the Keywords
        art[, keywords:=gsub("\\[", "", keywords)]
        art[, keywords:=gsub("\\]", "", keywords)]
        art[, keywords:=gsub("',", "|", keywords)]
        art[, keywords:=gsub("'", "", keywords)]

        art<-text_to_columns(as.data.frame(art), column="keywords", delimiters=c("|"))
        art<-as.data.table(art)

        #b. Loop through the data, combine the keywords and sort them
        tot<-nrow(art)
        art[, ord:=1:nrow(art)]
        for (i in 1:tot) {
            print(i)
            tmp<-art[ord==i, c(1, 7:30)]
            for (j in 1:24) {  #There are 24 "new" values created from the text_to_columns
                    val<-paste("new", j, sep="")
                    chk<-tmp[, .(articleID, new=get(val))]
                    chk[, new:=as.character(new)]
                if (j ==1) {
                    out<-chk
                }
                else {
                    out<-rbind(out, chk)
                }
            }
            out<-out[is.na(new)==FALSE,]
            out<-out[order(new)]
            if (i==1) {
```

```
            out2<-out
        }
        else {
            out2<-bind_rows(out,out2)
        }
    }

    #c. Combine and build rankings
    tot_out<-out2[, .N, by=list(new)]
    out3<-as.data.table(inner_join(out2, tot_out, by=c("new"="new")))
    out3<-out3[order(-N, new, articleID)]
    out3[new!=lag(new), keyword_rank:=1]
    out3[is.na(keyword_rank)==TRUE,keyword_rank:=0]
    out3[, keyword_rank:=cumsum(keyword_rank)]
    out3[, keyword_rank:=keyword_rank+1]
    out3<-out3[order(articleID, -N, keyword_rank)]
    out3[, ord2:=1]
    out3[, ord2:=cumsum(ord2), by=list(articleID)]

    #d. Keywords will be matched back to the article data
    keywords<-reshape(out3[, .(new, articleID, keyword_rank, ord2)], idvar=c("articleID"), timevar=c("ord2"),
    direction="wide", sep="")

#2. Date / Time
    art[, pubDate_dt:=gsub("T", "", pubDate)]
    art[, pubDate_dt:=gsub("Z", "", pubDate_dt)]
    art[, pubDate_dt:=ymd_hms(pubDate_dt)]

    #Weekday
    art[, dow:=weekdays(pubDate_dt)]
    art[, wkdy:=ifelse(dow %in% c("Sunday", "Saturday"), "Weekend", "Weekend")]

    #Time of Day (Morning / Afternoon / Night)
    art[, timeofday:=substr(as.character(pubDate_dt),12,255)]
    art[, hr:=substr(timeofday,1,2)]

    art[, timeofday:=ifelse(as.numeric(hr)>=8 & as.numeric(hr)<12, "Morning",
                     ifelse(as.numeric(hr)>=12 & as.numeric(hr)<18, "Afternoon",
                     ifelse(as.numeric(hr)>=18 & as.numeric(hr)<24, "Evening",
                     ifelse(as.numeric(hr)>=0 & as.numeric(hr)<8, "Late Night", NA))))
    ]

#3. General Topics - Condense the keywords into specific generalized topics
    #(Politics, Sports, International, Entertainment, etc...)
    out3[, topic:=ifelse(grepl("Politics|Trump|Republ|Democr|Government|Election|Presidential|Senate|House of Rep|Law and
    Legislation", new)==TRUE, "1. Politics",
                  ifelse(grepl("Income|Infrastructure|Economic|Economy|Trade|Tariffs|Labor and Jobs|Wages and
                  Salaries|Real Estate|Regulation", new)==TRUE, "2. Economy",
```

```
                        ifelse(grepl("State Legislatures|States (US)|Drug Abuse and Traffic|Florida|Federal Bureau of
                        Investigation|Mueller, Robert S III|News and News Media|Firearms|Zuckerberg|Gun|Shooting|United
                        States|Social Media|Education|Colleges and
                        Universities|Discrimination|#MeToo|Blacks|Sexual|Ethnicity|Demonstration|Sex Crimes|Justice
                        Department|Executive Changes|California|States (US)|Health Insurance|Murders|Ethics", new)==TRUE, "3.
                        National",
                        ifelse(grepl("Palestinians|Terrorism|Espionage|International|Immigration|Russia|China|Iran|North
                        Korea|Kim Jong-un|Great Britain|South Korea|Syria|Olympic|Israel|Vietnam|Canada", new)==TRUE, "4.
                        International",
                        ifelse(grepl("Research|Global
                        Warming|Science|Computer|Apple|PC|Data-Mining|Database|Facebook|Analy|Environmental", new)==TRUE, "5.
                        Science and Technology",
                        ifelse(grepl("Actors and
                        Actresses|Music|Fallon|Colbert|Television|Entertainment|Theater|Theatre|Movies|Art|Books", new)==TRUE,
                        "5. Entertainment",
                        ifelse(grepl("Restaurants|Food|Cooking|Chef",new)==TRUE, "6. Food",
                        ifelse(grepl("Writing and Writers|Exercise|Women and Girls|Men and
                        Boys|Weight|Nutrition|Fashion|Parenting|Vacations|Photography|Crossword|Children|Family", new)==TRUE,
                        "7. Lifestyle",
                        ifelse(grepl("Manhattan|NYC|New York City|New York Times|New York State", new)==TRUE, "8. Local","9.
                        Other"))))))))
                        ]
    out3[, topic:=min(topic, na.rm=TRUE), by=articleID]
    gen_topic<-out3[, list(topic=max(topic)), by=list(articleID)]

#4. Keyword Rank
    kwr<-out3[, list(kwr=sum(N, na.rm=TRUE),
                     minkwr=min(keyword_rank, na.rm=TRUE)
                    ), by=articleID]

#5. Specific Topics
    out3[, minkwr:=min(keyword_rank, na.rm=TRUE), by=articleID]
    spec_topic<-out3[minkwr==keyword_rank, .(specific=new, articleID)]

#6. Article Sentiment
    art[, element_id:=1:nrow(art)]
    snippet<-get_sentences(art[, .(snippet)])
    sent1<-sentiment(snippet, neutral.nonverb.like = FALSE, missing_value=NULL)
    sent1<-as.data.table(sent1)
    sent1<-sent1[, list(snip_sent=sum(sentiment, na.rm=TRUE)), by=element_id]
    sent1[, snip_cat:=ifelse(snip_sent<= -0.50, "Very Negative",
                      ifelse(snip_sent> -0.50 & snip_sent<= -0.10, "Negative",
                      ifelse(snip_sent> -0.10 & snip_sent< 0.10, "Neutral",
                      ifelse(snip_sent>= 0.10 & snip_sent< 0.50, "Positive",
                      ifelse(snip_sent>= 0.50, "Very Positive", NA
                      )))))
    ]
```

```
    #7. Match back together
        art<-as.data.table(left_join(art, keywords[, .(articleID, kw1=new1, kwr1=keyword_rank1, kw2=new2, kwr2=keyword_rank2,
        kw3=new3, kwr3=keyword_rank3)] , by=c("articleID"="articleID")))
        art<-as.data.table(left_join(art, gen_topic, by=c("articleID"="articleID")))
        art<-as.data.table(left_join(art, spec_topic, by=c("articleID"="articleID")))
        art<-as.data.table(left_join(art, kwr, by=c("articleID"="articleID")))
        art<-as.data.table(left_join(art, sent1, by=c("element_id"="element_id")))

        saveRDS(file="./art.rds", art)

#C. Comment Features
    #1. Number of Posters by Article
        num_post<-unique(com[, .(articleID, userID)], by=c("articleID", "userID"))
        num_post<-num_post[, .N, by=articleID]
        num_post<-num_post[, .(articleID, num_posts=N)]

    #2. Comment Order
        com<-com[order(articleID, commentSequence)]
        com[, com_ord:=1]
        com[, com_ord:=cumsum(com_ord), by=articleID]

        com[, com_pos_cat:=ifelse(com_ord<=10, "Top 10",
                           ifelse(com_ord>10 & com_ord<=50, "11 to 50",
                           ifelse(com_ord>50 & com_ord<=100, "51 to 100",
                           ifelse(com_ord>100 & com_ord<=200, "101 to 200",
                           ifelse(com_ord>200, ">200", NA)))))
        ]


    #3. Time - Number of seconds from 1970-01-01
        #Convert to ts
        com[, createDate_ts:=ymd_hms(as.character(as.POSIXct(createDate, origin="1970-01-01 00:00:00")))] #Convert to datetime
        com[, approveDate_ts:=ymd_hms(as.character(as.POSIXct(approveDate, origin="1970-01-01 00:00:00")))] #Convert to
        datetime

        #Add on Article Date
        art_dt<-art[, .(pubDate_dt, articleID)]
        com<-as.data.table(left_join(com, art_dt, by=c("articleID"="articleID")))

        com[, time_to_post:=as.numeric(round(difftime(approveDate_ts, pubDate_dt, units="mins"),0))] #Time in minutes to
        posting
        com[, time_to_post:=ifelse(time_to_post<0, 0, time_to_post)] #Assumption since we do not know why posts could occur
        before publication dates
                                                              #Maybe timezone issues
        com[, time_to_post_cat:=ifelse(time_to_post>=0 & time_to_post<15, "Within 15 minutes",
                                ifelse(time_to_post>=15 & time_to_post<30, "15 to 29 minutes",
                                ifelse(time_to_post>=30 & time_to_post<45, "30 to 44 minutes",
                                ifelse(time_to_post>=45 & time_to_post<=60, "45 minutes to 1 Hour",
```

```
                                    ifelse(time_to_post>60 & time_to_post<=120, "1 to 2 Hours",
                                    ifelse(time_to_post>120 & time_to_post<=240,"2 to 4 Hours",
                                    ifelse(time_to_post>240 & time_to_post<=720,"4 to 12 Hours",
                                    ifelse(time_to_post>720, ">12 Hours",
                                    ifelse(is.na(pubDate_dt)==TRUE, "No Article", NA
                                    )))))))))
        ]

    #4. Positive / Negative comment (need text analysis)
        com[, element_id:=1:nrow(com)]
        commentBody<-get_sentences(com[, .(commentBody)])
        sent2<-sentiment(commentBody, neutral.nonverb.like = FALSE, missing_value=NULL)
        sent2<-as.data.table(sent2)
        sent2<-sent2[, list(com_sent=sum(sentiment, na.rm=TRUE)), by=element_id]

        sent2[, com_cat:=ifelse(com_sent<= -0.50, "Very Negative",
                          ifelse(com_sent> -0.50 & com_sent<= -0.10, "Negative",
                          ifelse(com_sent> -0.10 & com_sent< 0.10, "Neutral",
                          ifelse(com_sent>= 0.10 & com_sent< 0.50, "Positive",
                          ifelse(com_sent>= 0.50, "Very Positive", NA
                          )))))
        ]

    #5. Comment Length
        com[, comment_length:=str_length(commentBody)]

    #6. Readability
        read<-as.data.frame(com[, .(commentBody)])
        read<-textstat_readability(read$commentBody, measure = c("Flesch.Kincaid", "Coleman.Liau.grade"))
        read<-as.data.table(read)
        read[, element_id:=1:nrow(read)]
        read[, readFR:=round(Flesch.Kincaid,0)]
        read[, readCL:=round(Coleman.Liau.grade,0)]
        read<-read[, .(element_id, readFR, readCL)]


    #7. Match it all back together
        com<-as.data.frame(com)
        com<-as.data.table(com)
        com[, element_id:=1:nrow(com)]
        com<-left_join(com, read, by=c("element_id"="element_id"))
        com<-left_join(com, sent2, by=c("element_id"="element_id"))
        com<-as.data.table(com)

        saveRDS(file="./com.rds", com)

#D. Add Article Features to Comment Features
    art<-readRDS("./art.rds")
```

```r
    com<-readRDS("./com.rds")

    #1. Reduce Article Data
        art_red<-art[, .(type, articleID, byline, kw1, kw2, kw3, kwr1, kwr2, kwr3, timeofday, wkdy, dow,
                         topic, specific, kwr, minkwr, snip_sent, snip_cat)]

    #2. Match Article Features to Comments
        com_final<-left_join(com, art_red, by=c("articleID"="articleID", "type"="type"))
        com_final<-as.data.table(com_final)

#IV. Data Output ------------------------------------------------------------------------------
#A. Split the data into test and train sets
    com_trn<-com_final[type=="trn", ]
    com_tst<-com_final[type=="tst", ]
    com_tst[, recommendations:=NULL]  #Removing this columns as it is NA for all values and this is what we are predicting

    stopifnot(nrow(com_trn)==nrow(trn_com)) #trn_com is the original
    stopifnot(nrow(com_tst)==nrow(tst_com)) #tst_com is the original

#B. Split the Train set into Train / Test sets (original test set is for submission)
    set.seed(19790324)
    samp<-as.data.table(sample_n(com_trn, round(nrow(com_trn)*0.70,0)))
    samp<-as.data.table(samp[, .(commentID)])
    samp[, samp:="trn"]

    trn_trn<-as.data.table(left_join(com_trn, samp, by=c("commentID"="commentID")))
    trn_trn[is.na(samp)==TRUE, samp:="tst"]
    trn_tst<-trn_trn[samp=="tst"][, samp:=NULL]
    trn_trn<-trn_trn[samp=="trn"][, samp:=NULL]

    stopifnot(round(nrow(trn_trn)/(nrow(trn_trn)+nrow(trn_tst)),2)==0.70)
    stopifnot(round(nrow(trn_tst)/(nrow(trn_trn)+nrow(trn_tst)),2)==0.30)

#C. Save objects
    saveRDS(file="./tst_submission.rds", com_tst)  #Final submission set
    saveRDS(file="./trn.rds", trn_trn)             #Training set
    saveRDS(file="./tst.rds", trn_tst)             #Initial validation of the Training set to use before submission
```

```
#####################################################################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                             #
#FileName       -   001b_an_data.r                                                                  #
#By             -   Jeremy Guinta (ID 604882679)                                                    #
#                                                                                                   #
#Last Update Date:  5/11/2017                                                                       #
#                                                                                                   #
#Purpose:       -   Initial Data Exploration                                                        #
#               -   Proposed Theory - Well written,timely comments on popular articles will        #
#                   have more comment recommendations                                               #
#               -   Use features developed in 001a_an_data.r                                        #
#####################################################################################################



#I. Setup ---------------------------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid)           #Plotting utilities
    require(gridExtra)      #Plotting utilities
    require(tidyverse)      #All things tidy
    require(data.table)     #Data table is better
    require(dtplyr)         #Make sure Data table and dplyr work together
    require(ggplot2)        #Graphing Utilities
    require(stringr)        #String Functions
    require(reshape2)       #Data Reshape
    require(GGally)         #Correlation
    require(sentimentr)     #Sentiment Analysis
    require(quanteda)       #Readability Scores
    require(lubridate)

    #Set Options
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
```

```
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
                axis.text.x = element_text(size= rel(1.0)),
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                          "#999999", "#000000",  "#800000", "#B23232")

    #Custom Functions
    source("./000_mk_functions.r")  #Loads Text to Columns

#II.  Data Loading -------------------------------------------------------------------------------
#A. Load Training Set

    trn<-readRDS(file="./trn.rds")
    tst<-readRDS(file="./tst.rds")

#III. Data Processing ----------------------------------------------------------------------------

    trn1<-nrow(trn) # 465777
    trn1_u<-nrow(unique(trn[, .(articleID)])) #
    trn<-trn[is.na(pubDate_dt)==FALSE,] #Remove comments that did not match to articles
    rem1<-nrow(trn) # 448556
    rem1_u<-nrow(unique(trn[, .(articleID)])) #3407

    tst1<-nrow(tst) # 465777
    tst1_u<-nrow(unique(tst[, .(articleID)])) #
    tst<-tst[is.na(pubDate_dt)==FALSE,] #Remove comments that did not match to articles
    rem2<-nrow(tst) # 448556
    rem2_u<-nrow(unique(tst[, .(articleID)])) #3407

    orig<-trn1+tst1
    orig_u<-trn1_u+tst1_u

    rem<-rem1+rem2
    rem_u<-rem1_u+rem2_u


#IV. Data Analysis -------------------------------------------------------------------------------

#A. Build Visuals to compare recommendations to key features

    #1. Recommendations by Editor Selection
    tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE)), by=list(articleID, editorsSelection, commentID)]
    setkey(tbl, editorsSelection, recs, commentID)
    tbl[, commentID:=as.factor(commentID)]
    ord<-tbl[order(editorsSelection, -recs)][, .(commentID)]
    ord<-as.matrix(ord)
```

```
    tbl[, commentID:=factor(commentID, levels=c(ord))]

    p<-ggplot(tbl, aes(x=commentID, y=recs, fill=editorsSelection))+geom_bar(stat="identity")
    p<-p+out_theme
    p<-p+scale_fill_manual(values=color_scheme)
    p<-p+labs(title="Recommendations by Editor Selection", x="commentID", y="Recommendations")
    p<-p+theme(axis.text.x=element_blank())
    graph1<-p

    #2. Recommendations by Editor Selection (Recommendations over time)
    tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(articleID,
    editorsSelection)]
    tbl[, avg_recs:=recs/num_com]
    tbl[, ord:=1]
    tbl[, ord:=cumsum(ord), articleID]
    tbl[, maxord:=max(ord), articleID]
    tbl[maxord==2, descr:="The Article has Editor Selected Comments"]
    tbl[maxord==1, descr:="The Article does not have Editor Selected Comments"]

    setkey(tbl, editorsSelection, avg_recs, articleID)
    tbl<-tbl[order(editorsSelection, -avg_recs, articleID)]
    tbl[, articleID:=as.factor(articleID)]

    p<-ggplot(tbl, aes(x=articleID, y=avg_recs, fill=editorsSelection))+geom_bar(stat="identity")
    p<-p+out_theme
    p<-p+facet_wrap(~descr)
    p<-p+scale_fill_manual(values=color_scheme)
    p<-p+labs(title="Average Recommendations by Number of Comments and Editor Selection", subtitle="For Articles when
    Articles have at Least One Editor Selected Comments", x="articleID", y="Average Recommendations")
    p<-p+theme(axis.text.x=element_blank())
    graph2<-p

    #3. Key Word Rank by Recommendations
    tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID), kwr=max(kwr)), by=list(articleID,
    editorsSelection)]
    tbl[, avg_recs:=recs/num_com]

    p<-ggplot(tbl, aes(x=kwr, y=avg_recs, color=editorsSelection)) + geom_point(position="jitter", alpha=0.5, shape=".") +
    geom_density2d()
    p<-p+stat_density_2d(geom="raster", aes(fill=..density..), contour=FALSE, alpha=0.1, show.legend = FALSE)
    p<-p+out_theme
    p<-p+scale_color_manual(values=color_scheme)
    p<-p+scale_fill_gradient(low="white", high="grey")
    p<-p+labs(title=c("Density of Average Recommendations by Number of Comments, Keyword Rank, and Editor Selection"),
    subtitle=c("All Articles"))
    p<-p+labs(x="Key Word Rank (Lower is Better)", y="Average Recommendations")
    p<-p+theme(legend.position="bottom")
    p<-p+xlim(-100,3000)
```

```
p<-p+ylim(-100,500)
graph3<-p

#4. Topic Analysis
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(topic)]
tbl[, topic:=gsub("[^A-Za-z]", "", topic)]
tbl[, avg_recs:=recs/num_com]

setkey(tbl, avg_recs, topic)
tbl<-tbl[order(-avg_recs, topic)]
tbl[, topic:=as.factor(topic)]
ord<-tbl[order(-avg_recs)][, .(topic)]
ord<-as.matrix(ord)
tbl[, topic:=factor(topic, levels=c(ord))]

p<-ggplot(tbl[is.na(topic)==FALSE,], aes(x=topic, y=avg_recs))+geom_bar(stat="identity")
p<-p+out_theme
p<-p+scale_fill_manual(values=color_scheme)
p<-p+labs(title="Average Recommendations by Number of Comments and Topic", subtitle="All Articles", x="Topic", y="Average
Recommendations")
p<-p+theme(axis.text.x = element_text(size= rel(0.50)))
graph4<-p

#5. Specific Topic (Highest Ranking Key Word)
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(specific)]
tbl[, specific:=gsub("[^A-Za-z]", "", specific)]
tbl[, avg_recs:=recs/num_com]

setkey(tbl, avg_recs, specific)
tbl<-tbl[order(-avg_recs, specific)]
tbl[, specific:=as.factor(specific)]
ord<-tbl[order(-avg_recs)][, .(specific)]
ord<-as.matrix(ord)
tbl[, specific:=factor(specific, levels=c(ord))]

p<-ggplot(tbl[is.na(specific)==FALSE,], aes(x=specific, y=avg_recs))+geom_bar(stat="identity")
p<-p+out_theme
p<-p+scale_fill_manual(values=color_scheme)
p<-p+labs(title="Average Recommendations by Number of Comments and Specific Topics", subtitle="All Articles", x="Specific
Topic", y="Average Recommendations")
p<-p+theme(axis.text.x=element_blank())
graph5<-p

#6. Key Word Rank (Top 3)
#1
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(kw1)]
tbl[, avg_recs:=recs/num_com]
```

```
    setkey(tbl, avg_recs, kw1)
    tbl<-tbl[order(-avg_recs, kw1)]
    tbl[, kw1:=as.factor(kw1)]
    ord<-tbl[order(-avg_recs)][, .(kw1)]
    ord<-unique(as.matrix(ord))
    tbl[, kw1:=factor(kw1, levels=c(ord))]
    tbl[1:10,]

    p<-ggplot(tbl[is.na(kw1)==FALSE,], aes(x=kw1, y=avg_recs))+geom_bar(stat="identity")
    p<-p+out_theme
    p<-p+scale_fill_manual(values=color_scheme)
    p<-p+labs(title="Average Recommendations by Number of Comments and Top Key Word", subtitle="All Articles", x="Top
    Keyword", y="Average Recommendations")
    p<-p+theme(axis.text.x=element_blank())
    graph6<-p

    #2
    tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(kw2)]
    tbl[, avg_recs:=recs/num_com]

    setkey(tbl, avg_recs, kw2)
    tbl<-tbl[order(-avg_recs, kw2)]
    tbl[, kw2:=as.factor(kw2)]
    ord<-tbl[order(-avg_recs)][, .(kw2)]
    ord<-unique(as.matrix(ord))
    tbl[, kw2:=factor(kw2, levels=c(ord))]
    tbl[1:10,]

    p<-ggplot(tbl[is.na(kw2)==FALSE,], aes(x=kw2, y=avg_recs))+geom_bar(stat="identity")
    p<-p+out_theme
    p<-p+scale_fill_manual(values=color_scheme)
    p<-p+labs(title="Average Recommendations by Number of Comments and Top Key Word", subtitle="All Articles", x="2nd Highest
    Keyword", y="Average Recommendations")
    p<-p+theme(axis.text.x=element_blank())
    graph7<-p

    #3
    tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(kw3)]
    tbl[, avg_recs:=recs/num_com]

    setkey(tbl, avg_recs, kw3)
    tbl<-tbl[order(-avg_recs, kw3)]
    tbl[, kw3:=as.factor(kw3)]
    ord<-tbl[order(-avg_recs)][, .(kw3)]
    ord<-unique(as.matrix(ord))
    tbl[, kw3:=factor(kw3, levels=c(ord))]
    tbl[1:10,]
```

```
p<-ggplot(tbl[is.na(kw3)==FALSE,], aes(x=kw3, y=avg_recs))+geom_bar(stat="identity")
p<-p+out_theme
p<-p+scale_fill_manual(values=color_scheme)
p<-p+labs(title="Average Recommendations by Number of Comments and Top Key Word", subtitle="All Articles", x="3rd Highest
Keyword", y="Average Recommendations")
p<-p+theme(axis.text.x=element_blank())
graph8<-p

#7. DOW / timeofday (Heatmap)
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(timeofday, dow)]
tbl[, avg_recs:=recs/num_com]
tbl[, rescale:=scale(avg_recs)]
tbl[, dow:=factor(dow, levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))]
tbl[, timeofday:=factor(timeofday, levels=c("Morning", "Afternoon", "Evening", "Late Night"))]

p<-ggplot(tbl, aes(x=dow, y=as.factor(timeofday)))+geom_tile(aes(fill=rescale), colour="white")
p<-p+scale_fill_gradient(low="white", high="#001933")
p<-p+labs(title="Density of Average Recommendations by Number of Comments", subtitle="All Articles", x="Day of Week",
y="Time of Day")
p<-p+out_theme
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "none")
graph9<-p

#8. DOW / timeofday (Heatmap)
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(timeofday, dow)]
tbl[, rescale:=scale(recs)]
tbl[, dow:=factor(dow, levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))]
tbl[, timeofday:=factor(timeofday, levels=c("Morning", "Afternoon", "Evening", "Late Night"))]

p<-ggplot(tbl, aes(x=dow, y=as.factor(timeofday)))+geom_tile(aes(fill=rescale), colour="white")
p<-p+scale_fill_gradient(low="white", high="#001933")
p<-p+labs(title="Density of Recommendations", subtitle="All Articles", x="Day of Week",  y="Time of Day")
p<-p+out_theme
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "none")
graph10<-p

#9. Sentiment / Grade level All Recommendations
p<-ggplot(trn[readCL>0 & readCL<=16], aes(readCL, recommendations, color=com_cat))+geom_point(alpha=0.25)
p<-p+labs(title="Scatter of Grade Reading Level, Recommendations, and Comment Sentiment", subtitle="All Comments",
x="Grade Level",  y="Recommendations")
p<-p+out_theme
p<-p+scale_color_manual(values=color_scheme)
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "bottom")
graph11<-p
```

```r
#10. Sentiment / Grade level (Average Recommendations)
tbl<-trn[readCL>0 & readCL<=16, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)),
by=list(readCL, com_cat)]
tbl[, avg_recs:=recs/num_com]

p<-ggplot(tbl, aes(readCL, avg_recs, color=com_cat))+geom_point(alpha=0.25)
p<-p+labs(title="Scatter of Grade Reading Level, Avereage Recommendations by Number of Comments, and Comment Sentiment",
subtitle="All Articles", x="Grade Level",  y="Average Recommendations")
p<-p+out_theme
p<-p+scale_color_manual(values=color_scheme)
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "bottom")
graph12<-p

#11. Sentiment / Grade level (Heatmap)
tbl<-trn[readCL>0 & readCL<=16, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)),
by=list(readCL, com_cat)]
tbl[, rescale:=scale(recs)]
tbl[, readCL:=as.factor(readCL)]

p<-ggplot(tbl, aes(x=readCL, y=as.factor(com_cat)))+geom_tile(aes(fill=rescale), colour="white")
p<-p+scale_fill_gradient(low="white", high="#001933")
p<-p+labs(title="Density of Recommendations", subtitle="All Articles", x="Grade Level",  y="Comment Sentiment")
p<-p+out_theme
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "none")
graph13<-p

#12. Time to Post Cat / timeofday (Heatmap)
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(timeofday,
time_to_post_cat)]
tbl[, rescale:=scale(recs)]
tbl[, time_to_post_cat:=factor(time_to_post_cat, levels=c("Within 15 minutes", "15 to 29 minutes", "30 to 44 minutes",
"45 minutes to 1 Hour", "1 to 2 Hours", "2 to 4 Hours", "4 to 12 Hours", ">12 Hours"))]
tbl[, timeofday:=factor(timeofday, levels=c("Morning", "Afternoon", "Evening", "Late Night"))]

p<-ggplot(tbl, aes(x=time_to_post_cat, y=as.factor(timeofday)))+geom_tile(aes(fill=rescale), colour="white")
p<-p+scale_fill_gradient(low="white", high="#001933")
p<-p+labs(title="Density of Recommendations", subtitle="All Articles", x="Time to Post Category",  y="Time of Day")
p<-p+out_theme
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "none")
p<-p+theme(axis.text.x = element_text(size= rel(0.50)))
graph14<-p

#13. Time to Post Cat / timeofday (Heatmap)
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(timeofday,
time_to_post_cat)]
```

```
tbl[, avg_recs:=recs/num_com]
tbl[, rescale:=scale(avg_recs)]
tbl[, time_to_post_cat:=factor(time_to_post_cat, levels=c("Within 15 minutes", "15 to 29 minutes", "30 to 44 minutes",
"45 minutes to 1 Hour", "1 to 2 Hours", "2 to 4 Hours", "4 to 12 Hours", ">12 Hours"))]
tbl[, timeofday:=factor(timeofday, levels=c("Morning", "Afternoon", "Evening", "Late Night"))]

p<-ggplot(tbl, aes(x=time_to_post_cat, y=as.factor(timeofday)))+geom_tile(aes(fill=rescale), colour="white")
p<-p+scale_fill_gradient(low="white", high="#001933")
p<-p+labs(title="Density of Average Recommendations", subtitle="All Articles", x="Time to Post Category",  y="Time of Day")
p<-p+out_theme
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "none")
p<-p+theme(axis.text.x = element_text(size= rel(0.50)))
graph15<-p

#14. Comment Order
tbl<-trn[, list(recs=sum(recommendations, na.rm=TRUE), num_com=n_distinct(commentID)), by=list(com_pos_cat, articleID)]
tbl[, avg_recs:=recs/num_com]
tbl[, com_pos_cat:=factor(com_pos_cat, levels=c("Top 10", "11 to 50", "51 to 100", "101 to 200", ">200"))]

p<-ggplot(tbl, aes(x=com_pos_cat, y=avg_recs))+geom_boxplot()
p<-p+scale_fill_gradient(low="white", high="#001933")
p<-p+labs(title="Boxplot of Average Recommendations", subtitle="All Articles", x="Position Category", y="Average
Recommendations")
p<-p+out_theme
p<-p+theme(legend.title=element_blank())
p<-p+theme(legend.position = "none")
graph16<-p
```

```r
####################################################################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                           #
#FileName       -   002_an_model.r                                                                #
#By             -   Jeremy Guinta (ID 604882679)                                                  #
#                                                                                                 #
#Last Update Date:  5/13/2017                                                                     #
#                                                                                                 #
#Purpose:       -   Initial Modeling
#
#               -   GLM, GBM, RF                                                                  #
#               -                                                                                 #
#                   Initial modeling that uses base parameters.                                   #
#                   h2o with randomized grid searching of 4 hours per algorithm                   #
#                   use baseline best models from this process to determine likely best           #
#                   candidate model and likely candidate parameters                               #
#               -   Training / Test derived from initial data using a 70/30 split                 #
#               -   5-fold CV used on all models.                                                 #
#               -   All modeling performed on Training set, Test set used to evaluate MAE         #
#                   before prediction on final submission set.                                    #
####################################################################################################



#I. Setup -------------------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid)           #Plotting utilities
    require(gridExtra)      #Plotting utilities
    require(tidyverse)      #All things tidy
    require(data.table)     #Data table is better
    require(dtplyr)         #Make sure Data table and dplyr work together
    require(ggplot2)        #Graphing Utilities
    require(stringr)        #String Functions
    require(reshape2)       #Data Reshape
    require(GGally)         #Correlation
    require(h2o)            #Auto ML

    #Set Options
```

```
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
                axis.text.x = element_text(size= rel(1.0)),
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                              "#999999", "#000000",  "#800000", "#B23232")


#II.  Data Loading ---------------------------------------------------------------------------------
trn<-readRDS("./trn.rds")
trn<-trn[is.na(byline)==FALSE]  #These are comments that do not have article information
trn[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

tst<-readRDS("./tst.rds")
tst<-tst[is.na(byline)==FALSE]  #These are comments that do not have article information
tst[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

#trn<-rbind(trn,tst) #Recombining sets for full training

tst_sub<-readRDS("./tst_submission.rds") #True Submission Test set

#III. Data Processing ------------------------------------------------------------------------------
#A. Prepare the data for h2o

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    write.csv(file="./trn.csv", trn)
    write.csv(file="./tst.csv", tst)
    write.csv(file="./tst_sub.csv", tst_sub)

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    h2o.init(nthreads=1, min_mem_size="16G")

    #Load into h2o
    trn<-h2o.importFile("./trn.csv")
    tst<-h2o.importFile("./tst.csv")

#B. Set up Grid Search
```

```
        xnames <- names(trn[grepl("log_rec|picURL|inReplyTo|parentID|parentUserDisplayName|createDate_ts|C1|approveDate|
                                    permID|createDate|commentTitle|commentSequence|commentBody|approveDate_ts|userTitle|
                                    approveDate|element_id|type|articleID|commentID|recommendedFlag|pubDate_dt|
                                    status|sharing|updateDate|userDisplayName|userID|userLocation|
                                    userTitle|userURL|byline|recommendations|printPage|reportAbuseFlag|typeOfMaterial",
                                    names(trn))==FALSE])

        #1. Generalize Linear Models - LM
        hyper_params_glm <- list(
          alpha = c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1)
        )

        #2. Gradient Boosted Model - GBM
        hyper_params_gbm <- list(
          ntrees = c(5, 10, 25, 50, 100, 250, 500),
          max_depth = 5:25,
          min_rows = c(5, 10,30,70,100),
          learn_rate = c(.01,.03,.05,.08,.1),
          sample_rate = c(.975,.99,.995,1),
          col_sample_rate = c(.4,.7,1,1),
          col_sample_rate_per_tree = c(.7,1,1),
          nbins_cats = c(64,256,1024)
        )

        #3. Random Forest
        hyper_params_rf <- list(
          ntrees = c(5, 10, 25, 50, 100, 250, 500),
          max_depth = 5:25,
          min_rows = c(1,5,10,30,70,100),
          sample_rate = c(.975,.99,.995,1),
          col_sample_rate_per_tree = c(.7,1,1),
          nbins=c(5,10,15,20,25),
          mtries=c(-1,5,10,15),
          nbins_cats = c(64,256,1024)
        )

        #4. Search Criteria
        search_criteria <- list(
          strategy = "RandomDiscrete",
          max_runtime_secs = 28800,  #4 hours per run
          max_models = 500
        )

#C. Generate the model
    #1. Generalize Linear Models
        glm2 <- h2o.grid(algorithm = "glm",
                          x = xnames, y = "recommendations",
                          training_frame = trn,
```

```
                              hyper_params = hyper_params_glm,
                              search_criteria = search_criteria,
                              stopping_metric = "mae", stopping_tolerance = 1e-3,
                              stopping_rounds = 3,
                              seed = 1,
                              nfolds = 5, fold_assignment = "Modulo",
                              keep_cross_validation_predictions = TRUE,
                              lambda_search=TRUE
                              )

    glm2_sort <- h2o.getGrid(grid_id = glm2@grid_id, sort_by = "MAE", decreasing = FALSE)
    glm2_sort

    glm2_best <- h2o.getModel(glm2_sort@model_ids[[1]])
    summary(glm2_best)

    #Prediction
    pred_glm2 <- h2o.predict(glm2_best, newdata = tst, type = "probs")
    pref_glm2<-h2o.performance(glm2_best, newdata=tst)

    #Manual Performance
    man_pred_glm2<-as.data.table(pred_glm2)
    man_pred_glm2<-man_pred_glm2[, .(pred_recs=round(predict,0))]
    man_tst<-as.data.table(tst)
    man_tst<-man_tst[, .(recommendations)]

    man<-cbind(man_pred_glm2, man_tst)
    man[, sum(abs(pred_recs-recommendations), na.rm=TRUE)]/nrow(man) #MAE   21.91653

#2. Gradient Boosted Machine
    gbm2 <- h2o.grid(algorithm = "gbm",
                    x = xnames, y = "recommendations",
                    training_frame = trn,
                    hyper_params = hyper_params_gbm,
                    search_criteria = search_criteria,
                    stopping_metric = "MAE", stopping_tolerance = 1e-3,
                    stopping_rounds = 3,
                    seed = -1,
                    nfolds = 5, fold_assignment = "Modulo",
                    distribution = "poisson",
                    keep_cross_validation_predictions = TRUE
    )

    gbm2_sort <- h2o.getGrid(grid_id = gbm2@grid_id, sort_by = "MAE", decreasing = FALSE)
    gbm2_sort

    gbm2_best <- h2o.getModel(gbm2_sort@model_ids[[1]])
    summary(gbm2_best)
```

```
        #Prediction
        pred_gbm2 <- h2o.predict(gbm2_best, newdata = tst, type = "probs")
        pref_gbm2<-h2o.performance(gbm2_best, newdata=tst)

        #Manual Performance
        man_pred_gbm2<-as.data.table(pred_gbm2)
        man_pred_gbm2<-man_pred_gbm2[, .(pred_recs=round(predict,0))]
        man_tst<-as.data.table(tst)
        #man_tst<-man_tst[, .(recommendations)]

        man<-cbind(man_pred_gbm2, man_tst)
        man[, sum(abs(pred_recs-recommendations), na.rm=TRUE)]/nrow(man) #MAE   14.77185

    #3. Random Forest
        rf2 <- h2o.grid(algorithm = "randomForest",
                        x = xnames, y = "recommendations",
                        training_frame = tst, #using the smaller data due to RF memory issues
                        hyper_params = hyper_params_rf,
                        search_criteria = search_criteria,
                        stopping_metric = "MAE", stopping_tolerance = 1e-3,
                        stopping_rounds = 3,
                        seed = -1,
                        nfolds = 5, fold_assignment = "Modulo",
                        distribution = "poisson",
                        keep_cross_validation_predictions = TRUE
        )

        rf2_sort <- h2o.getGrid(grid_id = rf2@grid_id, sort_by = "MAE", decreasing = FALSE)
        rf2_sort

        rf2_best <- h2o.getModel(rf2_sort@model_ids[[1]])
        summary(rf2_best)

        #Prediction
        pred_rf2 <- h2o.predict(rf2_best, newdata = trn, type = "probs")
        pref_rf2<-h2o.performance(rf2_best, newdata=trn)

#IV. Output

#A. Save Models
        glm2_best_save <- h2o.saveModel(
          object = glm2_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/glm2.h2o",
          force =TRUE
        )

        gbm2_best_save <- h2o.saveModel(
```

```
          object = gbm2_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/gbm2.h2o",
          force =TRUE
        )

        rf2_best_save <- h2o.saveModel(
          object = rf2_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/rf2.h2o",
          force =TRUE
        )

        save(file="./004_model_paths.h2o", gbm2_best_save, rf2_best_save, glm2_best_save)

#B. Perform Prediction for Kaggle Submission
    tst_sub<-h2o.importFile("C:/h2o/tst_sub.csv")

    load(file="C:/h2o/004_model_paths.h2o")
    rf2_best<-h2o.loadModel(rf2_best_save)
    gbm2_best<-h2o.loadModel(gbm2_best_save)

    #RF
    sub_rf2 <- h2o.predict(rf2_best, newdata = tst_sub, type = "probs")
    sub_rf2<-as.data.table(sub_rf2)
    sub_rf2<-sub_rf2[, pred_recs:=round((predict),0)]
    sub_rf2<-sub_rf2[pred_recs<0, pred_recs:=0,]

    tst_sub_rf2<-as.data.table(tst_sub)
    tst_sub_rf2<-tst_sub_rf2[, .(commentID)]

    submission_rf2<-cbind(tst_sub_rf2, sub_rf2)
    submission_rf2[, predict:=NULL]
    submission_rf2[, commentID:=as.double(commentID)]
    submission_rf2[, commentID:=as.character(as.double(commentID))]

    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission2a.csv",
    as.data.frame(submission_rf2), row.names=FALSE)

    #GBM
    sub_gbm2 <- h2o.predict(gbm2_best, newdata = tst_sub, type = "probs")
    sub_gbm2<-as.data.table(sub_gbm2)
    sub_gbm2<-sub_gbm2[, pred_recs:=round((predict),0)]
    sub_gbm2<-sub_gbm2[pred_recs<0, pred_recs:=0,]

    tst_sub_gbm2<-as.data.table(tst_sub)
    tst_sub_gbm2<-tst_sub_gbm2[, .(commentID)]

    submission_gbm2<-cbind(tst_sub_gbm2, sub_gbm2)
    submission_gbm2[, predict:=NULL]
```

```
    submission_gbm2[, commentID:=as.double(commentID)]
    submission_gbm2[, commentID:=as.character(as.double(commentID))]

    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission2b.csv",
    as.data.frame(submission_gbm2), row.names=FALSE)    #5/18/2018 - Current Leader with ~14 MAE

    #Simple Ensemble (rf2, gbm2)
    submission_ens<-cbind(submission_gbm2[, .(commentID, gbm2=pred_recs)], submission_rf2[, .(rf2=pred_recs)])
    submission_ens[, pred_recs:=round((gbm2+rf2)/2,0)][, gbm2:=NULL][, rf2:=NULL]
    submission_ens[, commentID:=as.double(commentID)]
    submission_ens[, commentID:=as.character(as.double(commentID))]

    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission2c.csv",
    as.data.frame(submission_ens), row.names=FALSE)
```

```
##########################################################################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                                 #
#FileName       -   003_an_model.r                                                                      #
#By             -   Jeremy Guinta (ID 604882679)                                                        #
#                                                                                                       #
#Last Update Date:  5/13/2017                                                                           #
#                                                                                                       #
#               -   AutoML, NN                                                                          #
#               -                                                                                       #
#                   Initial modeling that uses base parameters.                                         #
#                   h2o with randomized grid searching of 4 hours per algorithm                         #
#                   use baseline best models from this process to determine likely best                 #
#                   candidate model and likely candidate parameters                                     #
#               - AutoML runs the following:                                                            #
#                   1) RF                                                                               #
#                   2) Deep RF                                                                          #
#                   3) GBM                                                                              #
#                   4) GLM                                                                              #
#                   5) NN                                                                               #
#                   6) Ensemble of all models                                                           #
#                   7) Ensemble of the best (based on 1-5) models                                       #
#               -   Training / Test derived from initial data using a 70/30 split                       #
#               -   5-fold CV used on all models.                                                       #
#               -   All modeling performed on Training set, Test set used to evaluate MAE               #
#                   before prediction on final submission set.                                          #
##########################################################################################################



#I. Setup -----------------------------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid)           #Plotting utilities
    require(gridExtra)      #Plotting utilities
    require(tidyverse)      #All things tidy
    require(data.table)     #Data table is better
    require(dtplyr)         #Make sure Data table and dplyr work together
    require(ggplot2)        #Graphing Utilities
```

```r
    require(stringr)        #String Functions
    require(reshape2)       #Data Reshape
    require(GGally)         #Correlation
    require(h2o)            #Auto ML

    #Set Options
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
                axis.text.x = element_text(size= rel(1.0)),
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                          "#999999", "#000000",  "#800000", "#B23232")


#II.  Data Loading ----------------------------------------------------------------------------
trn<-readRDS("./trn.rds")
trn<-trn[is.na(byline)==FALSE]  #These are comments that do not have article information
trn[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

tst<-readRDS("./tst.rds")
tst<-tst[is.na(byline)==FALSE]  #These are comments that do not have article information
tst[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

#trn<-rbind(trn,tst) #Recombining sets for full training

tst_sub<-readRDS("./tst_submission.rds") #True Submission Test set

#III. Data Processing -------------------------------------------------------------------------
#A. Prepare the data for h2o

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    write.csv(file="./trn.csv", trn)
    write.csv(file="./tst.csv", tst)
    write.csv(file="./tst_sub.csv", tst_sub)

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    h2o.init(nthreads=6, min_mem_size="16G")
```

```
    #Load into h2o
    trn<-h2o.importFile("./trn.csv")
    tst<-h2o.importFile("./tst.csv")


#B. Set up Grid Search
        xnames <- names(trn[grepl("log_rec|picURL|inReplyTo|parentID|parentUserDisplayName|createDate_ts|C1|approveDate|
                                  permID|createDate|commentTitle|commentSequence|commentBody|approveDate_ts|userTitle|
                                  approveDate|element_id|type|articleID|commentID|recommendedFlag|pubDate_dt|
                                  status|sharing|updateDate|userDisplayName|userID|userLocation|
                                  userTitle|userURL|byline|recommendations|printPage|reportAbuseFlag|typeOfMaterial",
                                  names(trn))==FALSE])

        #1.  Deep Learning - Neural Net - NN
        hyper_params_nn <- list(
          epochs=20,
          overwrite_with_best_model=FALSE,
          hidden=list(c(32,32,32),c(64,64), c(128,128,128)),
          max_w2=10,
          score_duty_cycle=0.025,
          activation=c("Rectifier","Tanh","TanhWithDropout"),
          input_dropout_ratio=c(0,0.05),
          score_validation_samples=10000,
          l1=c(.00001,.000001,.0000001),
          l2=c(.00001,.000001,.0000001),
          rho = c(.99,.975,1,0.95),
          rate=c(.005,.0005,.00005),
          rate_annealing=c(.00000001,.0000001,.000001),
          momentum_start=c(.5,.1,.01,.05,.005),
          momentum_stable=c(0.1, 0.2, 0.3, 0.4,0.5),
          momentum_ramp=c(1000000,100000)
        )

        #2. GLM/GBM/NN Search Criteria
        search_criteria <- list(
          strategy = "RandomDiscrete",
          max_runtime_secs = 28800,  #4 hours per run
          max_models = 500
        )

#C. Generate the model
    #1. AutoML
        ml2<-h2o.automl(x=xnames, y="recommendations",
                    training_frame=trn,
                    stopping_metric="MAE",
                    stopping_tolerance=1e-3,
                    stopping_rounds=3,
```

```
                seed=1,
                nfolds=5,
                max_models =500,
                exclude_algos = c("GLM"),
                max_runtime_secs = 28800  #4 hours
    )

    ml2_best <- ml2@leader

    #Prediction
    pred_ml2 <- h2o.predict(ml2_best, newdata = tst, type = "probs")
    pref_ml2<-h2o.performance(ml2_best, newdata=tst)

    #Manual Performance
    man_pred_ml2<-as.data.table(pred_ml2)
    man_pred_ml2<-man_pred_ml2[, .(pred_recs=round(predict,0))]
    man_tst<-as.data.table(tst)
    man_tst<-man_tst[, .(recommendations)]

    man<-cbind(man_pred_ml2, man_tst)
    man[, sum(abs(pred_recs-recommendations), na.rm=TRUE)]/nrow(man) #MAE   14.77185

#2. Neural Net
    nn2 <- h2o.grid(algorithm = "deeplearning",
                    x = xnames, y = "recommendations",
                    training_frame = trn,
                    hyper_params = hyper_params_nn,
                    search_criteria = search_criteria,
                    stopping_metric = "MAE", stopping_tolerance = 1e-3,
                    stopping_rounds = 3,
                    seed = 1,
                    nfolds = 5, fold_assignment = "Modulo",
                    distribution = "poisson",
                    keep_cross_validation_predictions = TRUE
    )

    nn2_sort <- h2o.getGrid(grid_id = nn2@grid_id, sort_by = "MAE", decreasing = FALSE)
    nn2_sort

    nn2_best <- h2o.getModel(nn2_sort@model_ids[[1]])
    summary(nn2_best)

    #Prediction
    pred_nn2 <- h2o.predict(nn2_best, newdata = tst, type = "probs")
    pref_nn2<-h2o.performance(nn2_best, newdata=tst)

    #Manual Performance
    man_pred_nn2<-as.data.table(pred_nn2)
```

```
        man_pred_nn2<-man_pred_nn2[, .(pred_recs=round(predict,0))]
        man_tst<-as.data.table(tst)
        man_tst<-man_tst[, .(recommendations)]

        man<-cbind(man_pred_nn2, man_tst)
        man[, sum(abs(pred_recs-recommendations), na.rm=TRUE)]/nrow(man) #MAE

#IV. Output

#A. Save Models
        nn2_best_save <- h2o.saveModel(
          object = nn2_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/nn2.h2o",
          force =TRUE
        )
        ml2_best_save <- h2o.saveModel(
          object = ml2_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/ml2.h2o",
          force =TRUE
        )

        save(file="./005_model_paths.h2o", nn2_best_save, ml2_best_save)

#B. Perform Prediction for Kaggle Submission
    tst_sub<-h2o.importFile("C:/h2o/tst_sub.csv")

    load(file="C:/h2o/005_model_paths.h2o")
    nn2_best<-h2o.loadModel(nn2_best_save)
    ml2_best<-h2o.loadModel(ml2_best_save)

    #Auto ML
    sub_ml2 <- h2o.predict(ml2_best, newdata = tst_sub, type = "probs")
    sub_ml2<-as.data.table(sub_ml2)
    sub_ml2<-sub_ml2[, pred_recs:=round((predict),0)]
    sub_ml2<-sub_ml2[pred_recs<0, pred_recs:=0,]

    tst_sub_ml2<-as.data.table(tst_sub)
    tst_sub_ml2<-tst_sub_ml2[, .(commentID)]

    submission_ml2<-cbind(tst_sub_ml2, sub_ml2)
    submission_ml2[, predict:=NULL]
    submission_ml2[, commentID:=as.double(commentID)]
    submission_ml2[, commentID:=as.character(as.double(commentID))]

    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission3a.csv",
    as.data.frame(submission_ml2), row.names=FALSE)

    #NN
```

```
sub_nn2 <- h2o.predict(nn2_best, newdata = tst_sub, type = "probs")
sub_nn2<-as.data.table(sub_nn2)
sub_nn2<-sub_nn2[, pred_recs:=round((predict),0)]
sub_nn2<-sub_nn2[pred_recs<0, pred_recs:=0,]

tst_sub_nn2<-as.data.table(tst_sub)
tst_sub_nn2<-tst_sub_nn2[, .(commentID)]

submission_nn2<-cbind(tst_sub_nn2, sub_nn2)
submission_nn2[, predict:=NULL]
submission_nn2[, commentID:=as.double(commentID)]
submission_nn2[, commentID:=as.character(as.double(commentID))]

write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission3b.csv",
as.data.frame(submission_nn2), row.names=FALSE)

#Simple Ensemble (ml2, nn2)
submission_ens<-cbind(submission_nn2[, .(commentID, nn2=pred_recs)], submission_ml2[, .(ml2=pred_recs)])
submission_ens[, pred_recs:=round((nn2+ml2)/2,0)][, nn2:=NULL][, ml2:=NULL]
submission_ens[, commentID:=as.double(commentID)]
submission_ens[, commentID:=as.character(as.double(commentID))]

write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission3c.csv",
as.data.frame(submission_ens), row.names=FALSE)
```

```r
##########################################################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                 #
#FileName       -   004_an_model.r                                                      #
#By             -   Jeremy Guinta (ID 604882679)                                        #
#                                                                                       #
#Last Update Date:  5/13/2017                                                           #
#                                                                                       #
#Purpose:       -   Final Modeling
#
#               -   GBM, RF                                                             #
#               -                                                                       #
#                   Final modeling that uses "Best" parameters.                         #
#                   h2o with randomized grid searching of 8 hours per algorithm         #
#                   uses best parameters from prior GBM and RF models to adjust the tuning #
#                   parameters for the final modeling                                   #
#               -   Training / Test derived from initial data using a 70/30 split       #
#               -   5-fold CV used on all models.                                       #
#               -   All modeling performed on Training set, Test set used to evaluate MAE #
#                   before prediction on final submission set.                          #
##########################################################################################



#I. Setup -----------------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid)           #Plotting utilities
    require(gridExtra)      #Plotting utilities
    require(tidyverse)      #All things tidy
    require(data.table)     #Data table is better
    require(dtplyr)         #Make sure Data table and dplyr work together
    require(ggplot2)        #Graphing Utilities
    require(stringr)        #String Functions
    require(reshape2)       #Data Reshape
    require(GGally)         #Correlation
    require(h2o)            #Auto ML

    #Set Options
```

```
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
                axis.text.x = element_text(size= rel(1.0)),
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                              "#999999", "#000000",  "#800000", "#B23232")


#II.  Data Loading ----------------------------------------------------------------------------------
trn<-readRDS("./trn.rds")
trn<-trn[is.na(byline)==FALSE]  #These are comments that do not have article information
trn[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

tst<-readRDS("./tst.rds")
tst<-tst[is.na(byline)==FALSE]  #These are comments that do not have article information
tst[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

#trn<-rbind(trn,tst) #Recombining sets for full training

tst_sub<-readRDS("./tst_submission.rds") #True Submission Test set

#III. Data Processing -------------------------------------------------------------------------------
#A. Prepare the data for h2o

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    write.csv(file="./trn.csv", trn)
    write.csv(file="./tst.csv", tst)
    write.csv(file="./tst_sub.csv", tst_sub)

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    h2o.init(nthreads=6, min_mem_size="24G")

    #Load into h2o
    trn<-h2o.importFile("./trn.csv")
    tst<-h2o.importFile("./tst.csv")

#B. Set up Grid Search
```

```
        xnames <- names(trn[grepl("log_rec|picURL|inReplyTo|parentID|parentUserDisplayName|createDate_ts|C1|approveDate|
                                   permID|createDate|commentTitle|commentSequence|commentBody|approveDate_ts|userTitle|
                                   approveDate|element_id|type|articleID|commentID|recommendedFlag|pubDate_dt|
                                   status|sharing|updateDate|userDisplayName|userID|userLocation|
                                   userTitle|userURL|byline|recommendations|printPage|reportAbuseFlag|typeOfMaterial",
                                   names(trn))==FALSE])

        #1. Gradient Boosted Model - GBM
        hyper_params_gbm <- list(
          ntrees = c(50,100,150,200),
          max_depth = c(10,15,20),
          min_rows = c(5,10,15,20),
          learn_rate = c(.07,.08,.09,.1,.11),
          sample_rate = c(.975,.99,.995,1),
          col_sample_rate = c(.3,.4,.5,.6,.7),
          col_sample_rate_per_tree = c(.6,.7,.8,.9,1),
          nbins_cats = c(32,64,128,256),
          learn_rate_annealing=c(0.25,0.5,0.75, 1)

        )

        #2. Random Forest
        hyper_params_rf <- list(
          ntrees = c(25,50,75,100),
          max_depth = c(10,15,20),
          min_rows = c(5,10,30,70,100),
          sample_rate = c(.975,.99,.995,1),
          col_sample_rate_per_tree = c(.6,.7,.8,.9,1),
          nbins=c(10,15,20),
          mtries=c(-1,5,10,15,20,25),
          nbins_cats = c(512,1024,1536)
        )

        #3. Search Criteria
        search_criteria <- list(
          strategy = "RandomDiscrete",
          max_runtime_secs = 28800,  #8 hours per model
          max_models = 500
        )

#C. Generate the model
    #1. Gradient Boosted Machine
        gbm3 <- h2o.grid(algorithm = "gbm",
                         x = xnames, y = "recommendations",
                         training_frame = trn,
                         hyper_params = hyper_params_gbm,
                         search_criteria = search_criteria,
                         stopping_metric = "MAE", stopping_tolerance = 1e-3,
```

```
                        stopping_rounds = 3,
                        seed = -1,
                        nfolds = 5, fold_assignment = "Modulo",
                        distribution = "poisson",
                        keep_cross_validation_predictions = TRUE
        )

        gbm3_sort <- h2o.getGrid(grid_id = gbm3@grid_id, sort_by = "MAE", decreasing = FALSE)
        gbm3_sort

        gbm3_best <- h2o.getModel(gbm3_sort@model_ids[[1]])
        summary(gbm3_best)

        #Prediction
        pred_gbm3 <- h2o.predict(gbm3_best, newdata = tst, type = "probs")
        pref_gbm3<-h2o.performance(gbm3_best, newdata=tst)

    #2. Random Forest
        rf3 <- h2o.grid(algorithm = "randomForest",
                        x = xnames, y = "recommendations",
                        training_frame = trn,
                        hyper_params = hyper_params_rf,
                        search_criteria = search_criteria,
                        stopping_metric = "MAE", stopping_tolerance = 1e-3,
                        stopping_rounds = 3,
                        seed = -1,
                        nfolds = 5, fold_assignment = "Modulo",
                        distribution = "poisson",
                        keep_cross_validation_predictions = TRUE
        )

        rf3_sort <- h2o.getGrid(grid_id = rf3@grid_id, sort_by = "MAE", decreasing = FALSE)
        rf3_sort

        rf3_best <- h2o.getModel(rf3_sort@model_ids[[1]])
        summary(rf3_best)

        #Prediction
        pred_rf3 <- h2o.predict(rf3_best, newdata = tst, type = "probs")
        pref_rf3<-h2o.performance(rf3_best, newdata=tst)

#IV. Output

#A. Save Models
        gbm3_best_save <- h2o.saveModel(
          object = gbm3_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/gbm3.h2o",
          force =TRUE
```

```
        )

        rf3_best_save <- h2o.saveModel(
          object = rf3_best,
          path = "//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/rf3.h2o",
          force =TRUE
        )
        save(file="./006_model_paths.h2o", gbm3_best_save, rf3_best_save)

#B. Perform Prediction for Kaggle Submission
    tst_sub<-h2o.importFile("C:/h2o/tst_sub.csv")

    load(file="C:/h2o/006_model_paths.h2o")
    rf3_best<-h2o.loadModel(rf3_best_save)
    gbm3_best<-h2o.loadModel(gbm3_best_save)

    #RF
    sub_rf3 <- h2o.predict(rf3_best, newdata = tst_sub, type = "probs")
    sub_rf3<-as.data.table(sub_rf3)
    sub_rf3<-sub_rf3[, pred_recs:=round((predict),0)]
    sub_rf3<-sub_rf3[pred_recs<0, pred_recs:=0,]

    tst_sub_rf3<-as.data.table(tst_sub)
    tst_sub_rf3<-tst_sub_rf3[, .(commentID)]

    submission_rf3<-cbind(tst_sub_rf3, sub_rf3)
    submission_rf3[, predict:=NULL]
    submission_rf3[, commentID:=as.double(commentID)]
    submission_rf3[, commentID:=as.character(as.double(commentID))]

    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission4a.csv",
    as.data.frame(submission_rf3), row.names=FALSE)

    #GBM
    sub_gbm3 <- h2o.predict(gbm3_best, newdata = tst_sub, type = "probs")
    sub_gbm3<-as.data.table(sub_gbm3)
    sub_gbm3<-sub_gbm3[, pred_recs:=round((predict),0)]
    sub_gbm3<-sub_gbm3[pred_recs<0, pred_recs:=0,]

    tst_sub_gbm3<-as.data.table(tst_sub)
    tst_sub_gbm3<-tst_sub_gbm3[, .(commentID)]

    submission_gbm3<-cbind(tst_sub_gbm3, sub_gbm3)
    submission_gbm3[, predict:=NULL]
    submission_gbm3[, commentID:=as.double(commentID)]
    submission_gbm3[, commentID:=as.character(as.double(commentID))]

    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission4b.csv",
```

```
as.data.frame(submission_gbm3), row.names=FALSE)

#Simple Ensemble (rf3, gbm3)
submission_ens<-cbind(submission_gbm3[, .(commentID, gbm3=pred_recs)], submission_rf3[, .(rf3=pred_recs)])
submission_ens[, pred_recs:=round((gbm3+rf3)/2,0)][, gbm3:=NULL][, rf3:=NULL]
submission_ens[, commentID:=as.double(commentID)]
submission_ens[, commentID:=as.character(as.double(commentID))]

write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission4c.csv",
as.data.frame(submission_ens), row.names=FALSE)
```

```
#####################################################################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                            #
#FileName       -   004_an_model.r                                                                 #
#By             -   Jeremy Guinta (ID 604882679)                                                   #
#                                                                                                  #
#Last Update Date:  5/13/2017                                                                      #
#                                                                                                  #
#Purpose:       -   Average Ensemble
#
#               -   Determine lowest possible validation MAE via an average                        #
#                   ensemble of all models predicts (GLM, GBM, RF, NN, autoML)                     #
#####################################################################################################



#I. Setup -------------------------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid)           #Plotting utilities
    require(gridExtra)      #Plotting utilities
    require(tidyverse)      #All things tidy
    require(data.table)     #Data table is better
    require(dtplyr)         #Make sure Data table and dplyr work together
    require(ggplot2)        #Graphing Utilities
    require(stringr)        #String Functions
    require(reshape2)       #Data Reshape
    require(GGally)         #Correlation
    require(h2o)            #Auto ML

    #Set Options
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
```

```
                axis.text.x = element_text(size= rel(1.0)),
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                            "#999999", "#000000",  "#800000", "#B23232")


#II.  Data Loading ---------------------------------------------------------------------------------
trn<-readRDS("./trn.rds")
trn<-trn[is.na(byline)==FALSE]  #These are comments that do not have article information
trn[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

tst<-readRDS("./tst.rds")
tst<-tst[is.na(byline)==FALSE]  #These are comments that do not have article information
tst[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

#trn<-rbind(trn,tst) #Recombining sets for full training

tst_sub<-readRDS("./tst_submission.rds") #True Submission Test set

#III. Data Processing -------------------------------------------------------------------------------
#A. Prepare the data for h2o

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    write.csv(file="./trn.csv", trn)
    write.csv(file="./tst.csv", tst)
    write.csv(file="./tst_sub.csv", tst_sub)

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    h2o.init(nthreads=6, min_mem_size="24G")

    #Load into h2o
    trn<-h2o.importFile("./trn.csv")
    tst<-h2o.importFile("./tst.csv")

#A. Load Models

load(file="C:/h2o/006_model_paths.h2o")
load(file="C:/h2o/005_model_paths.h2o")
load(file="C:/h2o/004_model_paths.h2o")

    rf3_best<-h2o.loadModel(rf3_best_save)
    gbm3_best<-h2o.loadModel(gbm3_best_save)
```

```r
    gbm2_best<-h2o.loadModel(gbm2_best_save)
    rf2_best<-h2o.loadModel(rf2_best_save)

    ml2_best<-h2o.loadModel(ml2_best_save)
    glm2_best<-h2o.loadModel(glm2_best_save)
    nn2_best<-h2o.loadModel(nn2_best_save)

#B. Prediction on Test
    pred_nn2 <- as.data.table(h2o.predict(nn2_best, newdata = tst, type = "probs"))
    setnames(pred_nn2, "pred_nn2")

    pred_ml2 <- as.data.table(h2o.predict(ml2_best, newdata = tst, type = "probs"))
    setnames(pred_ml2, "pred_ml2")
    pred_ml2[pred_ml2<0, pred_ml2:=0]

    pred_rf2 <- as.data.table(h2o.predict(rf2_best, newdata = tst, type = "probs"))
    setnames(pred_rf2, "pred_rf2")

    pred_gbm2 <- as.data.table(h2o.predict(gbm2_best, newdata = tst, type = "probs"))
    setnames(pred_gbm2, "pred_gbm2")

    pred_glm2 <- as.data.table(h2o.predict(glm2_best, newdata = tst, type = "probs"))
    setnames(pred_glm2, "pred_glm2")
    pred_glm2[pred_glm2<0, pred_glm2:=0]

    pred_rf3 <- as.data.table(h2o.predict(rf3_best, newdata = tst, type = "probs"))
    setnames(pred_rf3, "pred_rf3")

    pred_gbm3 <- as.data.table(h2o.predict(gbm3_best, newdata = tst, type = "probs"))
    setnames(pred_gbm3, "pred_gbm3")

    final_pred<-cbind(as.data.table(tst), pred_nn2, pred_ml2, pred_rf2, pred_gbm2, pred_glm2, pred_rf3, pred_gbm3)

    final_pred[, final_pred:=round((pred_nn2+pred_ml2+(3*pred_rf2)+(2*pred_gbm2)+(0*pred_glm2)+pred_rf3+(2*pred_gbm3))/10,0)]
    final_pred[, mean( abs(final_pred-recommendations) , na.rm=TRUE)] #12.01023

#C. Function to find minimized combination of ensembled models

    for (i in 1:10000) {

        chk<-i %% 1000
        if (chk==0) {
            print(i)
        }

        nums<-round(runif(7, 0,10),0)
        tot_nums<-sum(nums)
        add_nums<-length(nums[nums>1])
```

```
        final_pred[, final_pred:=round( ( ( (nums[1]*pred_nn2)  +
                                          (nums[2]*pred_ml2)  +
                                          (nums[3]*pred_rf2)  +
                                          (nums[4]*pred_gbm2) +
                                          (nums[5]*pred_glm2) +
                                          (nums[6]*pred_rf3)  +
                                          (nums[7]*pred_gbm3)) / (tot_nums)),0)]
        final_pred[editorsSelection==TRUE, final_pred:=final_pred]
        MAE<-final_pred[, mean( abs(final_pred-recommendations) , na.rm=TRUE)]
        MAE<-as.data.table(cbind(MAE, nums))
        val<-nrow(MAE)
        MAE[, ord:=1:val]
        MAE<-reshape(MAE, idvar=c("MAE"), timevar=c("ord"), direction="wide", sep="")
        MAE<-cbind(MAE, random_to_add)
        MAE<-cbind(MAE, tot_nums)
        if (i==1) {
            out<-MAE
        }
        else {
            out<-rbind(MAE, out)
        }
    }
    val<-nrow(out)
    out[,ord:=1:val ]
    out[min(MAE)==MAE, ]

        # MAE nums1 nums2 nums3 nums4 nums5 nums6 nums7  ord
# 1: 10.85752     1     1     7     1     0     0     3 7403

#IV. Output

#A. Perform Prediction for Kaggle Submission
    tst_sub<-h2o.importFile("C:/h2o/tst_sub.csv")

    #Full Ensemble
    pred_nn2 <- as.data.table(h2o.predict(nn2_best, newdata = tst_sub, type = "probs"))
    setnames(pred_nn2, "pred_nn2")

    pred_ml2 <- as.data.table(h2o.predict(ml2_best, newdata = tst_sub, type = "probs"))
    setnames(pred_ml2, "pred_ml2")
    pred_ml2[pred_ml2<0, pred_ml2:=0]

    pred_rf2 <- as.data.table(h2o.predict(rf2_best, newdata = tst_sub, type = "probs"))
    setnames(pred_rf2, "pred_rf2")

    pred_gbm2 <- as.data.table(h2o.predict(gbm2_best, newdata = tst_sub, type = "probs"))
    setnames(pred_gbm2, "pred_gbm2")
```

```
    pred_glm2 <- as.data.table(h2o.predict(glm2_best, newdata = tst_sub, type = "probs"))
    setnames(pred_glm2, "pred_glm2")
    pred_glm2[pred_glm2<0, pred_glm2:=0]

    pred_rf3 <- as.data.table(h2o.predict(rf3_best, newdata = tst_sub, type = "probs"))
    setnames(pred_rf3, "pred_rf3")

    pred_gbm3 <- as.data.table(h2o.predict(gbm3_best, newdata = tst_sub, type = "probs"))
    setnames(pred_gbm3, "pred_gbm3")

    final_pred<-cbind(as.data.table(tst_sub), pred_nn2, pred_ml2, pred_rf2, pred_gbm2, pred_glm2, pred_rf3, pred_gbm3)

    #Ensemble 1
    final_pred[,
    final_pred:=round((pred_nn2+pred_ml2+(3*pred_rf2)+(2*pred_gbm2)+(0*pred_glm2)+pred_rf3+(2*pred_gbm3))/10,0)]
    sub_ens<-final_pred[, .(commentID, pred_recs=final_pred)]
    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission5a.csv",
    as.data.frame(sub_ens), row.names=FALSE)

    #Ensemble 2

        # MAE nums1 nums2 nums3 nums4 nums5 nums6 nums7  ord
# 1: 10.85752     1     1     7     1     0     0     3 7403

    final_pred[, final_pred:=round(( (1*pred_nn2)
    +(1*pred_ml2)+(7*pred_rf2)+(1*pred_gbm2)+(0*pred_glm2)+(0*pred_rf3)+(3*pred_gbm3))/13,0)]
    sub_ens<-final_pred[, .(commentID, pred_recs=final_pred)]
    write.csv(file="//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/submission5b.csv",
    as.data.frame(sub_ens), row.names=FALSE)
```

```r
#####################################################################################################
#Engagement     -   UCLA MAS - STAT 412 - Final Project                                            #
#FileName       -   007_an_lime.r                                                                  #
#By             -   Jeremy Guinta (ID 604882679)                                                   #
#                                                                                                  #
#Last Update Date:  5/31/2017                                                                      #
#                                                                                                  #
#Purpose:       -   Lime Review
#
#####################################################################################################



#I. Setup ------------------------------------------------------------------------------------------

    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
    #setwd("C:/Users/jguinta/Desktop/Working/005_GradSchool/003_Course/STAT412/FINALPROJ/")
    setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid)            #Plotting utilities
    require(gridExtra)       #Plotting utilities
    require(tidyverse)       #All things tidy
    require(data.table)      #Data table is better
    require(dtplyr)          #Make sure Data table and dplyr work together
    require(ggplot2)         #Graphing Utilities
    require(stringr)         #String Functions
    require(reshape2)        #Data Reshape
    require(GGally)          #Correlation
    require(h2o)             #Auto ML
    require(lime)            #Open the black box

    #Set Options
    options(scipen=20)

    #Graphic Themes
        out_theme <- theme_bw() +
          theme(panel.grid.major=element_line(color="white"),
                text=element_text(family="ArialMT"),
                legend.position="bottom",
                plot.title = element_text(size = rel(1.0)),
                axis.text.x = element_text(size= rel(1.0)),
```

```
                axis.text.y = element_text(size= rel(1.0)))

        color_scheme <- c("#6495ED", "#C90E17", "#001933", "#691b14", "#08519c", "#778899", "#B0C4DE",
                              "#999999", "#000000",  "#800000", "#B23232")


#II.  Data Loading ----------------------------------------------------------------------------
trn<-as.data.table(readRDS("./trn.rds"))
trn<-trn[is.na(byline)==FALSE]  #These are comments that do not have article information
trn[, log_rec:=log(ifelse(recommendations==0, 1, recommendations))]

#III. Data Processing ----------------------------------------------------------------------------
#A. Prepare the data for h2o

    setwd("C:/h2o/")    #The network pathways are too long.  Setting directory to local C:/h2o
                        #All h2o objects will be saved here

    write.csv(file="./trn.csv", trn)
    h2o.init(nthreads=6, min_mem_size="24G")

    #Load into h2o
    trn<-h2o.importFile("./trn.csv")
    xnames <- names(trn[grepl("log_rec|picURL|inReplyTo|parentID|parentUserDisplayName|createDate_ts|C1|approveDate|
                                   permID|createDate|commentTitle|commentSequence|commentBody|approveDate_ts|userTitle|
                                   approveDate|element_id|type|articleID|recommendedFlag|pubDate_dt|
                                   status|sharing|updateDate|userDisplayName|userID|userLocation|
                                   userTitle|userURL|byline|printPage|reportAbuseFlag|typeOfMaterial", names(trn))==FALSE])

    trn<-as.data.frame(trn)
    trn<-trn[, c(xnames)]

    write.csv(file="./trn.csv", trn)
    trn<-h2o.importFile("./trn.csv")

#B. Perform Lime Review
    load(file="C:/h2o/006_model_paths.h2o")
    rf3_best<-h2o.loadModel(rf3_best_save)
    gbm3_best<-h2o.loadModel(gbm3_best_save)

    # Check explainer
    trn<-as.data.table(trn)
    trn[, C1:=NULL]
    trn[, permID:=NULL]
    trn[, commentID:=NULL]
    trn[, recommendations:=NULL]
    trn<-as.data.frame(trn)
    names(trn)
```

```
    explainer <- lime(as.data.frame(trn), model = gbm3_best)

    # Check explanation
    trn<-as.data.table(trn)

    #Editor Selection == TRUE
    explanation <- lime::explain(as.data.frame(trn[152341,]), explainer, n_features = 20, kernel_width = 0.5) #Actual
    recommendations: 3
    lime1<-plot_features(explanation)

    tbl1<-data.frame(max(explanation$model_intercept),
                sum(explanation$feature_weight),
                max(explanation$model_prediction),
                val=3,
                ed=TRUE
                )
    tbl

setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")
ggsave("./006_lime1.png", lime1, height=8, width=11)
```

```
---
title: "STAT 412 - Final Project - Modeling New York Times Comment Recommendations"
output:
  word_document:
    reference_docx: 000_styles.docx
---

```{r echo=FALSE}
    #Remove Objects
    rm(list=ls())

    #Clear Memory
    gc(reset=TRUE)

    #Set Working Directory
  setwd("//chi1fls02/tsp/LosAngeles/Admin/001_Users/jjg/STAT412/FINALPROJ/")

    #Package Install
    require(grid, quietly=TRUE)           #Plotting utilities
    require(gridExtra, quietly=TRUE)    #Plotting utilities
    require(tidyverse, quietly=TRUE)    #All things tidy
    require(data.table, quietly=TRUE)   #Data table is better
    require(dtplyr, quietly=TRUE)           #Make sure Data table and dplyr work together
    require(ggplot2, quietly=TRUE)      #Graphing Utilities
    require(stringr, quietly=TRUE)      #String Functions
    require(reshape2, quietly=TRUE)     #Data Reshape
    require(h2o, quietly=TRUE)            #h2o Machine Learning
    require(knitr, quietly=TRUE)      #RMD
    require(GGally, quietly=TRUE)     #Correlation plots
    require(broom, quietly=TRUE)      #Nice and Tidy
    require(MASS, quietly=TRUE)       #Regression

    #Set Options
    options(scipen=20)
    options(warn=-1)
```

```{r global_options, include=FALSE}
knitr::opts_chunk$set(fig.width=7, fig.height=5,
                      echo=FALSE, warning=FALSE, message=FALSE)
opts_knit$set(root.dir = "./")
```



```{r echo=FALSE}
source("./001b_an_data.r")
trn_art<-fread("./train_articles.csv", showProgress=FALSE, verbose=FALSE)
trn_com<-fread("./train_comments.csv", showProgress=FALSE, verbose=FALSE)
```
```

~~~~
```

# Executive Summary

Attempting to predict user reaction, in the form of recommendations, to New York Times article comments between January 2018
to May 2018 is a challenging endeavor.  The feature building and model focus on well written, timely comments made on popular
articles along with a properly tuned Gradient Boosted Machines and Random Forest model.  For the majority of the data an
accurate portrayal of the number of recommendations for any comment is feasible. However, the prediction accuracy for
comments with extreme number of recommendations is poor.

# Data Relied Upon

There are two main sources of data used for this exercise:

1. train_comments.csv
2. train_articles.csv

The train_comments.csv file contains `r nrow(trn_com)` observations. The data contains information specific to a comment made
on an article for New York Times articles published between `r as.Date(substr(as.character(min(trn$pubDate_dt,
na.rm=TRUE)),1,10))` and `r as.Date(substr(as.character(max(trn$pubDate_dt, na.rm=TRUE)),1,10))`.

The train_comments.csv looks like the following:

```{r echo=FALSE}
tbl<-as.data.table(capture.output(glimpse(trn_com)))
for (i in 1:10) {
  tbl<-tbl[, V1:=gsub("  ", " ",V1)]
}
tbl<-as.data.frame(tbl)
tbl<-text_to_columns(tbl, column="V1", delimiters=c(" "))
tbl<-as.data.table(tbl)
tbl<-tbl[, new4:=paste(new4, new5, new6, new7, new8, new9, new10, new11, new12, new13, new14, new15, new16)]
tbl<-tbl[, .(new2, new3, new4)]
tbl<-tbl[, new4:=gsub("\\<NA\\>", "", new4)]
tbl<-tbl[, new4:=gsub("NA", "", new4)]
tbl<-tbl[, new3:=gsub(">", "", new3)]
tbl<-tbl[, new3:=gsub("<", "", new3)]
tbl<-tbl[3:nrow(tbl)]
setnames(tbl, c("Variable Name", "Variable Type", "Short Description"))
tbl<-as.data.frame(tbl)
knitr::kable(tbl, format="pandoc")
```

The train_article.csv file contains `r nrow(trn_art)` observations. The data contains information specific to a comment made
on an article for New York Times articles published between `r as.Date(substr(as.character(min(trn$pubDate_dt,
na.rm=TRUE)),1,10))` and `r as.Date(substr(as.character(max(trn$pubDate_dt, na.rm=TRUE)),1,10))`.

The train_article.csv looks like the following:
~~~~

```{r echo=FALSE}
tbl<-as.data.table(capture.output(glimpse(trn_art)))
for (i in 1:10) {
  tbl<-tbl[, V1:=gsub("  ", " ",V1)]
}
tbl<-as.data.frame(tbl)
tbl<-text_to_columns(tbl, column="V1", delimiters=c(" "))
tbl<-as.data.table(tbl)
tbl<-tbl[, new4:=paste(new4, new5, new6, new7, new8, new9, new10, new11, new12, new13, new14, new15, new16)]
tbl<-tbl[, .(new2, new3, new4)]
tbl<-tbl[, new4:=gsub("\\<NA\\>", "", new4)]
tbl<-tbl[, new4:=gsub("NA", "", new4)]
tbl<-tbl[, new3:=gsub(">", "", new3)]
tbl<-tbl[, new3:=gsub("<", "", new3)]
tbl<-tbl[3:nrow(tbl)]
setnames(tbl, c("Variable Name", "Variable Type", "Short Description"))
tbl<-as.data.frame(tbl)
knitr::kable(tbl, format="pandoc")
```

As will be discussed later these datasets are used to create interesting features for the modeling process. The information from the article data is combined with the information from the comments via the articleID field.

Data Exclusions

During the combination process to link information from the train_article.csv data to the train_comments.csv data via the articleID field, `r orig_u-rem_u` articleIDs found in the train_comments.csv data were *not* found in the train_article.csv data. These articles were removed from the analysis. The total number of original train_comments.csv data was reduced from `r orig` to `r rem`.

Data Splitting

The testing dataset provided along with the training dataset was not sufficient for feature building and modeling tuning because it was the final submission data. In order to develop the model and features the original training dataset was divided using a 70/30% split into a training set and a validation set. The final training set contained `r trn1` observations. The validation set contained `r tst1`.

All feature engineering, exploratory data analyis and model tuning are performed on the training set. The validation set was held out specifically to determine model performance

Features Generation and Purpose

Theory

Generally, articles that are discussing popular topics should get more views and page hits. As a result, these articles should get more comments and more comment recommendations. Furthermore, comments that are closer to end of the article are likely to be viewed more often, and thus should receive more recommendations. Additionally, how the comment is written and the sentiment of the comment may influence the number of recommenddations. With these factors in mind, features werre developed out of the data to identify popular content with well written, timely comments.

Features

Article Features

1. *Keyword Rank* - This feature puts a numerical score on the key words extracted from the *Keyword* field contained within the train_article.csv dataset. For example, the most popular key word is `r unique(trn[minkwr==1,.(kw1)])` and the second most popular is `r unique(trn[minkwr==2,.(kw1)])`. This is not surprising as the news has been very focused on politics and the president's dealings since his election. As another example, one of the least popular keywords is `r unique(trn[minkwr==974,.(kw1)])`. The keyword rank is a composite score of all keywords tagged to the article. Therefore, if an article has many popular keywords then it will be ranked higher than an article with fewer popular keywords. The overall purpose of this feature is to identify popular articles based on the key words that were tagged to the article.

2. *Topic Analysis* - This feature puts each of the articles into a more defined bucket of categories. The keywords of each article were used to define each article into a broad topic. The following topics were defined:

```{r echo=FALSE}
tbl<-trn[is.na(topic)==FALSE, .N, topic][order(topic)][, N:=NULL]
knitr::kable(tbl, format="pandoc")
```

3. *Time and Day of the Article* - This feature puts date, time, day of week, and time of day categories on each article. The purpose of this feature is to divide up the data into groups under the assumption that articles published for example, Monday morning) may get more attention than an article published for example, late Friday night.

4. *Sentiment of the Article* - This feature puts a numerical and categorical ranking on each article based on the *Snippet* field. The purpose is to rank the articles from "Very Negative" to "Very Positive." However, due to the limited number of words of the *Snippet* field, this feature may not be useful.

Comment Features

1. *Number of Users Commenting on an Article* - This feature calculates the unique number of users based on the *userDisplayName* that comment on the article. The purpose is to identify popular articles that are being viewed on and commented with greater frequency than other articles.

2. *Rank Order of the Comments* - This feature puts a numeric ranking on each comment on the article based on the sequence order of the comment. The purpose is to divide the data into categories that rank comments closer to the top of the list separately from comments made towards the end.

3. *Time to Post* - This feature calculates the time it takes from the publish date to when the post occurs. This is similar the rank order, but differs as it is based on time. The purpose is to differentiate comments that are made close to when the article is published (and thus more likely to be seen) versus comment made hours or days later (and thus less likely to be seen).

4. *Sentiment Analysis* - This feature puts a numerical score on the sentiment of an article. The sentiment function from the sentimentr package (https://cran.r-project.org/web/packages/sentimentr/sentimentr.pdf) was used.

5. *Comment length* - This feature calculates the the string length of the comment. The purpose is to separate the data based on long comments versus shorter comments.

6. *Reading Grade Level* - This feature put a numerical grade level score on each comment (e.g. the comment was written at the 8th grade level). The textstat_readability from the quanteda package (https://cran.r-project.org/web/packages/quanteda/quanteda.pdf) was used. Flesch Kincaid and Coleman Liau scores were calculated for each comment.

Other Features

The data contained other features that were useful without any modification. The *editorsSelection*, *replyCount*, *newDesk*, and *articleWordCount* all appeared to be useful variables for analysis.

Analysis of Features

The first feature analyzed is the *editorSelection* variable. The following graphs compare the *editorSelection* variable over all comments and articles against total receommendations and average recommendations.

Graph 1 - All Recommendations by Editor Selection

```{r echo=FALSE}
graph1
```

Graph 2 - Article Average Recommendations by Editor Selection

```{r echo=FALSE}
graph2
```

Both of these graphs show that comments made on articles that have an editors selection designation receive many more recommendations than articles that do not have an editor selection. The first graph also shows that many comments never receive or receive very few recommendations. It also shows that there is extreme outliers with the number of recommendations. Certain comments have thousands of recommendations.

###Graph 3 - Keyword Rank by Recommendations and Editor Selection

The next feature is keyword rank. The below graph is a density plot comparing the average number of recommendations by article against the keyword rank. The smaller the keyword rank the better, and as this graph shows the vast majority of red (editor selected comment) have significantly higher average number of recommendations than higher value keyword ranks and non-editor selected comments.

```{r echo=FALSE}
graph3
```

```
### Graph 4 - Topic Analysis and Average Number of Recommendations

Topic analysis was also performed on the data.  This process took the keywords from each article and categorized them into
general categories.  The graph below shows that average number of recommendations by comment for each topic.

```{r echo=FALSE}
graph4
```


Graph 5 - Time of Day and Day of Week and the Average Number of Recommendations

This graph shows the density of average recommendations by the day of the week and the time of day category based on the
publishing date of the article. It is clear that certain time periods and days of weeks when an article is published can
impact the average number of recommendations that any comment under that article will receive.

```{r echo=FALSE}
graph9
```


Graph 6 - Grade Level, Sentiment Analysis and Recommendations

This graph shows that average recommendation for comments across grade level and sentiment of the comment. Although there is
no clear pattern in regards to sentiment, there is a pattern based on the reading grade level of the comment. Comments in the
8 to 12 grade reading level tend to score higher average recommendations than comments written at higher or lower grade
levels.

```{r echo=FALSE}
graph12
```


In another look at the same variables, the graph below suggests that the comment sentiment is fairly consistent across the
average number of recommendations, but the reading grade level is very important when determining the average number of
recommendations.

```{r echo=FALSE}
graph13
```


Graph 7 - Time of Day,Time to Post, and Recommendations

In this graph, there is a clear pattern that suggests that posts that are within 15 minutes of the publishing datetime tend
to get more recommendations than other posts. More importantly, comments made hours after the publishing of the article tend
to have lower average recommendations than comments made closer to the publishing date and time.

```{r echo=FALSE}
graph15
```
```

### Graph 8 - Comment Position and Recommendations

This final graph shows clearly that comments positioned towards the top of the order have higher than average recommdations than comments that are made later.

```{r echo=FALSE}
graph16
```

Each of the proceeding analyses suggests that the initial theory is correct: Well written, timely comments on popular articles are going to have more recommendations.

Machine Learning Model

```{r echo=FALSE}
tst<-readRDS("./tst.rds")
tst<-tst[is.na(byline)==FALSE]
require(h2o)
setwd("C:/h2o/")

h2o.init(nthreads=4, min_mem_size="16G")
h2o.no_progress()

load(file="C:/h2o/006_model_paths.h2o")
gbm3_best<-h2o.loadModel(gbm3_best_save)
rf3_best<-h2o.loadModel(rf3_best_save)
write.csv(file="./tst.csv", tst)
tst<-h2o.importFile("./tst.csv")
pref_gbm3<-h2o.performance(gbm3_best, newdata=tst)
pref_rf3<-h2o.performance(rf3_best, newdata=tst)
pred_gbm3<-h2o.predict(gbm3_best, newdata=tst)
pred_rf3<-h2o.predict(rf3_best, newdata=tst)
```

The purpose of the model is to predict the number of recommendations based on features of the article and the commment. Two models were proposed, tuned, and tested to minimize prediction error. A Gradient Boosted Machine (GBM) and a Random Forest (RF) model were used. h2o was employed as the front-end to the models. h2o has a series of advantages over other packages that allow for many models to be be developed at once.

1. h2o allows randomized grid searching with a wide variety of parameters.
2. h2o automatically standardized all dependent variables.
3. h2o allows for various distributions to be assumed into the model.
4. h2o automatically encodes all categorical variables.

All models were trained for four to eight hours using Mean Absolute Error (MAE) as the stopping metric. Five fold cross validation was performed on each model run as well. The best model was determined by the model that produced the lowest MAE in the cross validation process.

The following variables were modeled against the number of recommendations.

```{r, echo=FALSE}
tbl<-gbm3_best@allparameters$x
tbl<-as.data.table(tbl)
tbl<-tbl[tbl!="permID"]

tbl[tbl=="articleWordCount", descr:="Word count of the article"]
tbl[tbl=="commentType", descr:="Categorical variable describing who made the comment"]
tbl[tbl=="depth", descr:="The depth of the comment"]
tbl[tbl=="editorsSelection", descr:="Categorical variable TRUE/FALSE"]
tbl[tbl=="newDesk", descr:="The department that published the article"]
tbl[tbl=="replyCount", descr:="The number of replies a comment receives"]
tbl[tbl=="sectionName", descr:="Newspaper section"]
tbl[tbl=="timespeople", descr:="Unknown 0/1 indicator"]
tbl[tbl=="trusted", descr:="Unknown 0/1 indicator"]
tbl[tbl=="com_ord", descr:="Comment order by article"]
tbl[tbl=="com_pos_cat", descr:="Comment position category"]
tbl[tbl=="time_to_post", descr:="Time in minutes to post comment from publish date and time"]
tbl[tbl=="time_to_post_cat", descr:="Categorical variable of time to post"]
tbl[tbl=="comment_length", descr:="String lenght of entire comment"]
tbl[tbl=="readFR", descr:="Flesch Kincaid grade reading level"]
tbl[tbl=="readCL", descr:="Coleman Liau grade reading level"]
tbl[tbl=="com_sent", descr:="Comment sentiment"]
tbl[tbl=="com_cat", descr:="Categorical breakdown of comment sentiment"]
tbl[tbl=="kw1",descr:="First keyword after reorg"]
tbl[tbl=="kw2",descr:="Second keyword after reorg"]
tbl[tbl=="kw3",descr:="Third keyword after reorg"]
tbl[tbl=="kwr1", descr:="First keyword after reorg ranking"]
tbl[tbl=="kwr2", descr:="Second keyword after reorg ranking"]
tbl[tbl=="kwr3", descr:="Third keyword after reorg ranking"]
tbl[tbl=="timeofday", descr:="Categorical variable for the time of day"]
tbl[tbl=="dow", descr:="Day of the week"]
tbl[tbl=="topic", descr:="General topic based on keywords"]
tbl[tbl=="specific", descr:="Specific topic based on keywords"]
tbl[tbl=="kwr", descr:="Composite Keyword ranking for the article"]
tbl[tbl=="minkwr", descr:="Lowest ranking keyword for the article"]
tbl[tbl=="snip_sent", descr:="Article snippet sentiment"]
tbl[tbl=="snip_cat", descr:="Categorical breakdown of article sentiment"]
setnames(tbl, c("Variable", "Description"))

knitr::kable(tbl, format="pandoc")
```

Gradient Boosted Machine

The selected GBM model has the following parameters.

```{r echo=FALSE}
```

```
tbl<-data.frame(
    gbm3_best@allparameters$ntrees
,   gbm3_best@allparameters$max_depth
,   gbm3_best@allparameters$min_rows
,   gbm3_best@allparameters$nbins_cats
,   gbm3_best@allparameters$nbins
,   gbm3_best@allparameters$stopping_metric
,   gbm3_best@allparameters$distribution
,   gbm3_best@allparameters$sample_rate
,   gbm3_best@allparameters$col_sample_rate
,   gbm3_best@allparameters$col_sample_rate_per_tree
,   gbm3_best@allparameters$learn_rate
,   gbm3_best@allparameters$learn_rate_annealing
)
tbl$id<-1
tbl<-melt(tbl, id.vars="id")
tbl<-as.data.table(tbl)
tbl[, id:=NULL]
tbl[, variable:=gsub("gbm3_best\\.allparameters\\.", "", variable)]
knitr::kable(tbl)
```

The GBM model with the best cross validated MAE was selected as the "best" model. For this model the overall cross validated training MAE is:

```{r echo=FALSE}
tbl<-gbm3_best@model$cross_validation_metrics_summary[1,]
tbl<-as.data.table(tbl)
tbl[, ord:=1]
tbl<-melt(tbl, id.vars="ord")
tbl<-tbl[variable!="sd"][, ord:=NULL]
tbl[, variable:=gsub("_valid", "", variable)]
tbl[, variable:=gsub("_", " ", variable)]
tbl[, value:=round(as.numeric(value),2)]
setnames(tbl, c("MAE Category","MAE"))
knitr::kable(tbl, format="pandoc")
```

Using this model and predicting recommendations and then calculating the MAE on the validation set results in a validation MAE of `r round(pref_gbm3@metrics$mae,2)`.

This model used many variables. The variable importance metrics that GBM creates can be used to detemine if any of the developed features are important.

```{r echo=FALSE}
tbl<-as.data.table(head(h2o.varimp(gbm3_best),21))
tbl<-tbl[variable!="permID"]
ord<-as.matrix(unique(tbl[order(scaled_importance)][, .(variable)]))
```

tbl[, variable:=factor(variable, levels=c(ord))]
p<-ggplot(tbl, aes(variable, scaled_importance))+geom_bar(stat="identity")
p<-p+out_theme
p<-p+labs(title="Top 20 Variable Importance: GBM", x="Variable", y="Scaled Importance")
p<-p+coord_flip()
p
```

The developed keyword rank (*kwr*) feature ranked third in importance after existing features *replyCount* and *editorSelection*. Other developed features *time_to_post_cat*, *com_ord*, *dow*, and *comment_length* all ranked in the the top ten. Surprisingly, *com_sent* and *readFR* ranked in the bottom half of the top 20 importance variarbles.

Random Forest

The selected RF model has the following parameters.

```{r echo=FALSE}
tbl<-data.frame(
    rf3_best@allparameters$ntrees
,   rf3_best@allparameters$max_depth
,   rf3_best@allparameters$min_rows
,   rf3_best@allparameters$nbins_cats
,   rf3_best@allparameters$nbins
,   rf3_best@allparameters$stopping_metric
,   rf3_best@allparameters$distribution
,   rf3_best@allparameters$sample_rate
,   rf3_best@allparameters$col_sample_rate_per_tree
)

tbl$id<-1
tbl<-melt(tbl, id.vars="id")
tbl<-as.data.table(tbl)
tbl[, id:=NULL]
tbl[, variable:=gsub("rf3_best\\.allparameters\\.", "", variable)]
knitr::kable(tbl)
```

The RF model with the best cross validated MAE was selected as the "best" model. For this model the overall cross validated training MAE is:

```{r echo=FALSE}
tbl<-rf3_best@model$cross_validation_metrics_summary[1,]
tbl<-as.data.table(tbl)
tbl[, ord:=1]
tbl<-melt(tbl, id.vars="ord")
tbl<-tbl[variable!="sd"][, ord:=NULL]
tbl[, variable:=gsub("_valid", "", variable)]
tbl[, variable:=gsub("_", " ", variable)]

tbl[, value:=round(as.numeric(value),2)]
setnames(tbl, c("MAE Category","MAE"))
knitr::kable(tbl, format="pandoc")
```

Using this model and predicting recommendations and then calculating the MAE on the validation set results in a validation MAE of `r round(pref_rf3@metrics$mae,2)`.

This model used many variables. The variable importance metrics that RF creates can be used to detemine if any of the developed features are important.

```{r echo=FALSE}
tbl<-as.data.table(head(h2o.varimp(rf3_best),21))
tbl<-tbl[variable!="permID"]
ord<-as.matrix(unique(tbl[order(scaled_importance)][, .(variable)]))
tbl[, variable:=factor(variable, levels=c(ord))]
p<-ggplot(tbl, aes(variable, scaled_importance))+geom_bar(stat="identity")
p<-p+out_theme
p<-p+labs(title="Top 20 Variable Importance: RF", x="Variable", y="Scaled Importance")
p<-p+coord_flip()
p
```

Other than *replyCount* and *editorsSelection* ranking very high, the RF importance plot shows dramatically different variables as being important. *kw2* and *kw3* rank much higher in the RF model than in the GBM model. Other developed features *time_to_post*, *time_to_post_cat*, *com_ord*, and *dow* all ranked in the top ten. Surprisingly, *com_sent* and *readFR* do not rank in the top 20 and *kwr* ranks towards the bottom of the top 20.

##Model Error

GBM

```{r echo=FALSE}
pred_tst<-cbind(as.data.table(tst),as.data.table(pred_gbm3))
pred_tst[, num_cat:=ifelse(recommendations<10, "Less than 10", ifelse(recommendations>=10 & recommendations<20, "Between 10 and 19", ifelse(recommendations>=20 & recommendations<50, "Between 20 and 49", ifelse(recoplmmendations>=50 & recommendations<100, "Between 50 and 99", ifelse(recommendations>=100 & recommendations<200, "Between 100 and 199", ifelse(recommendations>=200, "Greater than or equal to 200",NA))))))]
pred_tst[is.na(num_cat)==TRUE, num_cat:="Less than 10"]
pred_tst[is.na(recommendations)==TRUE, recommendations:=0]

tbl<-pred_tst[is.na(num_cat)==FALSE, list(MAE=mean(abs(predict-recommendations)), AE=sum(abs(predict-recommendations)), total_comments=.N), by=list(num_cat)]
tbl[, num_cat:=factor(num_cat, levels=c("Less than 10", "Between 10 and 19", "Between 20 and 49", "Between 50 and 99", "Between 100 and 199", "Greater than or equal to 200"))]
tbl<-tbl[order(num_cat)]
tbl[, tot:=sum(total_comments)][, pct:=total_comments/tot][, tot:=NULL]
tbl[, tot:=sum(AE)][, pct_AE:=AE/tot][, tot:=NULL]

```

The majority of the error in the GBM model is driven by comments with extreme numbers of recommendations. Using the validation set the number of comments are categorized and then the MAE is calculated based on the number of comments category. The results below show that error is being driven by a relatively small percentage of comment (as a percentage of all comments analyzed). The `r tbl[6,.(num_cat)]` category makes up `r 100*round(tbl[6, .(pct)],3)` % of the data, but `r 100*round(tbl[6, .(pct_AE)],3)` % of the absolute error from the model.

```{r echo=FALSE}
tbl[, pct:=paste(as.character(100*round(pct,3)), "%", sep="")]
tbl[, pct_AE:=paste(as.character(100*round(pct_AE,3)), "%", sep="")]
setnames(tbl, c("Category", "MAE", "AE", "# of Comments", "% of Comments", "% of Error"))
knitr::kable(tbl, format="pandoc")
```

RF

```{r echo=FALSE}
pred_tst<-cbind(as.data.table(tst),as.data.table(pred_rf3))
pred_tst[, num_cat:=ifelse(recommendations<10, "Less than 10", ifelse(recommendations>=10 & recommendations<20, "Between 10 and 19", ifelse(recommendations>=20 & recommendations<50, "Between 20 and 49", ifelse(recommendations>=50 & recommendations<100, "Between 50 and 99", ifelse(recommendations>=100 & recommendations<200, "Between 100 and 199", ifelse(recommendations>=200, "Greater than or equal to 200",NA))))))]
pred_tst[is.na(num_cat)==TRUE, num_cat:="Less than 10"]
pred_tst[is.na(recommendations)==TRUE, recommendations:=0]

tbl<-pred_tst[is.na(num_cat)==FALSE, list(MAE=mean(abs(predict-recommendations)), AE=sum(abs(predict-recommendations)), total_comments=.N), by=list(num_cat)]
tbl[, num_cat:=factor(num_cat, levels=c("Less than 10", "Between 10 and 19", "Between 20 and 49", "Between 50 and 99", "Between 100 and 199", "Greater than or equal to 200"))]
tbl<-tbl[order(num_cat)]
tbl[, tot:=sum(total_comments)][, pct:=total_comments/tot][, tot:=NULL]
tbl[, tot:=sum(AE)][, pct_AE:=AE/tot][, tot:=NULL]
```

Just like the GBM model, The majority of the error in the RF model is driven by comments with extreme numbers of recommendations. Using the validation set the number of comments are categorized and then the MAE is calculated based on the number of comments category. The results below show that error is being driven by a relatively small percentage of comment (as a percentage of all comments analyzed). The `r tbl[6,.(num_cat)]` category makes up `r 100*round(tbl[6, .(pct)],3)`% of the data, but `r 100*round(tbl[6, .(pct_AE)],3)`% of the absolute error from the model.

```{r echo=FALSE}
tbl[, pct:=paste(as.character(100*round(pct,3)), "%", sep="")]
tbl[, pct_AE:=paste(as.character(100*round(pct_AE,3)), "%", sep="")]
setnames(tbl, c("Category", "MAE", "AE", "# of Comments", "% of Comments", "% of Error"))
knitr::kable(tbl, format="pandoc")
```
```

# Summary and Conclusion

The two models performed well in terms of the training and validation MAE, and several developed features demonstrate high relative importance in each of the models conducted. However, both models performed poorly when predicting the number of recommendations for comments when the actual value of recommendations is extremely high. The Kaggle scored MAE for both models was relatively close to the training and validation MAE. This suggests that the models are not overfit.

The final submitted Kaggle competition models are:

1. GBM model. This model had a public Kaggle score of 14.59.
2. A simple ensemble of the GBM and the RF models. This model had a public Kaggle score of 14.95.