News Popularity Prediction

STAT 844/CM 764 Winter 2020

Donya Hamzeian 20852145

Introduction

Recently, predicting news popularity has become a hot topic in machine learning. Media companies like Mashable may be willing to tackle this problem for the following reasons:

1. This prediction is important for Mashable in order to know whether a candidate article will achieve the desired amount of popularity i.e will be popular or not. If this predicted popularity is below a threshold, Mashable may decide not to post this article.
2. Mashable can sell advertisements based on the predicted value of popularity.
3. In addition to the prediction aspect, this problem is also interesting from the inference aspect i.e. by the end of this project one can infer which predictors are more associated with the popularity of an article and what are their relationships.

Researchers have approached this problem either as a binary classification problem- i.e. predicting whether an article will hit a certain amount of popularity(1) or not(0)- or as a regression problem. The approach of this report is regression, and predicting popularity as a numeric value is the target of this report.

Data

The data used in this project belongs to Mashable Inc.[1] and was first pre-processed and made available by Fernandes et al [2] in 2015. The data is available here: http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity In this poject we are going to give a machine learning model to predict the number of times that an article has been shared using the attributes of the articles. This data contains 48 attributes of 39644 articles posted in Mashable. This data overall contains 49 columns, including target value and 2 non-predictive variables. The information about the columns is as follows:

1. url: URL of the article, which is non-predictive (char)
2. timedelta: Days between the article publication and the dataset acquisition, which is non-predictive (min:8 , max: 731)
3. nTokensTitle: Number of words in the title(min:2, max: 23)
4. nTokensContent: Number of words in the content(min:0, max:8474)
5. nUniqueTokens: Rate of unique words in the content(min:0, max: 701)
6. nNonStopWords: Rate of non-stop words in the content(min:0, max: 1042)
7. nNonStopUniqueTokens: Rate of unique non-stop words in the content(min:0, max: 650)

8. numHrefs: Number of links(min:0, max: 304)
9. numSelfHrefs: Number of links to other articles published by Mashable(min:0, max: 116)
10. numImgs: Number of images(min:0, max: 128)
11. numVideos: Number of videos(min:0, max: 91)
12. averageTokenLength: Average length of the words in the content(min:0, max: 8.042)
13. numKeywords: Number of keywords in the metadata(min:1, max:10)
14. channel: data channel(a categorical vaiable with 7 categories)
15. kwMinMin: Worst keyword (min. shares)(min:-1, max:377)
16. kwMaxMin: Worst keyword (max. shares)(min:0, max:298400)
17. kwAvgMin: Worst keyword (avg. shares)(min:-1, max:42827.9)
18. kwMinMax: Best keyword (min. shares)(min:0, max:843300)
19. kwMaxMax: Best keyword (max. shares)(min:0, max:843300)
20. kwAvgMax: Best keyword (avg. shares)(min:0, max:843300)
21. kwMinAvg: Avg. keyword (min. shares)(min:-1, max:3613)
22. kwMaxAvg: Avg. keyword (max. shares)(min:0, max:298400)
23. kwAvgAvg: Avg. keyword (avg. shares)(min:0, max:43568)
24. selfReferenceMinShares: Min. shares of referenced articles in Mashable(min:0, max:843300)
25. selfReferenceMaxShares: Max. shares of referenced articles in Mashable(min:0, max:843300)
26. selfReferenceAvgSharess: Avg. shares of referenced articles in Mashable(min:0, max:843300)
27. weekday: The day that the article was published on(A categorical variable with 7 categories)
28. isWeekend: Was the article published on the weekend?(Dummy variable)
29. LDA00: Closeness to LDA topic 0(min:0, max: 0.92699)
30. LDA01: Closeness to LDA topic 1(min:0, max: 0.92595)
31. LDA02: Closeness to LDA topic 2(min:0, max: 0.92000)
32. LDA03: Closeness to LDA topic 3(min:0, max: 0.92653)
33. LDA04: Closeness to LDA topic 4(min:0, max: 0.92719)
34. globalSubjectivity: Text subjectivity(min:0, max: 1.0000)
35. globalSentimentPolarity: Text sentiment polarity(min:-0.39375, max: 0.72784)
36. globalRatePositiveWords: Rate of positive words in the content(min:0, max: 0.15549)
37. globalRateNegativeWords: Rate of negative words in the content(min:0, max: 0.184932)
38. ratePositiveWords: Rate of positive words among non-neutral tokens(min:0, max: 1)
39. rateNegativeWords: Rate of negative words among non-neutral tokens(min:0, max: 1)
40. avgPositivePolarity: Avg. polarity of positive words(min:0, max: 1)

41. minPositivePolarity: Min. polarity of positive words(min:0, max: 1)
42. maxPositivePolarity: Max. polarity of positive words(min:0, max: 1)
43. avgNegativePolarity: Avg. polarity of negative words(min:-1, max: 0)
44. minNegativePolarity: Min. polarity of negative words(min:-1, max: 0)
45. maxNegativePolarity: Max. polarity of negative words(min:-1, max: 0)
46. titleSubjectivity: Title subjectivity(min:0, max: 1)
47. titleSentimentPolarity: Title polarity(min:-1, max: 1)
48. absTitleSubjectivity: Absolute subjectivity level(min:0, max: 0.5)
49. absTitleSentimentPolarity: Absolute polarity level(min:0, max: 1)

Some notes about the columns

In the following, we will provide some additional information about the above columns:

1. Fernandes et al. [2] describe how they have built the columns of 15-23. They first ranked the keywords of an article based on the number of their shares- in other articles that were published before this article. The keyword with the highest number of shares was categorized as the "best keyword", the one with the lowest was categorized as the "worst keyword", and the last one was categorized as the "average keyword". Then, for each keyword they calculated the "average", "maximum", and "minimum" number of shares of the other articles.
2. Fernandes et al. [2] describe how they have built the columns of 29-33. They applied Latent Dirichlet Allocation algorithm to find the top 5 topics and then calculated how close each article is to each of these 5 topics.
3. It is strange that the columns 5, 6, and 7 that are 'rate' by definition, have maximum values of 701, 1042, and 650 respectively. This is related to the row 31038 and it is certainly an outlier because after removing this row, the maximum values will be 1 which is sensible.

Preprocessing

Although the data is known to be a clean data with no missing values, doing some data exploratory analyses reveals that some columns of the data are highly skewed or contain outliers. Therefore, outlier removal techniques, transformations, feature reduction, and other methods were used in this article to prepare the data for model building step. The most highly correlated columns with `shares` was listed in section A.1 and there you can see that the most correlated column is `kw_avg_avg` which has the correlation of 0.11041286 ; however, after preprocessing steps, you can see that more correlated variables will appear, meaning that we can expect them to have a significant role in predicting our target value.

In the following, the necessity for the preprocessing methods was described.

1. Feature reduction

All codes for this section can be found in A.1.

- `is_weekend` was removed because its information is already in the `weekday` column. If weekday is Saturday or Sunday, `is_weekday` will be TRUE. Therefore, it is a redundant column.

- `timedelta` and `url` were removed because they are non-predictive features.
- `abs_title_sentiment_polarity` column is the absolute value of `title_sentiment_polarity` column. Therefore, the information in `abs_title_sentiment_polarity` is already in `title_sentiment_polarity` and only one of them should be kept, so `abs_title_sentiment_polarity` was removed.
- Similarly, `abs_title_subjectivity` is the absolute value of `title_subjectivity` column, and `abs_title_subjectivity` was removed.
- Sum of LDA columns is 1, so there is collinearity between them and LDA_04 was removed.

2. Transformation of the target variable

- As you can see in figure 1, the `shares` variable is highly skewed; however, log of this column is an approximately normal variable. Therefore, the log transformation is a sensible preprocessing for this column.(The code for this section and figure 1 is in A.2)

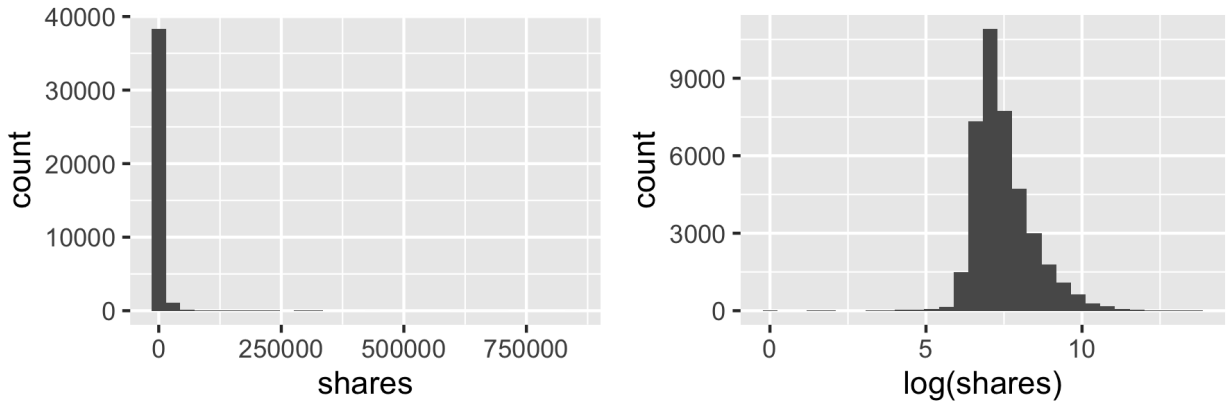

Figure 1: The histogram of shares vs the histogram of the log of shares

3. Grouping

- I) There are 1181 instances that there is no information about their content, although they have a content; this can be verified by going to their webpage by using their url. These instances have many all-zero columns including `n_tokens_content`, `n_unique_tokens`, `n_non_stop_words`, `n_non_stop_unique_tokens`, `num_hrefs`, `num_self_hrefs`, `average_token_length`, `self_reference_min_share` `self_reference_max_shares`, `self_reference_avg_sharess`, `global_subjectivity`, `global_sentiment_polarity`, `global_rate_positive_words`, `global_rate_negative_words`, `rate_negative_words`, `rate_positive_words`, `avg_positive_polarity`, `avg_negative_polarity`, `min_positive_polarity`, `min_negative_polarity`, `max_positive_polarity`, `max_negative_polarity`. As the zeros in these columns indicate missing values, not that their value is really zero, it may be better to treat this data differently. This is also proved by doing hypothesis testing on the shares of these two groups of data. Therefore, we extracted this data and removed the all-zero columns and fit a different ML model to this piece of data. We call the extracted data `no_content` and the remaining data `no_zero`. (The code and test results for this part is provided in A.3.I)
- II) Additionally, there are 5993 instances which lack the information about self reference shares, i.e. the columns `self_reference_min_shares`, `self_reference_max_shares`, and `self_reference_avg_sharess` are all-zeroes which is strange since being zero for them means that these instances do not have self reference, so this is indicating that this piece of the data is

different. This difference was also proved by doing hypothesis testing on the shares of these two groups of data. We call the extracted data `no_shares` and the remaining data `no_zero`.(The code and test results for this part is provided in A.3.b)

Preprocessing for no_zero data

The codes for this part is in section A.4.I and the related graphs are stored in the image folder.

- The `n_non_stop_words` is all ones and was removed
- self reference columns: As you can see in figure 2, the self reference columns, i.e. `self_reference_min_shares`, `self_reference_max_shares`, and `self_reference_avg_shares` are highly skewed and the log transform make them more like a normal variable.

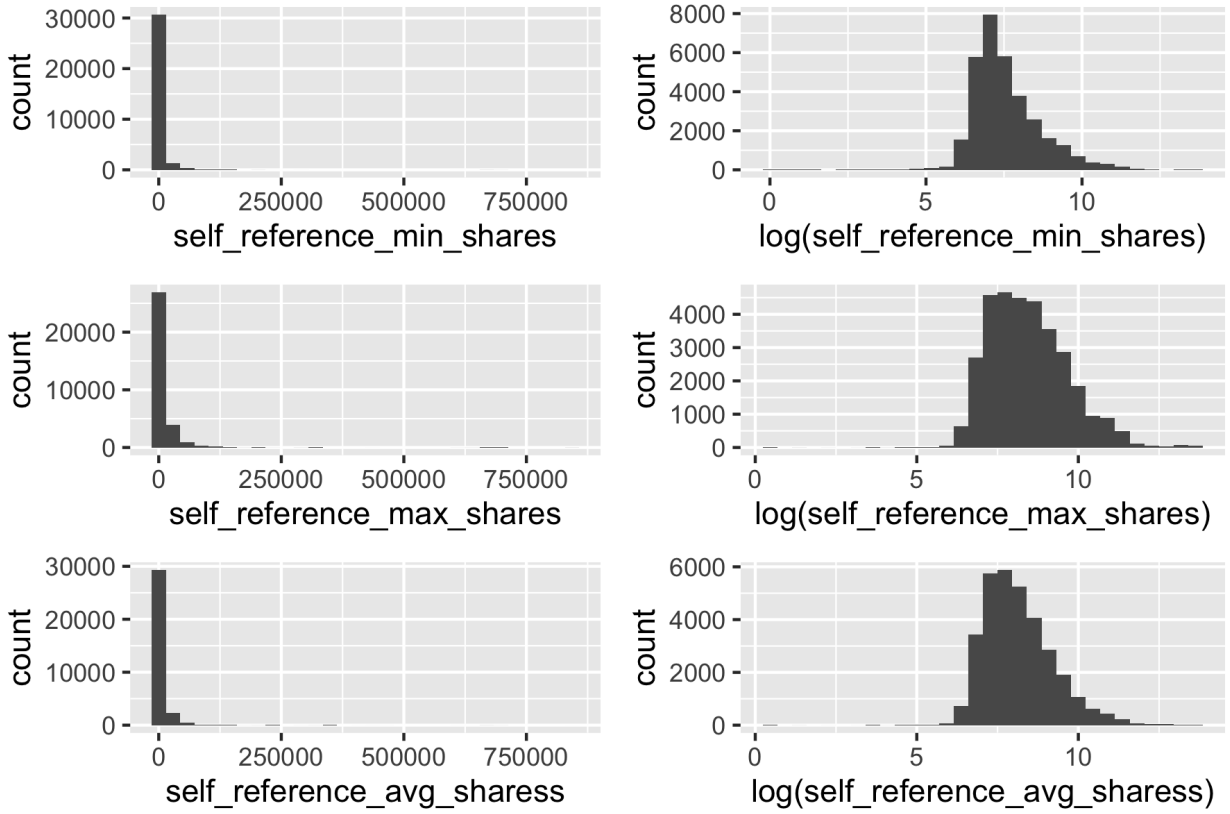

Figure 2: Histograms of self reference shares columns vs their log transform in no zero data

- `n_tokens_content`: As you can see in figure 3, `n_tokens_content` is a highly skewed variable and the log transform make it more like a normal variable.

- `num_hrefs`: Figure 4, shows the similar result for the `num_href` column.

- `num_self_hrefs`: As shown in figure 5, log transform does not solve the problem. Therefore, the outliers (the values higher than 30) were replaced by 30

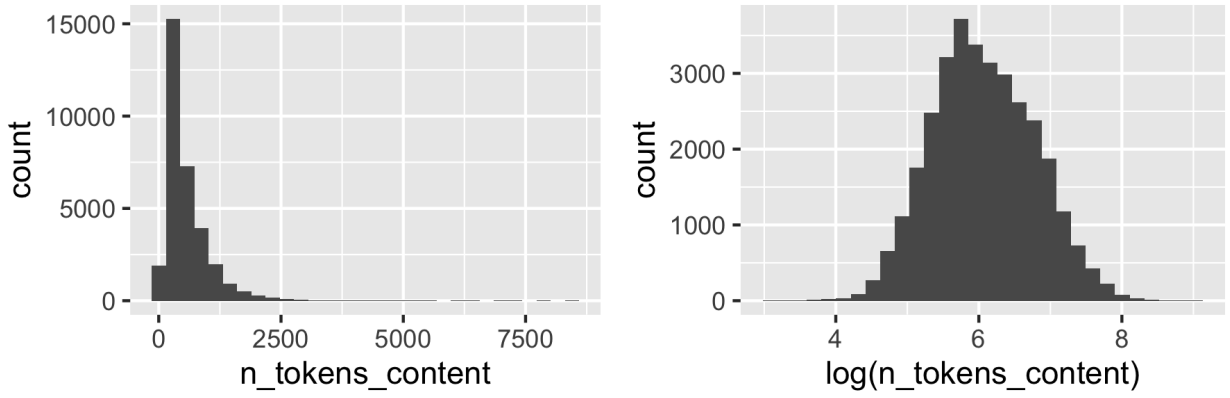

Figure 3: The histogram of n tokens content vs the log transform of this column in no zero data, height = 1

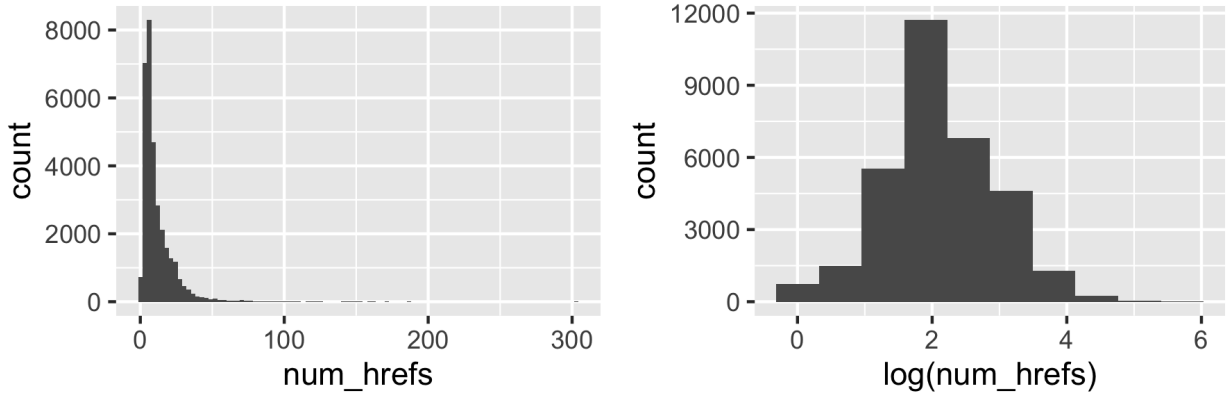

Figure 4: The histogram of num hrefs vs the log transform of this column in no zero data

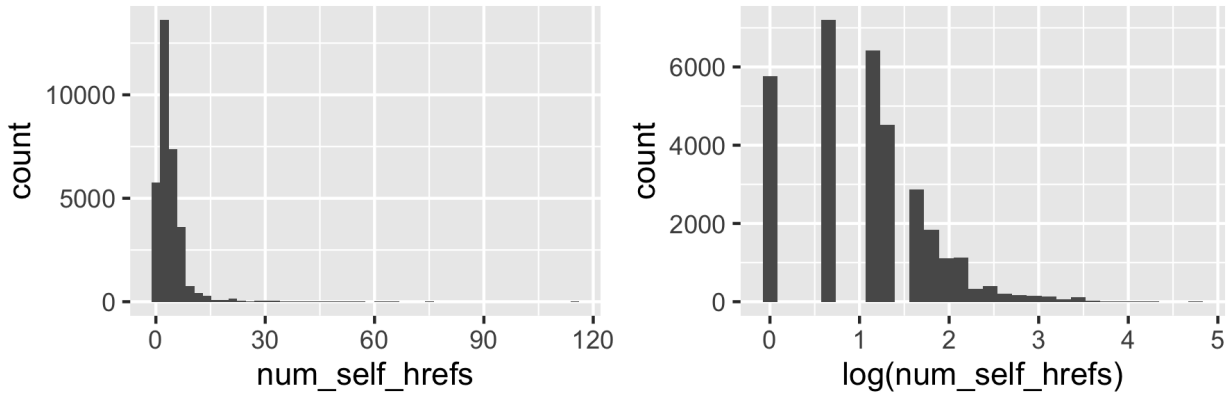

Figure 5: The histogram of num self hrefs vs the log transform of this column in no zero data

- `num_imgs` and `num_videos`: As shown in figure 6, there are many outliers in these columns. Therefore, the outliers in `num_imgs` (the values higher than 25) were replaced by 25, and the outliers in `num_videos` (the values higher than 6) were replaced by 6.

- `kw_min_min`: As you can see in figure 7, the `kw_min_min` is highly unbalanced, near 20000 of instances

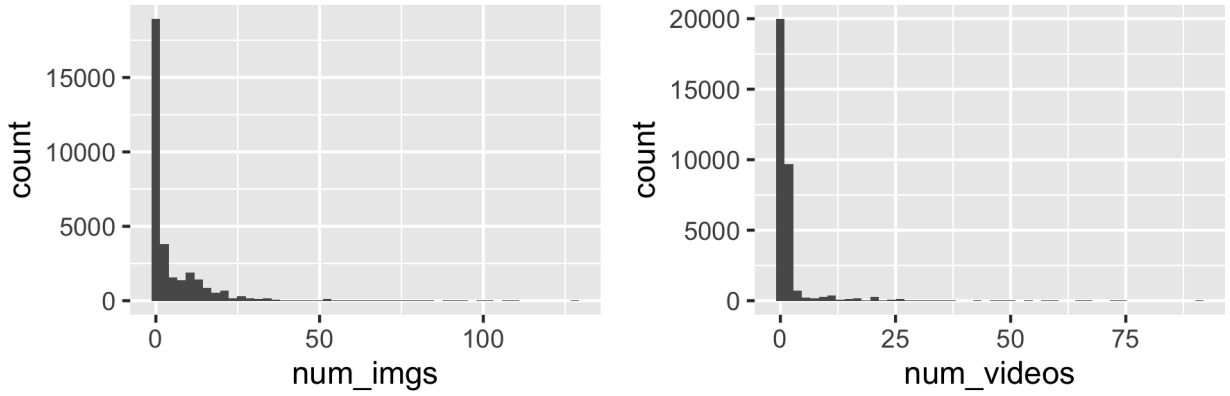

Figure 6: The histogram of num imgs and num videos in no zero data

take value of -1, which indicates that this data is missing since this column is the minimum number of shares for the worst keyword and cannot be -1. Therefore, we can colnclude that -1 is an indicator of missing value, so we binned this column into 2 groups and treated them as an indicator variable. The ones with value -1 are mapped into 0 and the others are mapped into 1.

- `kw_min_max` and `kw_max_max`: As you can see in figure 7, these columns were highly unbalanced and were not significant factors based on the p.values in linear model fitted to the data, so they were removed.
- `kw_avg_min` As you can see in figure 7, this column has around 500 values equal to -1 which is again an indicator of missing values. After replacing these missing values with the average of non-missing values, the transformation log(x+1) was done on this column to make it more normal
- `kw_min_avg` The preprocessing on this column is exactly like `kw_avg_min`.
- `kw_avg_avg`, `kw_max_avg`,`kw_max_min`: As you can see in figure 7, these columns are highly skewed and log(x+1) transformation was applied on them to make them more like a normal variable.

You can see the histogram of kw columns after the transformation explained above in figure `imgs/kw_after_no_zero.png` stored in imgs folder.

Thw unchanged columns

The following columns were unchanged and in the following images you can see their histogram and verify that there was no need to change them.

- `n_tokens_title`, `n_unique_tokens`, and `n_non_stop_unique_tokens`: See figure 8.
- `average_token_length`: See figure 9.
- `num_keywords`: See figure 10
- `LDA_00`, `LDA_01`, `LDA_02`, and `LDA_03`: See figure 11
- `kw_avg_max`: See figure 7

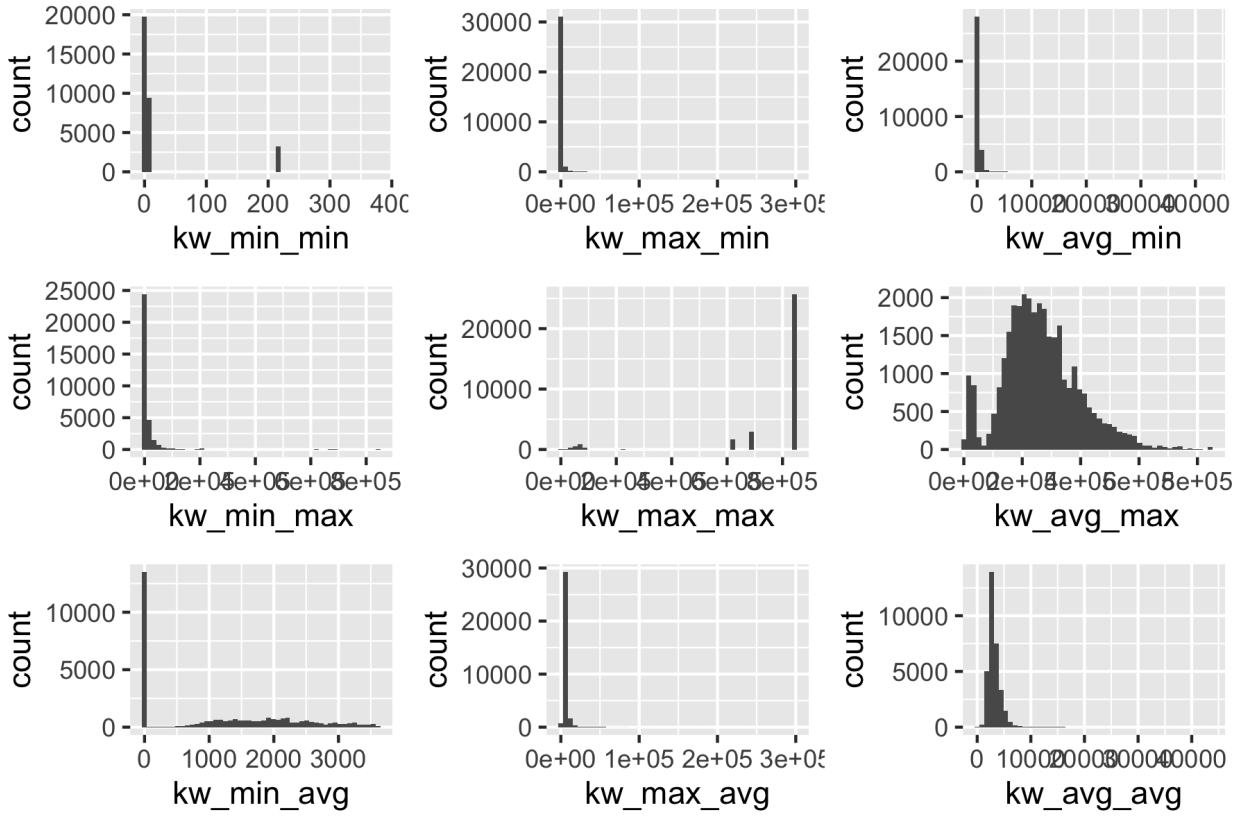

Figure 7: The histogram of kw columns in no zero data before transformation

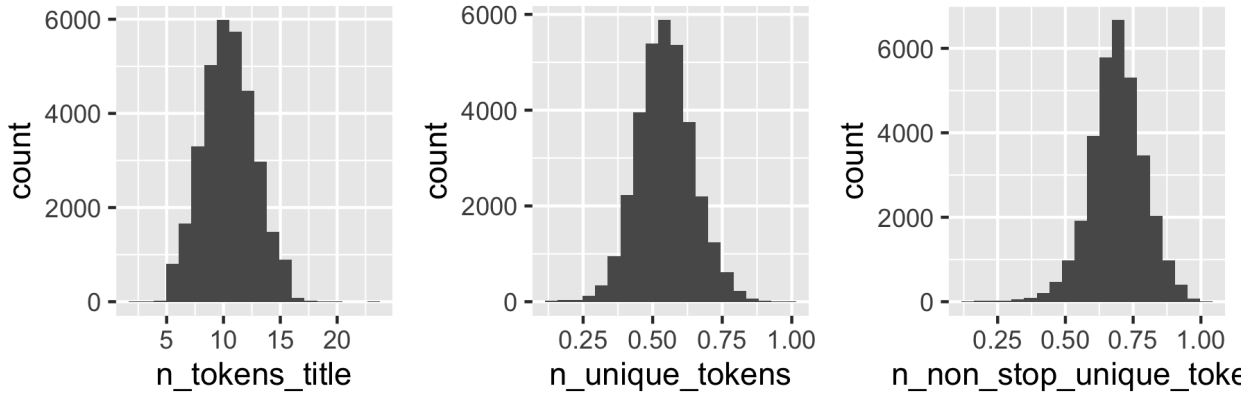

Figure 8: The histogram number of tokens columns in no zero data

- `avg_positive_polarity`, `max_positive_polarity`, `min_positive_polarity`, `avg_negative_polarity`, `max_negative_polarity`, and `min_negative_polarity`: See figure 12

- `global_sentiment_polarity`: See figure 13

- `title_sentiment_polarity`: This column has many values around zero, but rosnerTest identifies no outlier. See figure 14

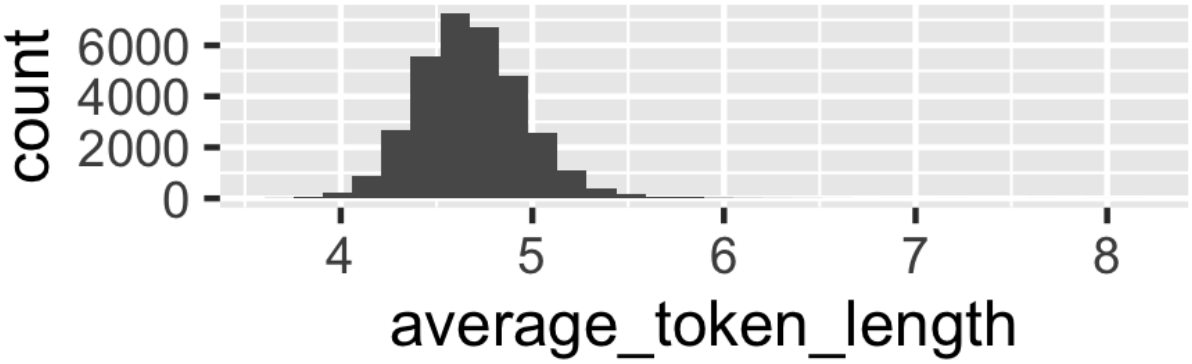

Figure 9: The histogram of average token length column in no zero data

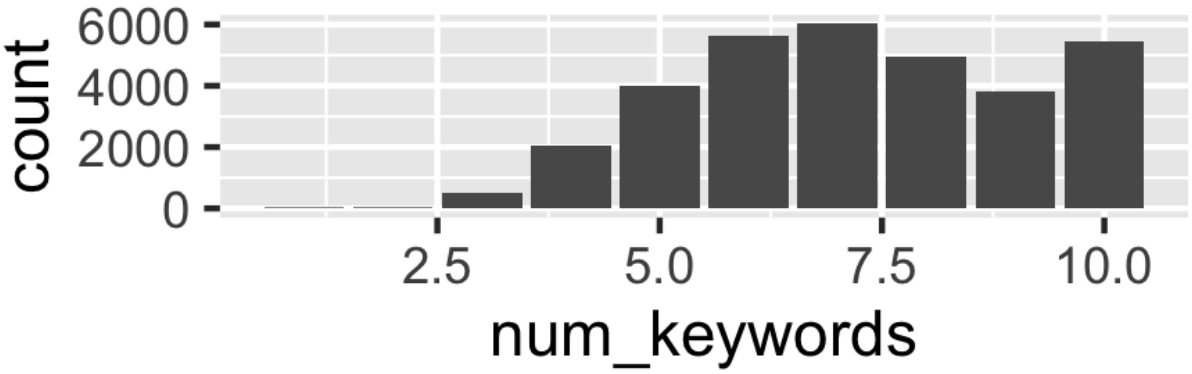

Figure 10: The histogram of num keywords in no zero data

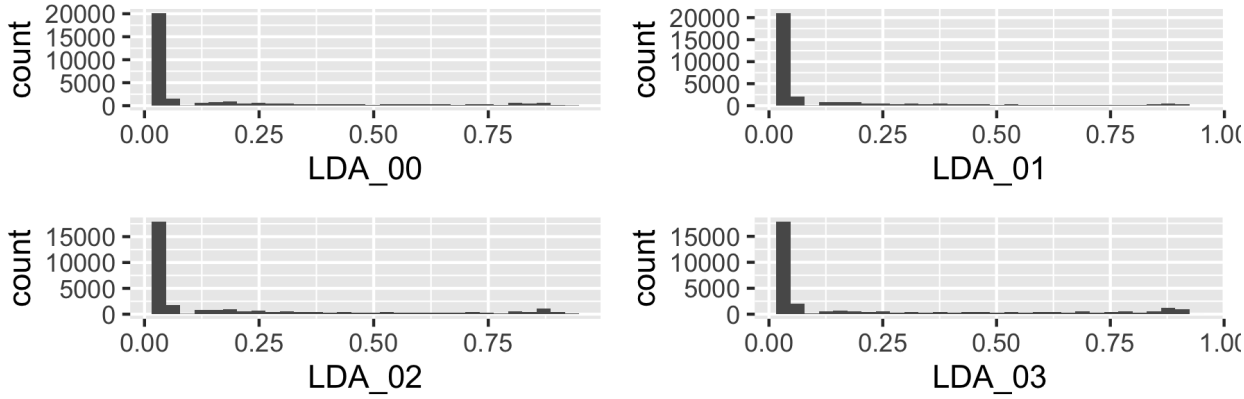

Figure 11: The histogram of LDA columns in no zero data

- `global_rate_positive_words` , `global_rate_negative_words`, `rate_positive_words`, and `rate_negative_words`: See figure 15

- `global_subjectivity` and `title_subjectivity`: `imgs/subjectivity_no_zero.png`. The

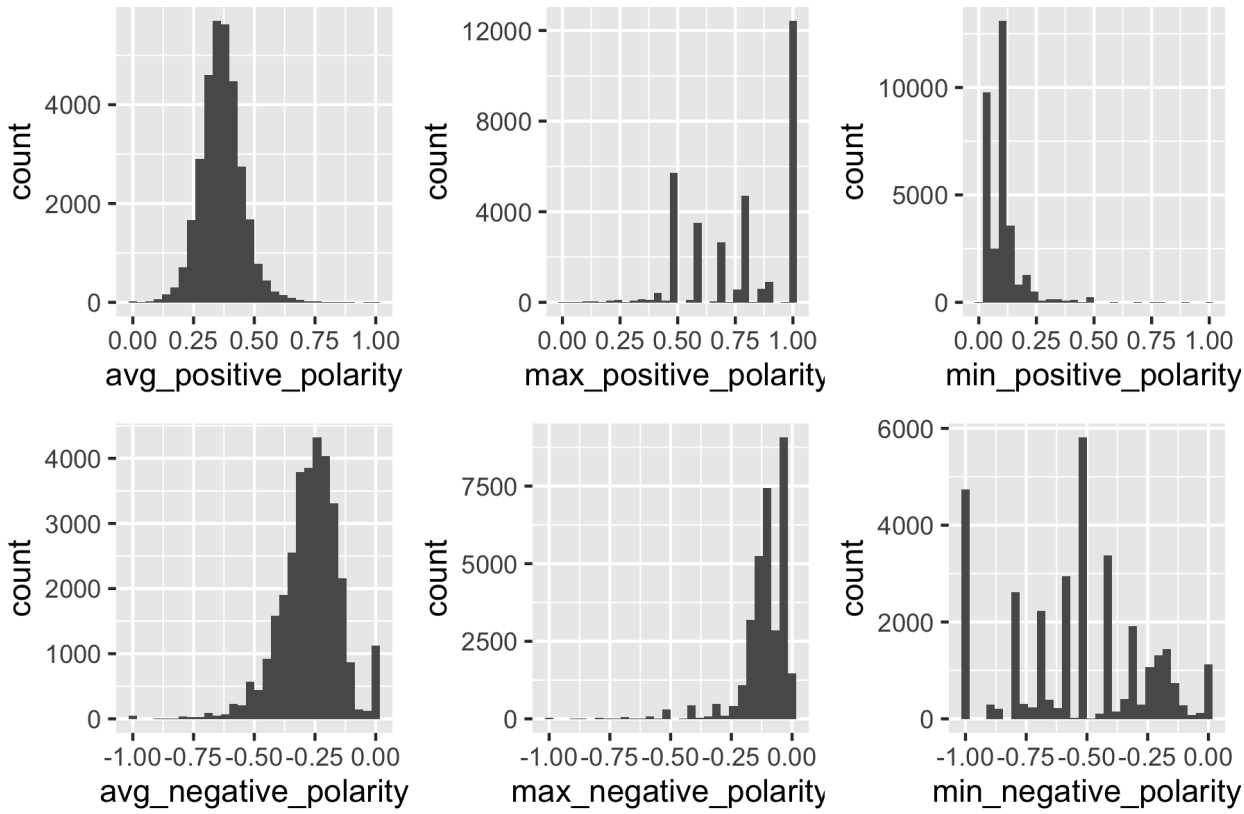

Figure 12: The histogram of polarity columns in no zero data

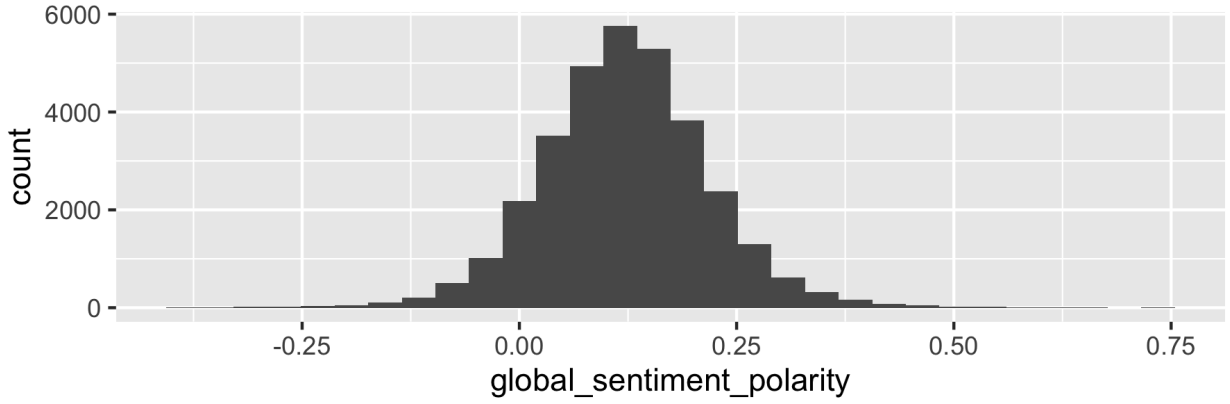

Figure 13: The histogram of global polarity columns in no zero data

`title_subjectivity` column has many values around zero, but rosnerTest identifies no outlier. See figure 16

Categorical variables:

No preprocessing was done on the categorical variables, but you can see the bar plot of `weekday` and `channel` columns in figure 17. It seems that the articles from miscellaneus channel are being shared more and also the articles posted on weekends are shared more.

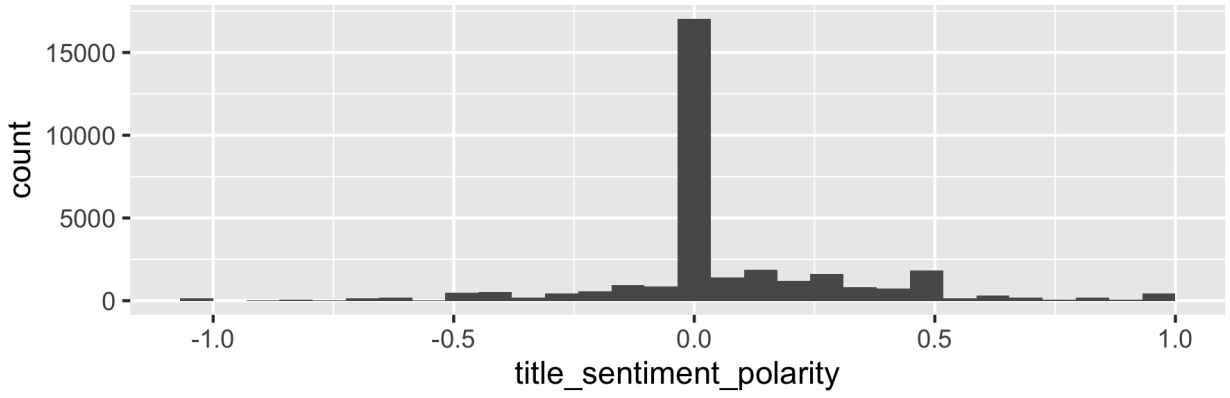

Figure 14: The histogram of title polarity columns in no zero data

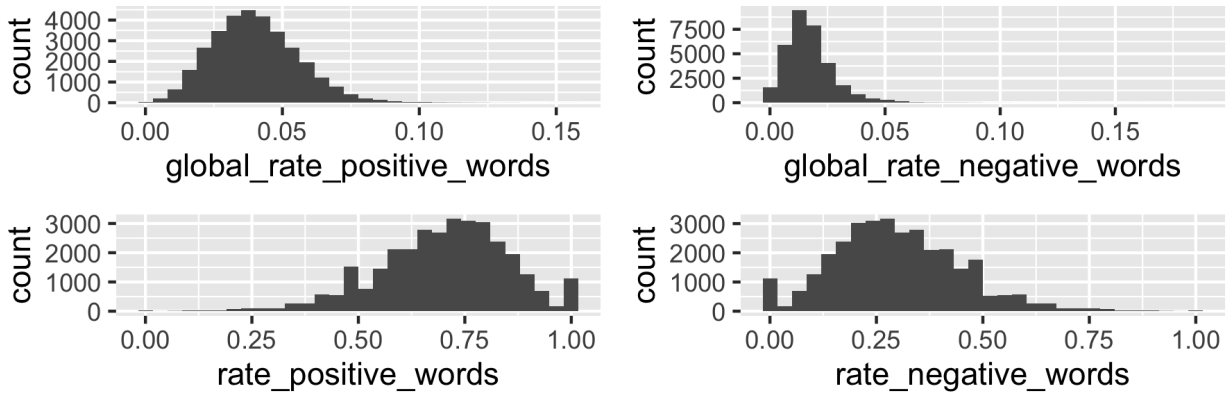

Figure 15: The histogram of rate columns in no zero data

Preprocessing for no_shares data

The codes for this part is in section A.4.II and the related graphs are stored in the image folder. The figures are quite similar to the previous section, so for further details please see the image folder.

- The `n_non_stop_words` is all ones and was removed
- `n_tokens_content` Similar to the `no_zero` data, log transformation was also done for `no_shares` data.(See `imgs/n_tokens_content_no_shares.png` and Figure 18 in the appendix for the code)
- `num_hrefs` This is also a highly skewed variable, and had contained some zeroes. Therefore, log(x+1) was used to get a more normal variable. (see `imgs/num_href_no_shares.png` and Figure 19 in the appendix for the code)
- `num_self_hrefs` Similar to `no_zero` data, log transformation was not helpful. Therefore, the outliers (the values higher than 5) were replaced by 5 (see `imgs/num_shref_no_shares.png` and Figure 20 in the appendix for the code)
- `num_imgs` and `num_videos`: Similar to `no_zero` data, the values higher than 7 in `num_imgs` were replaced by 7, and the outliers in `num_videos` (the values higher than 5) were replaced by 5.(see `imgs/img_videos_no_shares.png` and Figure 21 in the appendix for the code)

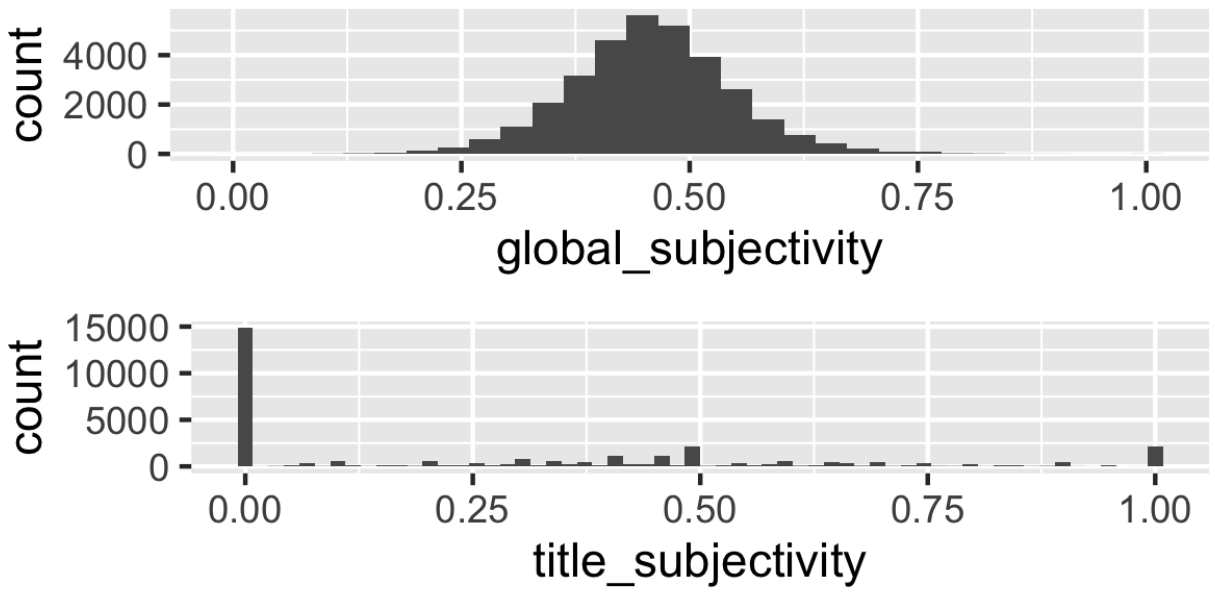

Figure 16: The histogram of subjectivity columns in no zero data

Figure 17: weekday and channel barplot in no zero data

- As you can see in the figure stored in `imgs\kw_before_no_shares.png`(See Figure 22 in the appendix for the code), `kw_min_min` is a highly unbalanced data, so we binned this column into 3 categories: -1 to group1 , 0-4 to group2, and >4 to group3. Similar to `no_zero` data, -1 in `kw_avg_min`implies missing data so it was imputed by average of other values, then applied the log(x+1) transform. `kw_max_min`, `kw_min_max`, and `kw_max_max` columns were removed again because the preliminary linear model on `no_shares` data resulted in these variables being non-significant. Finally, `kw_min_avg` and `kw_max_avg` were transformed by log(x+1) to get a more normal variable. You can see the histogram of these columns after transformation in figure stored in `"imgs/kw_after_no_shares.png"` (See Figure 23 in

the appendix for the code)

Thw unchanged columns

The following columns were unchanged and in the following images you can see their histogram and verify that there was no need to change them.

- `n_tokens_title`, `n_unique_tokens`, and `n_non_stop_unique_tokens`: `imgs/n_tokens_unchanged_no_shares.png`(S Figure 24 in the appendix for the code)
- `average_token_length`:`imgs/avg_token_no_shares.png`(See Figure 25 in the appendix for the code)
- `num_keywords`: `imgs/num_kw_no_share.png`(See Figure 26 in the appendix for the code)
- `LDA_00`, `LDA_01`, `LDA_02`, and `LDA_03`: `imgs/LDA_no_shares.png` (See Figure 27 in the appendix for the code)
- `kw_avg_max`, `kw_avg_avg`: `imgs/kw_before_no_shares.png` (See Figure 22 in the appendix for the code)
- `avg_positive_polarity`, `max_positive_polarity`, `min_positive_polarity`, `avg_negative_polarity`, `max_negative_polarity`, and `min_negative_polarity`: `imgs/polarity_no_shares.png` (See Figure 28 in the appendix for the code)
- `global_sentiment_polarity`: `imgs/global_polarity_no_shares.png` (See Figure 29 in the appendix for the code)
- `title_sentiment_polarity`: `imgs/title_polarity_no_shares.png`. This column has many values around zero, but rosnerTest identifies no outlier. (See Figure 30 in the appendix for the code)
- `global_rate_positive_words` , `global_rate_negative_words`, `rate_positive_words`, and `rate_negative_words`: `imgs/rate_no_shares.png` (See Figure 31 in the appendix for the code)
- `global_subjectivity` and `title_subjectivity`: `imgs/subjectivity_no_shares.png`. The `title_subjectivity` column has many values around zero, but rosnerTest identifies no outlier. (See Figure 32 in the appendix for the code)

Categorical variables

No preprocessing was done on the categorical variables, but you can see the bar plot of `weekday` and `channel` columns in `imgs/weekday_channel_no_shares.png`. (See Figure 33 in the appendix for the code)

Preprocessing for no_content data

The codes for this part is in section A.4.III and the related graphs are stored in the image folder.

- `num_imgs` and `num_videos`: Similar to `no_zero` data, the values higher than 12 were replaced by 12 in `num_imgs`, and the outliers in `num_videos` (the values higher than 5) were replaced by 2.(see `imgs/img_videos_no_content.png`. See Figure 34 in the appendix for the code)
- As you can see in the figure stored in `imgs/kw_before_no_content.png`(See Figure 35 in the appendix for the code), `kw_min_min` is a highly unbalanced column and most of the values are -1 which indicate missing value, so we removed it.`kw_max_max` was also removed because it was highly unbalanced. Additionally, log(x+1) transformation was applied on columns `kw_max_min` and `kw_min_max` to result in a more normal variable. `kw_avg_min` column contained some missing values(-1) which was imputed by the average of the others and then log(x+1) tranformation was applied on this column. Finally, `kw_min_avg` was binned into 2 groups: 0 and non-zero and was treated as a categorical variable.

You can see the histogram of these columns after transformation in figure stored in `imgs/kw_after_no_content.png` (See Figure 36 in the appendix for the code)

The unchanged columns

The following columns were unchanged and in the following images you can see their histogram and verify that there was no need to change them.

*`n_tokens_title` and `num_keywords`: `imgs/n_tokens_unchanged.png` (See Figure 37 in the appendix for the code)

- `LDA_00`,`LDA_01`, `LDA_02`, `LDA_03`: `imgs/LDA_no_content.png`(See Figure 38 in the appendix for the code)
- `title_sentiment_polarity` and `title_subjectivity`: `imgs/title_no_content.png` (See Figure 39 in the appendix for the code)
- `kw_avg_max`, `kw_avg_avg`, and `kw_max_avg`: `imgs/kw_before_no_content.png` (See Figure 35 in the appendix for the code)

Categorical variables

No preprocessing was done on the categorical variables, but you can see the bar plot of `weekday` and `channel` columns in `imgs/weekday_channel_no_content.png`(See Figure 40 in the appendix for the code). It can be seen in the picture that the articles from lifestyle channel are shared more. Also, it seems that the articles are shared more in weekends.

Statistical Analysis

The codes and more technical details for this section are in A.5 In order to predict `shares`, or log of shares more specifically, we used 3 statistical models. Random Forest, Boosting, and Lasso. The first two methods are tree-based methods and were expected to behave similarly, but the third one is more similar to the linear regression and was expected to give different predictions. All model building processes were done separately for each parts of the data, i.e. `no_zero`, `no_content`, and `no_shares` datasets. The train-test split is the first step in building a statistical model. In order to do so, 75% of each dataset was used as to train the model on that dataset. Each of the above models have hyperparameters that need to be tuned, and cross-validation was used to tune them on each of the datasets. Then, each of the tuned models were fitted on the train part of the datasets. Finally, the trained models were used to predict the log of shares in the test datasets. The goodness of predictions for each model and for each dataset were measured by the following measure of errors: RMSE and MAE. Note that RMSE on the log of shares is equivalant to RMSLE on the shares.

Below you can see the RMSE and MAE of each of the trained(and tuned) models on the test partition of `no_zero`, `no_shares`, and `no_content` dataset.

I) Results for `no_zero` data:

Error	RandomForest	Boosting	Lasso
RMSE	0.84996	0.8429697	0.8661247
MAE	0.6336508	0.6194361	0.6447485

As you can see in the above table, boosting has the lowest RMSE and MAE among the three models.

The 10 most important variables in predicting shares based on the random forest and boosting models for the `no_zero` data is as follows.

Priority	RandomForest	Boosting
1	self_reference_avg_sharess	kw_avg_avg
2	self_reference_min_shares	self_reference_avg_sharess
3	kw_avg_avg	kw_max_avg
4	channel	self_reference_min_shares
5	kw_max_avg	num_hrefs
6	LDA_02	kw_avg_max
7	num_hrefs	kw_min_avg
8	self_reference_max_shares	LDA_02
9	weekday	global_subjectivity
10	num_imgs	n_tokens_content

`kw_avg_avg`, `self_reference_avg_sharess`, `self_reference_min_shares`, `kw_max_avg`, `num_hrefs`, and `LDA_02` are among the 10 most important features in both models.

We cannot understand the direction of effect of each of the important variables from random forest or boosting models, but lasso can give us some insight about their effects. Below, you can see the variables with positive and negative coefficients from the lasso model on `no_zero` data. The variables that are not here were unimportant and have zero coefficient,

Table 3: Negative Coefficients

negative
n_unique_tokens
n_non_stop_unique_tokens
num_self_hrefs
kw_min_min1
LDA_01
LDA_02
min_positive_polarity
avg_negative_polarity
min_negative_polarity
channelentertainment
weekdayThursday
weekdayTuesday
weekdayWednesday

Table 4: Positive Coefficients

positive
num_hrefs
num_imgs
num_videos
num_keywords
kw_min_min0
kw_max_min
kw_max_avg
kw_avg_avg

positive
self_reference_min_shares
self_reference_avg_sharess
LDA_00
global_subjectivity
title_subjectivity
title_sentiment_polarity
channelmiscellaneous
channelsocmed
channeltech
weekdayMonday
weekdaySaturday
weekdaySunday

The variables with non zero coefficients are consistent with important variables from random forest and boosting methods.

II) Results for `no_shares` data:

Error	RandomForest	Boosting	Lasso
RMSE	0.8524058	0.8488679	0.8830334
MAE	0.6199316	0.6080459	0.6320053

As you can see in the above table, boosting has the lowest RMSE and MAE among the three models.

The 10 most important variables in predicting shares based on the random forest and boosting models for the `no_shares` data is as follows.

Priority	RandomForest	Boosting
1	LDA_02	kw_avg_avg
2	kw_avg_avg	LDA_02
3	channel	kw_max_avg
4	weekday	LDA_00
5	LDA_00	global_subjectivity
6	kw_max_avg	kw_avg_max
7	num_hrefs	n_unique_tokens
8	global_subjectivity	kw_avg_min
9	n_tokens_title	n_tokens_content
10	n_unique_tokens	LDA_01

`LDA_02`, `kw_avg_avg`, `kw_max_avg`, `global_subjectivity`, `n_unique_tokens` are among the 10 most important variables in both models.

Positive and negative coefficients from lasso model on `no_shares` data:

Table 7: Negative Coefficients

negative
n_unique_tokens

negative
n_non_stop_words
num_hrefs
average_token_length
num_keywords
kw_min_min2
kw_min_avg
LDA_01
LDA_02
max_positive_polarity
channelentertainment
channelworld
weekdayThursday
weekdayTuesday
weekdayWednesday

Table 8: Positive Coefficients

positive
n_tokens_title
num_self_hrefs
num_imgs
num_videos
kw_min_min3
kw_avg_min
kw_avg_avg
LDA_00
global_subjectivity
rate_negative_words
channellifestyle
channelmiscellaneous
channelsocmed
channeltech
weekdaySaturday
weekdaySunday

The variables with non zero coefficients are consistent with important variables from random forest and boosting methods except for `kw_max_avg`

III) Results for `no_content` data:

Error	RandomForest	Boosting	Lasso
RMSE	1.020378	1.01649	1.025581
MAE	0.7895751	0.7651273	0.793937

The 5 most important variables in predicting shares based on the random forest and boosting models for the `no_content` data is as follows.

Priority	RandomForest	Boosting
1	kw_avg_min	kw_avg_min
2	weekday	LDA_02
3	kw_avg_avg	LDA_01
4	kw_max_avg	kw_max_avg
5	title_subjectivity	kw_avg_avg

`kw_avg_min`, `kw_avg_avg`, `kw_max_avg` are among the 5 most important variables in both models.

Positive and negative coefficients from lasso model on `no_content` data:

Table 11: Negative Coefficients

negative
channelentertainment
weekdayTuesday
weekdayWednesday

Table 12: Positive Coefficients

positive
n_tokens_title
num_imgs
num_keywords
kw_max_min
kw_avg_min
kw_avg_avg
LDA_00
title_subjectivity
title_sentiment_polarity
channellifestyle
channeltech
weekdaySaturday
weekdaySunday

The variables with non zero coefficients are consistent with important variables from random forest and boosting methods except for `kw_max_avg`

Statistical Conclusions

Based on the results from the previous section, we can say that Boosting is the best model which minimizes both RMSE and MAE. The 10 most important features according to the boosting model on the majority of the data i.e. `no_zero` data includes `self_reference_avg_sharess` and `self_reference_min_shares` which are missing in the other datasets.Therefore, there is no wonder that the RMSE and MAE are higher in the other datasets. In addition, `global_subjectivity` and `n_tokens_content` are common in the most important feautures of `no_zero` and `no_shares` data. However, these columns, which happen to be important columns, are missing in the `no_content` dataset. `kw_avg_avg`, `kw_max_avg`, and `LDA_02` are the common important features for all the three parts of the data. Neither of boosting and random forest models provide

the direction of effect of the variables, however, we can use the coefficients in the lasso model to get an insight about the direction of the effect of these variables.

To sum up, three statistical models were fitted on each part of the dataset. The overall error of all three models was higher in `no_shares` than the `no_zero` data, and the error was higher in `no_content` than in `no_shares` data. The `no_content` data can be assumed as an outlier due to the small number of instances in it(~1181) and one can use the obtained model in the previous section to identify the most important features in prediction that are missing for `no_content` data and improve the overall prediction error by completing the missing information of these instances. However, the `no_shares` data is not an outlier because first, it has 5993 instances and second, the only column that were missing are `self_reference_min_shares`, `self_reference_max_shares`, and `self_reference_avg_sharess` which is completely normal for an article to have zero values for all these columns, so one may need the obtained model (for `no_shares`) data for prediction.

Conclusions in the context of the problem

In the previous section, we talked about the important variables that are derived from each of the statistical methods. The 10 most important variables based on the best performing model, boosting, on the majority of data includes `kw_avg_avg`, `self_reference_avg_sharess`, `kw_max_avg`, `self_reference_min_shares`, `num_hrefs` `kw_avg_max`, `kw_min_avg`, `LDA_02`, `global_subjectivity`, and `n_tokens_content`. By looking at their effects in table 3 and 4, we can tell that in order to have more number of shares, i.e. more popularity for an article, it is important to have more references to articles that are shared more(positive `self_ref_avg_shares` and `self_ref_min_shares`), include more keywords from the articles that have more average shares(positive `kw_max_avg` and `kw_avg_avg`), include more number of links (positive `num_href`), have a more positive global subjectivity(positive `global_subjectivity`), and finally the article should be less close to the 3rd LDA topic(negative `LDA_02`).

Future Work

The data that I think might have been very beneficial in predicting the number of times an article will be shared is the average length of each paragraph because people might not like long paragraphs. The average number of tokens between two pictures or two videos can also be useful because people might continue reading an article(and share it at last) if it is interrupted by several images or videos from time to time. This oscillation between paragraph and image(or video) must be tuned and this study along with the data of the average number of tokens between images(or videos) can help find this optimum value. Finally, some more features about the images might help with the prediction, e.g some colors in the images might attract people and encourage them to read more.

Literature Review

As explained in the main report, there has been two approaches to predict the popularity of articles in Mashable, regression and classification. Fernandes et al [2] - who also first published and cleaned this data- approached this problem as a classification problem. The goal in [2] was to predict whether an article reaches a pre-defined number of shares or in other words whether an article becomes popular or not. Different classification methods including Random Forest, Adaboost, SVM, and KNN were used for classification and the highest AUC that they obtained was 0.73 resulted from the random forest model. Also, the accuracy of their model was 0.67. In [3], again the approach was classification and the highest accuracy that they obtained was 0.748268 using gradient boosting.
In [4], the approach was regression and although `no_content` data was identified as na outlier, it was completely removed. However, here in this project we fitted a separate model on this part of the data. The smallest mse they found was 1.024665 which is higher than our best mse(071). In [5], both approaches are

present and for the regression part, their best test mean squared error was 1.225438 which is again higher than what we have obtained.

In [6], the approach was regression on shares and their best rmse 0.011 was the result of random forest model on the data after min-max scaling the whole dataset. The `n_non_stop_words` in their data is not all ones, so it is not possible to compare their rmse with the rmse in this project.

Appendix

load libraries

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(ggcorrplot)
library(gridExtra)
library(xgboost)
library(ranger)
library(caret)
library(Metrics)
library(EnvStats)
library(glmnet)
```

A.1

```
#Read data
newsPopularity =read.csv("OnlineNewsPopularity.csv", stringsAsFactors = F)
df = newsPopularity[,c(1,14:19)]
#convert one-hot encoding of channel variable into a factor variable with 7 levels
colnames(df) = c("url", "lifestyle", "entertainment", "bus", "socmed", "tech", "world")
df$miscellaneous =  ifelse( df[,2]==0 & df[,3]==0& df[,4]==0& df[,5]==0& df[,6]==0& df[, 7]==0,1,0)
df2 = df %>% gather(channel, value, -url) %>% arrange(url) %>% filter(value>0) %>% select(-value)
newsPopularity = merge(newsPopularity, df2, by = "url")
newsPopularity = newsPopularity[, -c(14:19)]
newsPopularity$channel = as.factor(newsPopularity$channel)

#convert one-hot encoding of weekday variable into a factor variable with 7 levels
df = newsPopularity %>%
  select(url, starts_with("weekday_"))
colnames(df) = c("url", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
df = df%>% gather(weekday, value, -url) %>% arrange(url) %>% filter(value>0) %>% select(-value)

newsPopularity = merge(newsPopularity, df, by = "url")
newsPopularity = newsPopularity %>% select(-starts_with(("weekday_")))
newsPopularity$weekday = as.factor(newsPopularity$weekday)

#removed is_weekend because its information is already in is_saturday or is_sunday (categorical variabl
newsPopularity$is_weekend =NULL
```

```
#removed url because it is non-predictive
newsPopularity$url = NULL
#removed timedelta because it's non-predictive
newsPopularity$timedelta = NULL
#removed abs_title_subjectivity because its information is in title_subjectivity
newsPopularity$abs_title_subjectivity = NULL
#removed abs_title_sentiment_polarity because its information is in title_sentiment_polarity
newsPopularity$abs_title_sentiment_polarity = NULL
#removed LDA_04 because of collinearity
newsPopularity$LDA_04 = NULL
#outlier removed, row 31038
newsPopularity %>% filter(n_unique_tokens<=1)-> newsPopularity
```

Correlations with shares column

```
nums <- unlist(lapply(newsPopularity, is.numeric ))
cor(newsPopularity[, nums])-> corr
sort(abs(corr[, "shares"]), decreasing = T)[1:10]
```

```
##                     shares                   kw_avg_avg
##                 1.00000000                   0.11041389
##                     LDA_03                   kw_max_avg
##                 0.08377627                   0.06430793
##                     LDA_02 self_reference_avg_sharess
##                 0.05915904                   0.05778881
##  self_reference_min_shares  self_reference_max_shares
##                 0.05595851                   0.04711666
##                  num_hrefs                   kw_avg_max
##                 0.04540398                   0.04468317
```

In the above, you can see the 10 columns that have the highest correlation with shares column.

A.2

figure1

```
s1 = ggplot(newsPopularity, aes(shares))+geom_histogram(bins = 30)
s2 = ggplot(newsPopularity, aes(log(shares)))+geom_histogram(bins = 30)
ggsave(filename = "imgs/shares_histogram.png", grid.arrange(s1, s2, ncol= 2 ), width = 6, height = 2)
```

Preprocessing on the target variable

```
newsPopularity$shares = log(newsPopularity$shares)
```

A.3

I)

```
#extracting no_zero and  no_content
no_zero = newsPopularity %>% filter(n_tokens_content>0 & !(self_reference_min_shares==0 & self_referenc
no_content  = newsPopularity %>% filter(n_tokens_content==0)
#removing all-zeroes columns in no_content
no_content$n_tokens_content = NULL
no_content$n_unique_tokens = NULL
no_content$n_non_stop_unique_tokens = NULL
no_content$num_hrefs = NULL
no_content$num_self_hrefs = NULL
no_content$average_token_length = NULL
no_content$self_reference_min_shares = NULL
no_content$self_reference_max_shares = NULL
no_content$self_reference_avg_sharess = NULL
no_content$global_subjectivity = NULL
no_content$global_sentiment_polarity = NULL
no_content$global_rate_positive_words = NULL
no_content$global_rate_negative_words = NULL
no_content$rate_negative_words = NULL
no_content$rate_positive_words = NULL
no_content$rate_positive_words = NULL
no_content$avg_positive_polarity = NULL
no_content$avg_negative_polarity = NULL
no_content$min_positive_polarity = NULL
no_content$min_negative_polarity = NULL
no_content$max_positive_polarity = NULL
no_content$max_negative_polarity = NULL
no_content$n_non_stop_words = NULL
```

Hypothesis test to see if the distribution of log of shares in `no_content` data is significantly different from in `no_zero` data

```
t.test(no_content$shares, no_zero$shares)
```

```
##
##  Welch Two Sample t-test
##
## data:  no_content$shares and no_zero$shares
## t = 5.1347, df = 1245.4, p-value = 3.276e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.09970981 0.22301643
## sample estimates:
## mean of x mean of y
##  7.656152  7.494789
```

The result of t.test, very small p.value, shows that we can reject the null hypothesis(That their distribution is the same)

II)

```
#extracting no_shares data
no_shares = newsPopularity %>% filter(n_tokens_content>0 & self_reference_min_shares==0 & self_referenc

#removing all-zeroes columns in no_shares data
no_shares$self_reference_min_shares = NULL
no_shares$self_reference_max_shares = NULL
no_shares$self_reference_avg_sharess = NULL
```

Hypothesis test to see if the distribution of log of shares in `no_shares` data is significantly different from in `no_zero` data

```
t.test(no_shares$shares, no_zero$shares)
```

```
##
##  Welch Two Sample t-test
##
## data:  no_shares$shares and no_zero$shares
## t = -12.635, df = 8364.1, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1892859 -0.1384394
## sample estimates:
## mean of x mean of y
##  7.330926  7.494789
```

The result of t.test, very small p.value, shows that we can reject the null hypothesis(That their distribution is the same)

A.4

I) preprocessing for no_zero data

```
summary(no_zero$n_non_stop_words)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```
no_zero$n_non_stop_words = NULL
```

The above statistics show that this column is all-ones and must be removed

Preliminary linear model to see the significant variables in predicting log of shares in `no_zero` data

```
summary(lm(data = no_zero, shares~.))
```

```
##
## Call:
## lm(formula = shares ~ ., data = no_zero)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1084 -0.5509 -0.1749  0.3864  5.9536
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  6.601e+00  5.126e-01  12.879  < 2e-16 ***
## n_tokens_title               1.361e-03  2.345e-03   0.580 0.561664
## n_tokens_content             4.680e-05  1.846e-05   2.536 0.011229 *
## n_unique_tokens              1.278e-01  1.583e-01   0.807 0.419437
## n_non_stop_unique_tokens    -2.150e-01  1.331e-01  -1.615 0.106273
## num_hrefs                    5.011e-03  5.248e-04   9.548  < 2e-16 ***
## num_self_hrefs              -1.310e-02  1.448e-03  -9.046  < 2e-16 ***
## num_imgs                     7.652e-04  7.205e-04   1.062 0.288240
## num_videos                   2.087e-04  1.220e-03   0.171 0.864137
## average_token_length        -4.777e-02  2.025e-02  -2.359 0.018339 *
## num_keywords                 1.898e-02  3.090e-03   6.143 8.17e-10 ***
## kw_min_min                   5.744e-04  1.423e-04   4.036 5.45e-05 ***
## kw_max_min                   1.536e-05  4.161e-06   3.691 0.000223 ***
## kw_avg_min                  -1.047e-04  2.631e-05  -3.980 6.92e-05 ***
## kw_min_max                  -3.246e-07  9.919e-08  -3.273 0.001067 **
## kw_max_max                  -2.927e-08  5.065e-08  -0.578 0.563369
## kw_avg_max                  -2.669e-07  6.750e-08  -3.955 7.68e-05 ***
## kw_min_avg                  -4.881e-05  6.210e-06  -7.861 3.94e-15 ***
## kw_max_avg                  -4.273e-05  2.085e-06 -20.498  < 2e-16 ***
## kw_avg_avg                   3.303e-04  1.179e-05  28.002  < 2e-16 ***
## self_reference_min_shares    8.580e-07  5.644e-07   1.520 0.128481
## self_reference_max_shares    1.674e-07  3.068e-07   0.546 0.585374
## self_reference_avg_sharess   9.933e-07  7.846e-07   1.266 0.205496
## LDA_00                       1.851e-01  3.834e-02   4.828 1.38e-06 ***
## LDA_01                      -1.487e-01  4.177e-02  -3.561 0.000370 ***
## LDA_02                      -2.392e-01  3.842e-02  -6.226 4.84e-10 ***
## LDA_03                      -1.295e-01  3.931e-02  -3.295 0.000986 ***
## global_subjectivity          3.425e-01  6.875e-02   4.983 6.31e-07 ***
## global_sentiment_polarity   -1.731e-01  1.344e-01  -1.288 0.197710
## global_rate_positive_words  -1.267e+00  5.912e-01  -2.143 0.032154 *
## global_rate_negative_words   1.132e+00  1.109e+00   1.022 0.307011
## rate_positive_words          2.372e-01  4.996e-01   0.475 0.635008
## rate_negative_words         -3.294e-02  5.031e-01  -0.065 0.947806
## avg_positive_polarity       -3.465e-02  1.103e-01  -0.314 0.753444
## min_positive_polarity       -2.470e-01  9.217e-02  -2.680 0.007366 **
## max_positive_polarity        4.080e-02  3.511e-02   1.162 0.245230
## avg_negative_polarity       -1.785e-01  1.021e-01  -1.748 0.080465 .
## min_negative_polarity       -3.915e-03  3.753e-02  -0.104 0.916919
## max_negative_polarity        1.282e-01  8.474e-02   1.513 0.130309
## title_subjectivity           4.179e-02  1.568e-02   2.666 0.007683 **
## title_sentiment_polarity     8.237e-02  1.922e-02   4.285 1.83e-05 ***
## channelentertainment        -2.485e-02  3.000e-02  -0.828 0.407496
## channellifestyle             2.647e-02  3.152e-02   0.840 0.400990
## channelmiscellaneous         1.768e-01  3.153e-02   5.608 2.07e-08 ***
## channelsocmed                3.501e-01  2.601e-02  13.459  < 2e-16 ***
## channeltech                  2.605e-01  2.776e-02   9.384  < 2e-16 ***
## channelworld                 1.170e-01  3.004e-02   3.895 9.85e-05 ***
```

```
## weekdayMonday              -3.250e-04  1.719e-02  -0.019 0.984921
## weekdaySaturday             1.792e-01  2.307e-02   7.769 8.16e-15 ***
## weekdaySunday               2.038e-01  2.229e-02   9.146  < 2e-16 ***
## weekdayThursday            -6.212e-02  1.694e-02  -3.667 0.000246 ***
## weekdayTuesday             -7.665e-02  1.682e-02  -4.558 5.18e-06 ***
## weekdayWednesday           -6.983e-02  1.683e-02  -4.149 3.34e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8619 on 32416 degrees of freedom
## Multiple R-squared:  0.131,  Adjusted R-squared:  0.1296
## F-statistic: 93.96 on 52 and 32416 DF,  p-value: < 2.2e-16
```

Figure2

```
sh10 = ggplot(no_zero, aes(self_reference_min_shares)) + geom_histogram(bins = 30)
sh20 = ggplot(no_zero, aes(self_reference_max_shares)) + geom_histogram(bins = 30)
sh30 = ggplot(no_zero, aes(self_reference_avg_sharess)) + geom_histogram(bins = 30)

sh1 = ggplot(no_zero, aes(log(self_reference_min_shares))) + geom_histogram(bins = 30)
sh2 = ggplot(no_zero, aes(log(self_reference_max_shares))) + geom_histogram(bins = 30)
sh3 = ggplot(no_zero, aes(log(self_reference_avg_sharess))) + geom_histogram(bins = 30)

ggsave(filename = 'imgs/self_reference_shares.png', grid.arrange(  sh10,sh1,sh20, sh2, sh30, sh3, nrow =
```

Figure3

```
n0 = ggplot(no_zero, aes(n_tokens_content)) + geom_histogram(bins = 30 )

n1 = ggplot(no_zero, aes(log(n_tokens_content))) + geom_histogram(bins = 30 )
ggsave(grid.arrange(n0, n1 , ncol = 2), filename = "imgs/n_tokens_content_no_zero.png", width = 6, heigh
```

Figure4

```
href1 = ggplot(no_zero, aes(num_hrefs)) + geom_histogram(bins = 100)
href2 = ggplot(no_zero, aes(log(num_hrefs))) + geom_histogram(bins = 10)
ggsave(grid.arrange(href1, href2 , ncol = 2), filename = "imgs/num_href_no_zero.png", width = 6, height
```

Figure5

```
shref1 = ggplot(no_zero, aes(num_self_hrefs)) + geom_histogram(bins = 50)
shref2 = ggplot(no_zero, aes(log(num_self_hrefs))) + geom_histogram(bins = 30)
ggsave(grid.arrange(shref1, shref2 , ncol = 2), filename = "imgs/num_shref_no_zero.png", width = 6, heig
```

Figure6

```
images = ggplot(no_zero, aes(num_imgs)) + geom_histogram(bins = 50)
videos = ggplot(no_zero, aes(num_videos)) + geom_histogram(bins = 50)
ggsave(grid.arrange(images, videos , ncol = 2), filename = "imgs/img_videos_no_zero.png", width = 6, hei
```

Figure7

```
kw1= ggplot(no_zero, aes(kw_min_min)) + geom_histogram(bins = 50)
kw2= ggplot(no_zero, aes(kw_max_min)) + geom_histogram(bins = 50)
kw3= ggplot(no_zero, aes(kw_avg_min)) + geom_histogram(bins = 50)
kw4= ggplot(no_zero, aes(kw_min_max)) + geom_histogram(bins = 50)
kw5= ggplot(no_zero, aes(kw_max_max)) + geom_histogram(bins = 50)
kw6= ggplot(no_zero, aes(kw_avg_max)) + geom_histogram(bins = 50)
kw7= ggplot(no_zero, aes(kw_min_avg)) + geom_histogram(bins = 50)
kw8= ggplot(no_zero, aes(kw_max_avg)) + geom_histogram(bins = 50)
kw9= ggplot(no_zero, aes(kw_avg_avg)) + geom_histogram(bins = 50)

ggsave(grid.arrange( kw1, kw2, kw3, kw4, kw5, kw6, kw7, kw8, kw9, nrow = 3, ncol = 3), width = 6, heigh
```

preprocessing on no_zero data

```
no_zero$self_reference_min_shares = log(no_zero$self_reference_min_shares)

no_zero$self_reference_max_shares = log(no_zero$self_reference_max_shares)

no_zero$self_reference_avg_sharess = log(no_zero$self_reference_avg_sharess)

no_zero$n_tokens_content =log(no_zero$n_tokens_content)
no_zero$num_hrefs = log(no_zero$num_hrefs)

no_zero[no_zero$num_self_hrefs>30, ]$num_self_hrefs= 30
no_zero[no_zero$num_imgs>25, ]$num_imgs = 25
no_zero[no_zero$num_videos>6, ]$num_videos = 6
no_zero$kw_min_min = as.factor(ifelse(no_zero$kw_min_min==-1, 0, 1))

no_zero$kw_min_max = NULL
no_zero$kw_max_max = NULL


no_zero[no_zero$kw_avg_min ==-1, ]$kw_avg_min = mean(no_zero[no_zero$kw_avg_min!=-1, ]$kw_avg_min)
no_zero[no_zero$kw_min_avg ==-1, ]$kw_min_avg = mean(no_zero[no_zero$kw_min_avg!=-1, ]$kw_min_avg)

no_zero$kw_min_avg = log(no_zero$kw_min_avg+1)
no_zero$kw_avg_min = log(no_zero$kw_avg_min+1)

no_zero$kw_avg_avg = log(no_zero$kw_avg_avg+1)
no_zero$kw_max_avg = log(no_zero$kw_max_avg+1)
no_zero$kw_max_min = log(no_zero$kw_max_min+1)
```

kw histograms after preprocessing

```
kw2= ggplot(no_zero, aes(kw_max_min)) + geom_histogram(bins = 50)
kw3= ggplot(no_zero, aes(kw_avg_min)) + geom_histogram(bins = 50)
kw7= ggplot(no_zero, aes(kw_min_avg)) + geom_histogram(bins = 50)
kw8= ggplot(no_zero, aes(kw_max_avg)) + geom_histogram(bins = 50)
kw9= ggplot(no_zero, aes(kw_avg_avg)) + geom_histogram(bins = 50)

ggsave(grid.arrange(  kw2, kw3, kw7, kw8, kw9, nrow = 5),  filename = "imgs/kw_after_no_zero.png")
```

Correlation with shares after preprocessing in `no_zero` data

```
nums <- unlist(lapply(no_zero, is.numeric ))
cor(no_zero[, nums])-> corr
sort(abs(corr[, "shares"]), decreasing = T)[1:10]
```

```
##                     shares self_reference_avg_sharess
##                  1.0000000                  0.2371480
##  self_reference_min_shares  self_reference_max_shares
##                  0.2096973                  0.2062730
##                 kw_avg_avg                 kw_max_avg
##                  0.1931371                  0.1773696
##                     LDA_02        global_subjectivity
##                  0.1568988                  0.1380272
##                  num_hrefs                     LDA_03
##                  0.1317192                  0.1252842
```

The correlation of columns with shares column has become greater.

Unchanged columns

Figure8

```
n1 = ggplot(no_zero, aes(n_tokens_title)) + geom_histogram(bins = 20)
n2 = ggplot(no_zero, aes(n_unique_tokens)) + geom_histogram(bins = 20)
n3 = ggplot(no_zero, aes(n_non_stop_unique_tokens)) + geom_histogram(bins = 20)

ggsave(grid.arrange(n1, n2,n3, ncol = 3), filename = "imgs/n_tokens_unchanged_no_zero.png", width = 6, h
```

Figure9

```
ggsave(ggplot(no_zero, aes(average_token_length)) + geom_histogram(bins = 30), filename = "imgs/avg_toke
```

Figure10

```
ggsave(ggplot(no_zero, aes(num_keywords)) + geom_bar(), filename = "imgs/num_kw_no_zero.png", width = 3
```

Figure11

```
lda0= ggplot(no_zero, aes(LDA_00)) + geom_histogram()
lda1= ggplot(no_zero, aes(LDA_01)) + geom_histogram()
lda2= ggplot(no_zero, aes(LDA_02)) + geom_histogram()
lda3= ggplot(no_zero, aes(LDA_03)) + geom_histogram()

ggsave(grid.arrange(lda0, lda1, lda2, lda3, nrow = 2, ncol = 2), filename = "imgs/LDA_no_zero.png", widt
```

Figure12

```
p1 = ggplot(no_zero, aes(avg_positive_polarity )) + geom_histogram()

p2 = ggplot(no_zero, aes(max_positive_polarity )) + geom_histogram()
p3 = ggplot(no_zero, aes(min_positive_polarity )) + geom_histogram()
```

```
p4 = ggplot(no_zero, aes(avg_negative_polarity )) + geom_histogram()

p5 = ggplot(no_zero, aes(max_negative_polarity )) + geom_histogram()
p6 = ggplot(no_zero, aes(min_negative_polarity )) + geom_histogram()

ggsave(grid.arrange(p1, p2, p3, p4, p5, p6,  nrow = 2, ncol =3), filename = "imgs/polarity_no_zero.png"
```

Figure13

```
ggsave(ggplot(no_zero, aes(global_sentiment_polarity )) + geom_histogram(bins = 30),filename = "imgs/gl
```

Figure14

```
ggsave(ggplot(no_zero, aes(title_sentiment_polarity )) + geom_histogram(bins = 30),filename = "imgs/tit

rt = rosnerTest(no_zero$title_sentiment_polarity, k=10)
print("number of outliers detected by rosner test for title_sentiment_polarity:")
```

```
## [1] "number of outliers detected by rosner test for title_sentiment_polarity:"
```

```
rt$n.outliers
```

```
## [1] 0
```

rosner test identifies no outliers in `title_sentiment_polarity` column

Figure15

```
r1 = ggplot(no_zero, aes(global_rate_positive_words )) + geom_histogram(bins = 30)
r2 = ggplot(no_zero, aes(global_rate_negative_words )) + geom_histogram(bins = 30)
r3 = ggplot(no_zero, aes(rate_positive_words )) + geom_histogram(bins = 30)
r4 = ggplot(no_zero, aes(rate_negative_words )) + geom_histogram(bins = 30)
ggsave(grid.arrange(r1,r2,r3,r4,  nrow = 2, ncol =2), filename = "imgs/rate_no_zero.png", width = 6, he
```

Figure16

```
s1 = ggplot(no_zero, aes(global_subjectivity )) + geom_histogram(bins = 30)
s2 = ggplot(no_zero, aes(title_subjectivity )) + geom_histogram(bins = 60)

ggsave(grid.arrange(s1, s2,  nrow = 2), filename = "imgs/subjectivity_no_zero.png", width = 4, height =
rt = rosnerTest(no_zero$title_subjectivity, 10)
print("number of outliers detected by rosner test for title_subjectivity:")
```

```
## [1] "number of outliers detected by rosner test for title_subjectivity:"
```

```
rt$n.outliers
```

```
## [1] 0
```

rosnerTest identifies no outlier for the `title_subjectivity` column.

Figure17

```
weekday = ggplot(no_zero, aes(x= weekday, y= shares))+ geom_bar(stat = "summary", fun.y = "mean")
channel = ggplot(no_zero, aes(x= channel, y= shares))+ geom_bar(stat = "summary", fun.y = "mean")
ggsave(grid.arrange(weekday, channel,  nrow = 2), filename = "imgs/weekday_channel_no_zero.png", height
```

II) preprocessing for no_shares data

```
summary(no_shares$n_non_stop_words)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```
no_zero$n_non_stop_words = NULL
```

The above statistics show that this column is all-ones and must be removed

Preliminary linear model to see the significant variables in predicting log of shares in `no_shares` data

```
summary(lm(data = no_shares, shares~.))
```

```
##
## Call:
## lm(formula = shares ~ ., data = no_shares)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7057 -0.4844 -0.0945  0.3982  4.7581
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  6.379e+00  9.029e-01   7.065 1.79e-12 ***
## n_tokens_title               1.725e-02  5.720e-03   3.016 0.002575 **
## n_tokens_content             1.786e-04  4.310e-05   4.145 3.44e-05 ***
## n_unique_tokens             -5.211e-02  3.698e-01  -0.141 0.887935
## n_non_stop_words                    NA         NA      NA       NA
## n_non_stop_unique_tokens     2.122e-01  3.309e-01   0.641 0.521354
## num_hrefs                   -1.965e-03  1.922e-03  -1.022 0.306816
## num_self_hrefs               1.716e-02  9.222e-03   1.861 0.062760 .
## num_imgs                    -1.182e-03  4.048e-03  -0.292 0.770309
## num_videos                   1.329e-02  6.245e-03   2.128 0.033337 *
## average_token_length        -1.709e-01  4.442e-02  -3.848 0.000120 ***
## num_keywords                -2.046e-02  7.283e-03  -2.809 0.004990 **
## kw_min_min                   1.896e-03  2.395e-04   7.919 2.83e-15 ***
## kw_max_min                   1.199e-05  1.151e-05   1.042 0.297347
## kw_avg_min                  -1.542e-04  5.891e-05  -2.618 0.008874 **
## kw_min_max                  -3.237e-07  2.382e-07  -1.359 0.174166
## kw_max_max                   2.163e-07  9.157e-08   2.362 0.018189 *
## kw_avg_max                  -6.422e-07  1.949e-07  -3.295 0.000991 ***
## kw_min_avg                  -7.160e-05  1.621e-05  -4.417 1.02e-05 ***
## kw_max_avg                  -3.928e-05  5.166e-06  -7.604 3.32e-14 ***
## kw_avg_avg                   3.665e-04  3.076e-05  11.913  < 2e-16 ***
## LDA_00                       4.030e-01  8.026e-02   5.021 5.28e-07 ***
```

```
## LDA_01                     -1.007e-01  1.008e-01  -0.999 0.317643
## LDA_02                     -5.339e-02  8.164e-02  -0.654 0.513140
## LDA_03                     -2.959e-02  1.014e-01  -0.292 0.770511
## global_subjectivity         7.809e-01  1.711e-01   4.563 5.15e-06 ***
## global_sentiment_polarity   1.353e-01  3.461e-01   0.391 0.695987
## global_rate_positive_words  1.937e-01  1.325e+00   0.146 0.883840
## global_rate_negative_words -2.689e+00  2.752e+00  -0.977 0.328543
## rate_positive_words         3.175e-01  8.690e-01   0.365 0.714811
## rate_negative_words         7.587e-01  8.817e-01   0.860 0.389564
## avg_positive_polarity       2.393e-01  2.740e-01   0.873 0.382509
## min_positive_polarity      -4.005e-01  2.365e-01  -1.693 0.090493 .
## max_positive_polarity      -2.769e-01  8.434e-02  -3.283 0.001032 **
## avg_negative_polarity       2.027e-01  2.465e-01   0.822 0.410896
## min_negative_polarity       6.991e-03  8.662e-02   0.081 0.935674
## max_negative_polarity      -2.284e-01  2.077e-01  -1.100 0.271490
## title_subjectivity         -6.408e-03  3.815e-02  -0.168 0.866622
## title_sentiment_polarity    4.349e-03  4.699e-02   0.093 0.926270
## channelentertainment        7.629e-04  7.301e-02   0.010 0.991664
## channellifestyle            1.256e-01  6.229e-02   2.017 0.043768 *
## channelmiscellaneous        7.496e-02  8.173e-02   0.917 0.359096
## channelsocmed               1.500e-01  5.977e-02   2.510 0.012100 *
## channeltech                 2.593e-01  5.935e-02   4.368 1.27e-05 ***
## channelworld                1.237e-02  6.306e-02   0.196 0.844424
## weekdayMonday              -5.797e-02  4.044e-02  -1.434 0.151755
## weekdaySaturday             3.999e-01  5.524e-02   7.240 5.05e-13 ***
## weekdaySunday               1.960e-01  5.249e-02   3.735 0.000190 ***
## weekdayThursday            -8.308e-02  3.833e-02  -2.167 0.030266 *
## weekdayTuesday             -6.530e-02  3.901e-02  -1.674 0.094193 .
## weekdayWednesday           -5.540e-02  3.855e-02  -1.437 0.150732
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8594 on 5943 degrees of freedom
## Multiple R-squared:  0.1387, Adjusted R-squared:  0.1316
## F-statistic: 19.53 on 49 and 5943 DF,  p-value: < 2.2e-16
```

Figure 18

```
n0 = ggplot(no_shares, aes(n_tokens_content)) + geom_histogram(bins = 30 )

n1 = ggplot(no_shares, aes(log(n_tokens_content))) + geom_histogram(bins = 30 )
ggsave(grid.arrange(n0, n1 , ncol = 2), filename = "imgs/n_tokens_content_no_shares.png", width = 6, he
```

Figure 19

```
href1 = ggplot(no_shares, aes(num_hrefs)) + geom_histogram(bins = 100)
href2 = ggplot(no_shares, aes(log(num_hrefs+1))) + geom_histogram(bins = 10)
ggsave(grid.arrange(href1, href2 , ncol = 2), filename = "imgs/num_href_no_shares.png", width = 6, heig
```

Figure 20

```
shref1 = ggplot(no_shares, aes(num_self_hrefs)) + geom_histogram(bins = 30)
shref2 = ggplot(no_shares, aes(log(num_self_hrefs+1))) + geom_histogram(bins = 10)
ggsave(grid.arrange(shref1, shref2 , ncol = 2), filename = "imgs/num_shref_no_shares.png", width = 6, h
```

Figure 21

```
images = ggplot(no_shares, aes(num_imgs)) + geom_histogram(bins = 50)
videos = ggplot(no_shares, aes(num_videos)) + geom_histogram(bins = 50)
ggsave(grid.arrange(images, videos , ncol = 2), filename = "imgs/img_videos_no_shares.png", width = 6, h
```

Figure 22

```
kw1= ggplot(no_shares, aes(kw_min_min)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw2= ggplot(no_shares, aes(kw_max_min)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw3= ggplot(no_shares, aes(kw_avg_min)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw4= ggplot(no_shares, aes(kw_min_max)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw5= ggplot(no_shares, aes(kw_max_max)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw6= ggplot(no_shares, aes(kw_avg_max)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw7= ggplot(no_shares, aes(kw_min_avg)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw8= ggplot(no_shares, aes(kw_max_avg)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang
kw9= ggplot(no_shares, aes(kw_avg_avg)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(ang

ggsave(grid.arrange( kw1, kw2, kw3, kw4, kw5, kw6, kw7, kw8, kw9, nrow = 3, ncol = 3), width = 6, height
```

Preprocessing on `no_shares` data

```
no_shares$n_tokens_content = log(no_shares$n_tokens_content)
no_shares$num_hrefs = log(no_shares$num_hrefs+1)

no_shares[no_shares$num_self_hrefs>5, ]$num_self_hrefs= 5
no_shares[no_shares$num_imgs>7, ]$num_imgs = 7
no_shares[no_shares$num_videos>5, ]$num_videos = 5
tmp = rep('1', nrow(no_shares))
tmp[no_shares$kw_min_min>=0 & no_shares$kw_min_min<=4]= '2'
tmp[no_shares$kw_min_min>4]= '3'

no_shares$kw_min_min = as.factor(tmp)
no_shares$kw_min_avg = log(no_shares$kw_min_avg+1)
no_shares[no_shares$kw_avg_min ==-1, ]$kw_avg_min = mean(no_shares[no_shares$kw_avg_min!=-1, ]$kw_avg_mi

no_shares$kw_avg_min = log(no_shares$kw_avg_min+1)
no_shares$kw_max_avg = log(no_shares$kw_max_avg+1)

no_shares$kw_max_min = NULL
no_shares$kw_min_max = NULL
no_shares$kw_max_max = NULL
```

Figure 23

```
kw3= ggplot(no_shares, aes(kw_avg_min)) + geom_histogram(bins = 50)
kw7= ggplot(no_shares, aes(kw_min_avg)) + geom_histogram(bins = 50)
kw8= ggplot(no_shares, aes(kw_max_avg)) + geom_histogram(bins = 50)
```

```
ggsave(grid.arrange(   kw3, kw7, kw8, nrow = 3),  filename = "imgs/kw_after_no_shares.png")
```

The unchanged columns

Figure 24

```
n1 = ggplot(no_shares, aes(n_tokens_title)) + geom_histogram(bins = 15)
n2 = ggplot(no_shares, aes(n_unique_tokens)) + geom_histogram(bins = 20)
n3 = ggplot(no_shares, aes(n_non_stop_unique_tokens)) + geom_histogram(bins = 20)

ggsave(grid.arrange(n1, n2, n3, ncol = 3), filename = "imgs/n_tokens_unchanged_no_shares.png", width = 
```

Figure 25

```
ggsave(ggplot(no_shares, aes(average_token_length)) + geom_histogram(bins = 30), filename = "imgs/avg_t
```

Figure 26

```
ggsave(ggplot(no_shares, aes(num_keywords)) + geom_bar(), filename = "imgs/num_kw_no_share.png")
```

Figure 27

```
lda0= ggplot(no_shares, aes(LDA_00)) + geom_histogram()
lda1= ggplot(no_shares, aes(LDA_01)) + geom_histogram()
lda2= ggplot(no_shares, aes(LDA_02)) + geom_histogram()
lda3= ggplot(no_shares, aes(LDA_03)) + geom_histogram()

ggsave(grid.arrange(lda0, lda1, lda2, lda3, nrow = 4), filename = "imgs/LDA_no_shares.png")
```

Figure 28

```
p1 = ggplot(no_shares, aes(avg_positive_polarity )) + geom_histogram()

p2 = ggplot(no_shares, aes(max_positive_polarity )) + geom_histogram()
p3 = ggplot(no_shares, aes(min_positive_polarity )) + geom_histogram()

p4 = ggplot(no_shares, aes(avg_negative_polarity )) + geom_histogram()

p5 = ggplot(no_shares, aes(max_negative_polarity )) + geom_histogram()
p6 = ggplot(no_shares, aes(min_negative_polarity )) + geom_histogram()

ggsave(grid.arrange(p1, p2, p3, p4, p5, p6,  nrow = 2, ncol =3), filename= "imgs/polarity_no_shares.png
```

Figure 29

```
ggsave(ggplot(no_shares, aes(global_sentiment_polarity )) + geom_histogram(bins = 30),filename = "imgs/
```

Figure 30

```
ggsave(ggplot(no_shares, aes(title_sentiment_polarity )) + geom_histogram(bins = 30),filename = "imgs/t

rt = rosnerTest(no_shares$title_sentiment_polarity, k=10)
print("number of outliers detected by rosner test for title_sentiment_polarity :")
```

```
## [1] "number of outliers detected by rosner test for title_sentiment_polarity :"
```

```
rt$n.outliers
```

```
## [1] 0
```

rosnerTest identifies no outlier in `title_sentiment_polarity` column.

Figure 31

```
r1 = ggplot(no_shares, aes(global_rate_positive_words )) + geom_histogram(bins = 30)
r2 = ggplot(no_shares, aes(global_rate_negative_words )) + geom_histogram(bins = 30)
r3 = ggplot(no_shares, aes(rate_positive_words )) + geom_histogram(bins = 30)
r4 = ggplot(no_shares, aes(rate_negative_words )) + geom_histogram(bins = 30)
ggsave(grid.arrange(r1,r2,r3,r4,  nrow = 2, ncol =2), filename = "imgs/rate_no_shares.png", width = 6, h
```

Figure 32

```
s1 = ggplot(no_shares, aes(global_subjectivity )) + geom_histogram(bins = 30)
s2 = ggplot(no_shares, aes(title_subjectivity )) + geom_histogram(bins = 60)

ggsave(grid.arrange(s1, s2,  nrow = 2), filename = "imgs/subjectivity_no_shares.png")
rt = rosnerTest(no_shares$title_subjectivity, 10)
print("number of outliers detected by rosner test:")
```

```
## [1] "number of outliers detected by rosner test:"
```

```
rt$n.outliers
```

```
## [1] 0
```

rosnerTest identifies no outlier for the `title_subjectivity` column.

categorical variables

Figure 33

```
weekday = ggplot(no_shares, aes(x= weekday, y= shares))+ geom_bar(stat = "summary", fun.y = "mean")
channel = ggplot(no_shares, aes(x= channel, y= shares))+ geom_bar(stat = "summary", fun.y = "mean")
ggsave(grid.arrange(weekday, channel,  nrow = 2), filename = "imgs/weekday_channel_no_shares.png", heig
```

III) preprocessing for no_content data

Preliminary linear model to see the significant variables in predicting log of shares in `no_content` data

```r
summary(lm(data = no_content, shares~.))
```

```
##
## Call:
## lm(formula = shares ~ ., data = no_content)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7944 -0.7247 -0.2154  0.5856  4.3738
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               6.670e+00  7.260e-01   9.188  < 2e-16 ***
## n_tokens_title            3.092e-02  1.537e-02   2.011  0.04454 *
## num_imgs                  1.019e-02  4.177e-03   2.439  0.01487 *
## num_videos                8.643e-03  2.816e-02   0.307  0.75891
## num_keywords              3.450e-02  2.435e-02   1.417  0.15676
## kw_min_min               -1.163e-03  1.786e-03  -0.651  0.51489
## kw_max_min               -3.157e-05  4.317e-05  -0.731  0.46475
## kw_avg_min                2.431e-04  3.070e-04   0.792  0.42874
## kw_min_max                4.114e-08  4.282e-07   0.096  0.92347
## kw_max_max               -8.840e-07  5.935e-07  -1.490  0.13663
## kw_avg_max                5.030e-07  4.785e-07   1.051  0.29341
## kw_min_avg               -2.544e-05  3.555e-05  -0.716  0.47443
## kw_max_avg               -1.796e-05  1.332e-05  -1.349  0.17758
## kw_avg_avg                2.278e-04  8.146e-05   2.797  0.00525 **
## LDA_00                    3.398e-01  4.460e-01   0.762  0.44633
## LDA_01                   -5.633e-01  3.407e-01  -1.653  0.09852 .
## LDA_02                   -5.302e-01  3.353e-01  -1.581  0.11406
## LDA_03                   -5.560e-01  3.290e-01  -1.690  0.09136 .
## title_subjectivity       -4.372e-02  9.225e-02  -0.474  0.63567
## title_sentiment_polarity  2.106e-01  1.044e-01   2.018  0.04382 *
## channelentertainment      2.601e-01  3.013e-01   0.863  0.38818
## channellifestyle          8.812e-01  3.647e-01   2.416  0.01584 *
## channelmiscellaneous      5.028e-01  2.910e-01   1.728  0.08428 .
## channelsocmed             5.225e-01  3.993e-01   1.309  0.19092
## channeltech               7.670e-01  3.874e-01   1.980  0.04795 *
## channelworld              6.708e-01  3.176e-01   2.112  0.03489 *
## weekdayMonday             4.803e-02  1.110e-01   0.433  0.66546
## weekdaySaturday           1.774e-01  1.453e-01   1.220  0.22265
## weekdaySunday             2.894e-01  1.435e-01   2.017  0.04397 *
## weekdayThursday          -1.333e-02  1.082e-01  -0.123  0.90192
## weekdayTuesday           -3.428e-02  1.075e-01  -0.319  0.74990
## weekdayWednesday         -1.132e-01  1.066e-01  -1.061  0.28877
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.032 on 1149 degrees of freedom
## Multiple R-squared:  0.08636,    Adjusted R-squared:  0.06171
## F-statistic: 3.504 on 31 and 1149 DF,  p-value: 5.513e-10
```

Figure 34

```
images = ggplot(no_content, aes(num_imgs)) + geom_histogram(bins = 50)
videos = ggplot(no_content, aes(num_videos)) + geom_histogram(bins = 50)
ggsave(grid.arrange(images, videos , ncol = 2), filename = "imgs/img_videos_no_content.png", width = 6,
```

Figure 35

```
kw1= ggplot(no_content, aes(kw_min_min)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw2= ggplot(no_content, aes(kw_max_min)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw3= ggplot(no_content, aes(kw_avg_min)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw4= ggplot(no_content, aes(kw_min_max)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw5= ggplot(no_content, aes(kw_max_max)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw6= ggplot(no_content, aes(kw_avg_max)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw7= ggplot(no_content, aes(kw_min_avg)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw8= ggplot(no_content, aes(kw_max_avg)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a
kw9= ggplot(no_content, aes(kw_avg_avg)) + geom_histogram(bins = 50)+theme(axis.text.x = element_text(a

ggsave(grid.arrange( kw1, kw2, kw3, kw4, kw5, kw6, kw7, kw8, kw9, nrow = 3, ncol = 3), width = 6, heigh
```

Preprocessing on `no_content` data

```
no_content[no_content$num_imgs>12, ]$num_imgs = 12
no_content[no_content$num_videos>2, ]$num_imgs = 2
no_content$kw_min_min = NULL
no_content$kw_max_max = NULL
no_content$kw_max_min = log(no_content$kw_max_min+1)
no_content$kw_min_max = log(no_content$kw_min_max+1)
no_content$kw_min_avg = as.factor(ifelse(no_content$kw_min_avg==0, 0, 1))
no_content[no_content$kw_avg_min==-1, ]$kw_avg_min = mean(no_content[no_content$kw_avg_min!=-1, ]$kw_avg
no_content$kw_avg_min = log(no_content$kw_avg_min+1)
```

Figure 36

```
kw2= ggplot(no_content, aes(kw_min_max)) + geom_histogram()

kw3= ggplot(no_content, aes(kw_avg_min)) + geom_histogram()
kw8= ggplot(no_content, aes(kw_max_min)) + geom_histogram()

ggsave(grid.arrange(  kw2,  kw3,  kw8, nrow = 3),  filename = "imgs/kw_after_no_content.png")
```

The unchanged columns

Fiigure 37

```
n1 = ggplot(no_content, aes(n_tokens_title)) + geom_histogram(bins = 15)
n2 = ggplot(no_content, aes(num_keywords)) + geom_histogram(bins = 15)
ggsave(grid.arrange(n1, n2, nrow = 2), filename = "imgs/n_tokens_unchanged.png")
```

Figure 38

```
lda0= ggplot(no_content, aes(LDA_00)) + geom_histogram()
lda1= ggplot(no_content, aes(LDA_01)) + geom_histogram()
lda2= ggplot(no_content, aes(LDA_02)) + geom_histogram()
lda3= ggplot(no_content, aes(LDA_03)) + geom_histogram()

ggsave(grid.arrange(lda0, lda1, lda2, lda3, nrow = 4), filename = "imgs/LDA_no_content.png")
```

Figure 39

```
s1 = ggplot(no_content, aes(title_sentiment_polarity )) + geom_histogram(bins = 30)
s2 = ggplot(no_content, aes(title_subjectivity )) + geom_histogram(bins = 60)
ggsave(grid.arrange(s1,s2, nrow = 2), filename = "imgs/title_no_content.png")
rt = rosnerTest(no_content$title_sentiment_polarity, k=10)
print("number of outliers detected by rosner test for title_sentiment_polarity:")
```

```
## [1] "number of outliers detected by rosner test for title_sentiment_polarity:"
```

```
rt$n.outliers
```

```
## [1] 0
```

```
rt = rosnerTest(no_content$title_subjectivity, k=10)
print("number of outliers detected by rosner test for title_subjectivity:")
```

```
## [1] "number of outliers detected by rosner test for title_subjectivity:"
```

```
rt$n.outliers
```

```
## [1] 0
```

rosnerTest identifies no outlier in the `title_sentiment_polarity` column rosnerTest identifies no outlier in the `title_subjectivity` column

Categorical variables

Figure 40

```
weekday = ggplot(no_content, aes(x= weekday, y= shares))+ geom_bar(stat = "summary", fun.y = "mean")
channel = ggplot(no_content, aes(x= channel, y= shares))+ geom_bar(stat = "summary", fun.y = "mean")
ggsave(grid.arrange(weekday, channel,  nrow = 2), filename = "imgs/weekday_channel_no_content.png", hei
```

A.5

I) Model Building for no_zero

Train/test split

createDataPartitio in caret library was used to split the `no_zero` data to train and test. This function does startified splitting. 75% of the `no_zero` data was used as the train data and the rest was used as the test data.

```
set.seed(116)
inTrain_no_zero = createDataPartition(y= no_zero$shares, p=0.75, list = F)
train_no_zero = no_zero[inTrain_no_zero, ]
test_no_zero = no_zero[-inTrain_no_zero, ]
```

random Forest

For fitting a random forest model, `ranger` function from `ranger` package was used. The values for the hyperparameters of this model were tuned by cross-validation which is more elaborated on in the next paragraph. The best hyperparameters found for this model on `no_zero` dataset are: splitrule ="extratrees", mtry = 24, and min.node.size = 1

tuning for random forest

For tuning the hyperparameters of the random forest model, 3 fold crsoss-validation with random search was used. The tuneLength was set to 30, which means that 30 different setting of hyperparameters were trained on every 2 folds of the train dataset and tested on the third fold. The result of this cross validation is the average of the measure of errors, including RMSE and MAE, on the test folds, for each hyperparameter setting. The best model can be chosen in order to minimze(or maximize) each of those measure of errors. However, we chose the best model in order to minimize the RMSE. Since the tuning part takes so long to run, we set `eval = False`

```
set.seed(820)

# Train control with random search
rs_control <- trainControl(method = "cv",
                           number = 3,
                           search = "random",
                           verboseIter = TRUE
                           )

# Training
cvranger <- train(form = shares ~ .,
                  data = train_no_zero,
                  method = "ranger",
                  tuneLength = 30,
                  trControl = rs_control
                 )
#best RMSE: mtry = 24, splitrule:extratrees, min.node.size = 1
```

In this part, we are going to train the random forest on the train dataset of `no_zero` data with the best hyperparameters and 500 trees.

```
set.seed(820)

ranger.fit = ranger(data = train_no_zero,shares~., importance ="impurity", splitrule ="extratrees", mtry
```

```
## Growing trees.. Progress: 28%. Estimated remaining time: 1 minute, 18 seconds.
## Growing trees.. Progress: 56%. Estimated remaining time: 48 seconds.
## Growing trees.. Progress: 83%. Estimated remaining time: 19 seconds.
```

```
pred <- predict(ranger.fit, data=test_no_zero)$predictions

rmse_rf_no_zero = rmse(actual = test_no_zero$shares, predicted = pred)
print("The rmse on the test dataset of no_zero data is:")
```

```
## [1] "The rmse on the test dataset of no_zero data is:"
```

```
print(rmse_rf_no_zero)
```

```
## [1] 0.8489735
```

```
mae_rf_no_zero = mae(actual = test_no_zero$shares, predicted = pred)
print("The mae on the test dataset of no_zero data is:")
```

```
## [1] "The mae on the test dataset of no_zero data is:"
```

```
print(mae_rf_no_zero)
```

```
## [1] 0.6322695
```

feature importance from random forest

The importance of the 40 predictive variables were calculated according to the `impurity` measure. Below, you can see the predictive variables sorted by their impurity. The ones with the highest impurity are the most important ones in the model.

```
imp = data.frame(features = names(importance(ranger.fit)), impurity= importance(ranger.fit))
imp = arrange(imp, desc(impurity))
knitr:: kable(imp)
```

features	impurity
self_reference_avg_sharess	819.6568
channel	774.5044
self_reference_min_shares	764.3043
kw_avg_avg	762.8581
kw_max_avg	665.0478
LDA_02	665.0273
self_reference_max_shares	649.8210
num_hrefs	633.9405
weekday	596.6984
num_imgs	585.7674
LDA_03	546.3932
n_tokens_title	523.2614
kw_avg_max	499.9863
kw_min_avg	494.5822
global_subjectivity	493.8217
LDA_01	492.8726
average_token_length	492.5144
kw_max_min	480.5497

features	impurity
title_sentiment_polarity	474.1232
n_tokens_content	473.1979
n_non_stop_unique_tokens	472.0072
n_unique_tokens	470.8627
num_keywords	465.9442
LDA_00	462.3819
kw_avg_min	461.0391
num_self_hrefs	460.2566
global_rate_positive_words	460.2327
avg_positive_polarity	458.5805
max_negative_polarity	456.2396
title_subjectivity	455.1625
num_videos	448.9328
min_positive_polarity	446.3506
avg_negative_polarity	441.6085
global_sentiment_polarity	439.9055
min_negative_polarity	434.1660
global_rate_negative_words	418.8683
max_positive_polarity	416.9420
rate_negative_words	394.9953
rate_positive_words	389.9113
kw_min_min	321.7857

Boosting

For training a boosting model on the data, `xgb.train` function from `xgboost` package was used, and also for tuning the hyperparameters, the `xgb.cv` function from the same library was used to do the cross-validation.

prepare data for boosting

The `xgb.cv` and `xgb.train` functions require the datasets(train and test datasets) to be an object of `xgb.DMatrix`. To make the datasets an object of `xgb.DMatrix`, the datasets first need to be matrices. Therefore I first had to hot-encode the factor columns of the datasets.

```
labels = train_no_zero$shares
 # ts_label <- test$target
new_tr <- model.matrix(~.+0,data = train_no_zero[,-39])
new_ts <- model.matrix(~.+0,data = test_no_zero[,-39])
dim(new_tr)
```

```
## [1] 24353    51
```

```
dim(new_ts)
```

```
## [1] 8116   51
```

```
trainObject = xgb.DMatrix(new_tr, label= labels)
testObject = xgb.DMatrix(new_ts)
```

tuning parameters for boosting

The `xgboost` package has an internal cross-validation function `xgb.cv`. This function does the nfold cross validation(3 fold in our case) on the the data(`train_no_zero` in our case). It trains the model on the training folds and calculates the test RMSE(`eval_metric = RMSE`) on the validation folds. This process is repeated until the test rmse is not further reduced for a certain number of rounds(This parameter is early stopping round and was set to 10 in our tuning process.). This function only does the cross validation part of model tuning, so for the `search` part we randomly selected 50 parameter settings for the `xgb.train` function from the following ranges:

- `max_depth` a random selected number from 6,7,8,9, and 10.
- `eta` = a uniform random number from this range (0.01, 0.3)
- `gamma` = a uniform random number from this range (0, 1)
- `subsample` = a uniform random number from this range (0.6, 0.9)
- `colsample_bytree` = a uniform random number from this range (0.5, 0.8)
- `min_child_weight` = a random selected number from 1,2,3,. . . ,40.
- `nrounds`: This parameter is selected by `xgb.cv` for each parameter setting.
- Seed.number: This is not an argument of `xgb.train` function, but affects the model training. Therefore, I also tried different seeds in each iteration.

The best parameter setting, and best seed number was updated at the end of each iteration only if the new parameter setting had led to a lower test_rmse. The best setting of parameters are: `max_depth`: 7, `eta` : 0.01987414, `gamma`: 0.1376651, `subsample`: 0.6533908, `colsample_bytree` : 0.7397481, `min_child_weight`: 38, `nrounds`: 328 The best seed number was 339.
Since the tuning part takes so long to run, eval was set to False

```
set.seed(820)
best_param = list()
best_seednumber = 1234
best_logloss = Inf
best_logloss_index = 0
start = Sys.time()
for (iter in 1:50) {
    param <- list(objective = "reg:squarederror",
          eval_metric = "rmse",
          max_depth = sample(6:10, 1),
          eta = runif(1, .01, .3),
          gamma = runif(1, 0.0, 1),
          subsample = runif(1, .6, .9),
          colsample_bytree = runif(1, .5, .8),
          min_child_weight = sample(1:40, 1)
          )
    cv.nround = 1000
    cv.nfold = 3
    seed.number = sample.int(10000, 1)[[1]]
    set.seed(seed.number)
    mdcv <- xgb.cv(data=trainObject, params = param, nthread=6,
                   nfold=cv.nfold, nrounds=cv.nround,
                   verbose = T, early.stop.round=8, maximize=FALSE, print_every_n = 20)

    min_logloss = min(mdcv$evaluation_log[, test_rmse_mean])
    min_logloss_index = which.min(mdcv$evaluation_log[, test_rmse_mean])
```

```
    if (min_logloss < best_logloss) {
        best_logloss = min_logloss
        best_logloss_index = min_logloss_index
        best_seednumber = seed.number
        best_param = param
    }
}
finish = Sys.time()
```

model fitting boosting

In this part, we are going to train the boosting model on the train dataset of `no_zero` data with the best hyperparameters.

```
set.seed(339)
nround = 328
best_param= list(
          max_depth = 7,
          eta = 0.01987414,
          gamma = 0.1376651,
          subsample = 0.6533908,
          colsample_bytree = 0.7397481,
          min_child_weight = 38)
xgb.fit <- xgb.train(data=trainObject, params=best_param, nrounds=nround, nthread=6)
pred <- predict(xgb.fit, newdata=testObject)
res = data.frame(pred = pred, actual= test_no_zero$shares)
```

```
rmse_xgb_no_zero = rmse(actual = test_no_zero$shares, predicted = pred)
print("The rmse on the test dataset of no_zero data is:")
```

```
## [1] "The rmse on the test dataset of no_zero data is:"
```

```
print(rmse_xgb_no_zero)
```

```
## [1] 0.8429064
```

```
mae_xgb_no_zero = mae(actual = test_no_zero$shares, predicted = pred)
print("The mae on the test dataset of no_zero data is:")
```

```
## [1] "The mae on the test dataset of no_zero data is:"
```

```
print(mae_xgb_no_zero)
```

```
## [1] 0.617826
```

feature importance

The importance of the 40 predictive variables were calculated based on their relative influence. This measure which indicates the relative contribution of each feature to the model is stored in the `Gain` column in the table below. You can see the predictive variables sorted by their Gain. The ones with the highest Gain are the most important ones in the model.

```
knitr::kable(xgb.importance(model = xgb.fit))
```

Feature	Gain	Cover	Frequency
kw_avg_avg	0.1646130	0.0857647	0.0505160
self_reference_avg_sharess	0.0692272	0.0448054	0.0355433
kw_max_avg	0.0587727	0.0482209	0.0434343
self_reference_min_shares	0.0569632	0.0488925	0.0371619
kw_avg_max	0.0337267	0.0418092	0.0439738
num_hrefs	0.0310450	0.0405367	0.0314966
kw_min_avg	0.0305128	0.0465791	0.0323059
LDA_02	0.0276923	0.0335359	0.0347339
n_unique_tokens	0.0272131	0.0323696	0.0344642
n_tokens_content	0.0260322	0.0264146	0.0339920
LDA_01	0.0251473	0.0223389	0.0362177
kw_avg_min	0.0233817	0.0240638	0.0335874
global_subjectivity	0.0219128	0.0254275	0.0309570
LDA_03	0.0218547	0.0214185	0.0317664
self_reference_max_shares	0.0216660	0.0195017	0.0252243
n_non_stop_unique_tokens	0.0211546	0.0171392	0.0291360
num_imgs	0.0204615	0.0250917	0.0213799
average_token_length	0.0201945	0.0254030	0.0306873
kw_max_min	0.0196725	0.0176418	0.0285290
global_rate_positive_words	0.0194069	0.0172263	0.0306198
avg_negative_polarity	0.0188980	0.0127767	0.0279895
avg_positive_polarity	0.0187456	0.0198958	0.0275174
LDA_00	0.0181215	0.0190950	0.0261685
global_rate_negative_words	0.0166019	0.0119745	0.0251568
global_sentiment_polarity	0.0156634	0.0196536	0.0236056
title_sentiment_polarity	0.0126071	0.0190480	0.0162541
channelentertainment	0.0124355	0.0182026	0.0075538
channeltech	0.0124293	0.0185224	0.0063398
weekdaySunday	0.0120759	0.0270339	0.0087003
rate_positive_words	0.0119536	0.0089178	0.0179402
weekdaySaturday	0.0109714	0.0244379	0.0075538
num_videos	0.0109516	0.0213502	0.0122749
title_subjectivity	0.0108584	0.0090202	0.0161192
min_positive_polarity	0.0106144	0.0178529	0.0128819
channelsocmed	0.0092623	0.0135980	0.0063398
num_self_hrefs	0.0081897	0.0129318	0.0113981
n_tokens_title	0.0078799	0.0063242	0.0123423
max_negative_polarity	0.0069387	0.0082453	0.0102516
min_negative_polarity	0.0066469	0.0036206	0.0103190
channelmiscellaneous	0.0049504	0.0048430	0.0040467
max_positive_polarity	0.0039937	0.0016207	0.0066096
rate_negative_words	0.0034643	0.0031491	0.0053281
num_keywords	0.0034371	0.0023849	0.0056653

Feature	Gain	Cover	Frequency
weekdayWednesday	0.0026716	0.0091812	0.0039118
kw_min_min0	0.0023382	0.0024835	0.0037094
channellifestyle	0.0020146	0.0073151	0.0024280
weekdayTuesday	0.0019749	0.0082811	0.0026303
weekdayMonday	0.0007954	0.0004070	0.0008768
weekdayThursday	0.0007852	0.0025831	0.0010791
channelworld	0.0006974	0.0007634	0.0004721
kw_min_min1	0.0003816	0.0003055	0.0008093

lasso

The last model that was trained on the `no_zero` data is lasso. This model also has a hyperparameter which controls the regularization and needs to be tuned. `glmnet` library was used to both tune the model and fit it on the train data. `cv.glmnet` does a k fold(10 fold in our case) cross-validation and searches for the best lambda among a given list of lambdas. This list in our case was given by $lambda_seq <- 10^seq(2, -2, by = -.1)$ and the best lambda which minimized the average `mse` on the test folds was 0.01. Note that rmse that we tried to minimize in the other sections is the root of mse and since square root is monotone, the lambda which minimizes mse also minimized the rmse.

```
set.seed(820)
lambda_seq <- 10^seq(2, -2, by = -.1)
cv_output <- cv.glmnet(new_tr, labels,
            alpha = 1, lambda = lambda_seq, type.measure = "mse", nfold = 10)

# identifying best lamda
best_lam <- cv_output$lambda.min
```

In this part, we are going to train the lasso model on the train dataset of `no_zero` data with the best lambda.

```
best_lam = 0.01
lasso.fit = glmnet(new_tr, labels,  alpha = 1, lambda = 0.01)
pred <- predict(lasso.fit, s = best_lam, newx = new_ts)
rmse_lasso_no_zero = rmse(actual = test_no_zero$shares, predicted = pred)
print("The rmse on the test dataset of no_zero data is:")
```

```
## [1] "The rmse on the test dataset of no_zero data is:"
```

```
print(rmse_lasso_no_zero)
```

```
## [1] 0.8641841
```

```
mae_lasso_no_zero = mae(actual = test_no_zero$shares, predicted = pred)
print("The mae on the test dataset of no_zero data is:")
```

```
## [1] "The mae on the test dataset of no_zero data is:"
```

```
print(mae_lasso_no_zero)
```

```
## [1] 0.6418774
```

Feature importance from lasso

Below, you can see the coefficients of each feature, and the ones with no value means that they were shrinked to zero. In other words, those shrinked to zero are the unimportant variables according to the lasso model.

```
coef_lasso_no_zero = coef(lasso.fit)
coef_lasso_noZero =data.frame(names = rownames(as.matrix(coef_lasso_no_zero)), values = as.matrix(coef_
positive_no_zero = as.character(coef_lasso_noZero[ coef_lasso_noZero$values>0, 'names'])
negative_no_zero = as.character(coef_lasso_noZero[ coef_lasso_noZero$values<0, 'names'])
print(coef_lasso_no_zero)
```

```
## 52 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                 4.486598e+00
## n_tokens_title              .
## n_tokens_content            .
## n_unique_tokens            -1.172203e-01
## n_non_stop_unique_tokens    .
## num_hrefs                   6.651380e-02
## num_self_hrefs             -1.814109e-03
## num_imgs                    3.513828e-03
## num_videos                  5.788530e-03
## average_token_length       -7.590171e-03
## num_keywords                1.181978e-02
## kw_min_min0                 2.871480e-03
## kw_min_min1                -1.995060e-14
## kw_max_min                  7.864206e-03
## kw_avg_min                  2.271145e-04
## kw_avg_max                  .
## kw_min_avg                  .
## kw_max_avg                  .
## kw_avg_avg                  1.785541e-01
## self_reference_min_shares   7.061749e-02
## self_reference_max_shares   .
## self_reference_avg_sharess  8.583648e-02
## LDA_00                      1.987535e-02
## LDA_01                     -3.444703e-02
## LDA_02                     -2.383843e-01
## LDA_03                      .
## global_subjectivity         3.396369e-01
## global_sentiment_polarity   .
## global_rate_positive_words  .
## global_rate_negative_words  .
## rate_positive_words         9.685667e-03
## rate_negative_words         .
## avg_positive_polarity       .
## min_positive_polarity      -1.747080e-01
## max_positive_polarity       .
```

```
## avg_negative_polarity      -1.359936e-02
## min_negative_polarity       .
## max_negative_polarity       .
## title_subjectivity          4.050657e-02
## title_sentiment_polarity    5.225845e-02
## channelentertainment       -1.040717e-01
## channellifestyle            .
## channelmiscellaneous        1.425271e-01
## channelsocmed               2.639451e-01
## channeltech                 1.017529e-01
## channelworld                .
## weekdayMonday               .
## weekdaySaturday             1.805498e-01
## weekdaySunday               1.888215e-01
## weekdayThursday            -1.219025e-02
## weekdayTuesday             -3.760479e-02
## weekdayWednesday           -4.229175e-02
```

II) Model Building for no_shares

Train/test split

```
set.seed(116)
inTrain_no_shares = createDataPartition(y= no_shares$shares, p=0.75, list = F)
train_no_shares = no_shares[inTrain_no_shares, ]
test_no_shares = no_shares[-inTrain_no_shares, ]
```

random Forest

The best hyperparameters found for this model on `no_shares` dataset are: splitrule =“extratrees”, mtry = 20, and min.node.size = 4

tuning for random forest

```
set.seed(820)

# Train control with random search
rs_control <- trainControl(method = "cv",
                           number = 3,
                           search = "random",
                           verboseIter = TRUE
                           )

# Training
cvranger <- train(form = shares ~ .,
                  data = train_no_shares,
                  method = "ranger",
                  tuneLength = 30,
```

```
                    trControl = rs_control
                   )
```

In this part, we are going to train the random forest on the train dataset of `no_shares` data with the best hyperparameters and 500 trees.

```
set.seed(820)

ranger.fit = ranger(data = train_no_shares,shares~., importance ="impurity", splitrule ="extratrees", m
pred <- predict(ranger.fit, data=test_no_shares)$predictions

rmse_rf_no_shares = rmse(actual = test_no_shares$shares, predicted = pred)
print("The rmse on the test dataset of no_shares data is:")
```

```
## [1] "The rmse on the test dataset of no_shares data is:"
```

```
print(rmse_rf_no_shares)
```

```
## [1] 0.8342359
```

```
mae_rf_no_shares = mae(actual = test_no_shares$shares, predicted = pred)
print("The mae on the test dataset of no_shares data is:")
```

```
## [1] "The mae on the test dataset of no_shares data is:"
```

```
print(mae_rf_no_shares)
```

```
## [1] 0.6137843
```

feature importance from random forest

```
imp = data.frame(features = names(importance(ranger.fit)), impurity= importance(ranger.fit))
imp = arrange(imp, desc(impurity))
knitr:: kable(imp)
```

features	impurity
LDA_02	190.66438
kw_avg_avg	155.70096
channel	148.47902
weekday	117.24566
average_token_length	116.43835
num_hrefs	114.91621
kw_max_avg	113.75449
kw_min_min	112.61240
kw_avg_max	112.09700
LDA_00	109.28902
n_tokens_title	108.11441

features	impurity
LDA_01	108.10627
num_keywords	107.94733
n_unique_tokens	106.49390
global_subjectivity	106.37646
kw_avg_min	100.93811
n_non_stop_unique_tokens	98.00146
LDA_03	97.63831
n_tokens_content	94.01922
avg_positive_polarity	92.39849
global_rate_positive_words	88.13223
min_negative_polarity	87.82390
min_positive_polarity	84.53970
global_sentiment_polarity	84.34962
title_subjectivity	84.15530
title_sentiment_polarity	83.68925
kw_min_avg	83.33394
max_negative_polarity	81.99742
num_videos	81.27456
max_positive_polarity	80.18585
global_rate_negative_words	79.22512
avg_negative_polarity	79.15829
rate_negative_words	75.76975
n_non_stop_words	75.36827
rate_positive_words	75.18847
num_imgs	70.42634
num_self_hrefs	67.83773

Boosting

prepare data for boosting

```
labels = train_no_shares$shares
new_tr <- model.matrix(~.+0,data = train_no_shares[,-36])
new_ts <- model.matrix(~.+0,data = test_no_shares[,-36])
dim(new_tr)
```

```
## [1] 4497   49
```

```
dim(new_ts)
```

```
## [1] 1496   49
```

```
trainObject = xgb.DMatrix(new_tr, label= labels)
testObject = xgb.DMatrix(new_ts)
```

tuning parameters for boosting

The best setting of parameters are: `max_depth`: 8, `eta` : 0.01351051, `gamma`: 0.7468976, `subsample`: 0.8414548, `colsample_bytree` : 0.5062725, `min_child_weight`: 31, `nrounds`: 404 The best seed number was 3361

```
set.seed(820)
best_param = list()
best_seednumber = 1234
best_logloss = Inf
best_logloss_index = 0
start = Sys.time()
for (iter in 1:50) {
    param <- list(objective = "reg:squarederror",
          eval_metric = "rmse",
          max_depth = sample(6:10, 1),
          eta = runif(1, .01, .3),
          gamma = runif(1, 0.0, 1),
          subsample = runif(1, .6, .9),
          colsample_bytree = runif(1, .5, .8),
          min_child_weight = sample(1:40, 1)
          )
    cv.nround = 1000
    cv.nfold = 3
    seed.number = sample.int(10000, 1)[[1]]
    set.seed(seed.number)
    mdcv <- xgb.cv(data=trainObject, params = param, nthread=6,
                   nfold=cv.nfold, nrounds=cv.nround,
                   verbose = T, early.stop.round=8, maximize=FALSE, print_every_n = 20)

    min_logloss = min(mdcv$evaluation_log[, test_rmse_mean])
    min_logloss_index = which.min(mdcv$evaluation_log[, test_rmse_mean])

    if (min_logloss < best_logloss) {
        best_logloss = min_logloss
        best_logloss_index = min_logloss_index
        best_seednumber = seed.number
        best_param = param
    }
}
finish = Sys.time()
```

model fitting boosting

In this part, we are going to train the boosting model on the train dataset of `no_shares` data with the best hyperparameters.

```
set.seed(3361)
nround = 404
best_param= list(
          max_depth = 8,
          eta = 0.01351051,
          gamma = 0.7468976,
          subsample = 0.8414548,
          colsample_bytree = 0.5062725,
          min_child_weight = 31)
xgb.fit <- xgb.train(data=trainObject, params=best_param, nrounds=nround, nthread=6)
pred <- predict(xgb.fit, newdata=testObject)
```

```
res = data.frame(pred = pred, actual= test_no_shares$shares)
```

```
rmse_xgb_no_shares = rmse(actual = test_no_shares$shares, predicted = pred)
print("The rmse on the test dataset of no_shares data is:")
```

```
## [1] "The rmse on the test dataset of no_shares data is:"
```

```
print(rmse_xgb_no_shares)
```

```
## [1] 0.8235383
```

```
mae_xgb_no_shares = mae(actual = test_no_shares$shares, predicted = pred)
print("The mae on the test dataset of no_shares data is:")
```

```
## [1] "The mae on the test dataset of no_shares data is:"
```

```
print(mae_xgb_no_shares)
```

```
## [1] 0.5985847
```

feature importance

```
knitr::kable(xgb.importance(model = xgb.fit))
```

Feature	Gain	Cover	Frequency
kw_avg_avg	0.0809967	0.0851020	0.0526445
LDA_02	0.0577114	0.0377634	0.0390535
n_unique_tokens	0.0471176	0.0412178	0.0399542
kw_max_avg	0.0463170	0.0427616	0.0415916
kw_avg_max	0.0427812	0.0372436	0.0478140
LDA_03	0.0419904	0.0485901	0.0472409
LDA_00	0.0404595	0.0345583	0.0441297
LDA_01	0.0387679	0.0370964	0.0416735
n_tokens_content	0.0364957	0.0420088	0.0383986
avg_positive_polarity	0.0347900	0.0547209	0.0420829
kw_avg_min	0.0347837	0.0290248	0.0385623
global_rate_positive_words	0.0325134	0.0258890	0.0374980
n_non_stop_unique_tokens	0.0312331	0.0228621	0.0361880
global_subjectivity	0.0309817	0.0294596	0.0341411
average_token_length	0.0306708	0.0301026	0.0364336
num_hrefs	0.0290511	0.0444688	0.0308662
channelworld	0.0266653	0.0127962	0.0035206
avg_negative_polarity	0.0249175	0.0227196	0.0318487
kw_min_avg	0.0244177	0.0317457	0.0283281
global_sentiment_polarity	0.0230770	0.0222388	0.0287375
rate_positive_words	0.0218425	0.0128107	0.0276732
global_rate_negative_words	0.0191802	0.0115039	0.0241526

Feature	Gain	Cover	Frequency
kw_min_min3	0.0185117	0.0242273	0.0086786
weekdaySaturday	0.0171977	0.0325610	0.0080236
n_tokens_title	0.0163634	0.0167983	0.0186671
min_negative_polarity	0.0128542	0.0082278	0.0161290
kw_min_min2	0.0127044	0.0107850	0.0121991
title_subjectivity	0.0125595	0.0085607	0.0174390
rate_negative_words	0.0118988	0.0087494	0.0144916
title_sentiment_polarity	0.0100879	0.0109375	0.0126904
num_keywords	0.0089516	0.0062282	0.0093336
max_positive_polarity	0.0087706	0.0086690	0.0112985
min_positive_polarity	0.0086059	0.0068498	0.0110529
num_imgs	0.0076542	0.0074579	0.0079417
max_negative_polarity	0.0074638	0.0059335	0.0108891
channelsocmed	0.0072401	0.0154346	0.0065499
channeltech	0.0069029	0.0145959	0.0054036
channelentertainment	0.0065039	0.0113770	0.0056493
num_videos	0.0059017	0.0129613	0.0058130
weekdaySunday	0.0058164	0.0183965	0.0054036
num_self_hrefs	0.0049944	0.0052517	0.0054036
channellifestyle	0.0042370	0.0037366	0.0040118
kw_min_min1	0.0032845	0.0016301	0.0042574
weekdayThursday	0.0023444	0.0023615	0.0028656
weekdayWednesday	0.0009144	0.0005195	0.0012281
weekdayMonday	0.0006634	0.0004694	0.0009006
weekdayTuesday	0.0005902	0.0003309	0.0009006
channelmiscellaneous	0.0002217	0.0002649	0.0002456

lasso

The best lambda is 0.01

```
set.seed(820)
lambda_seq <- 10^seq(2, -2, by = -.1)
cv_output <- cv.glmnet(new_tr, labels,
            alpha = 1, lambda = lambda_seq, type.measure = "mse", nfold = 10)

# identifying best lamda
best_lam <- cv_output$lambda.min
```

In this part, we are going to train the lasso model on the train dataset of `no_shares` data with the best lambda.

```
best_lam = 0.01
lasso.fit = glmnet(new_tr, labels,  alpha = 1, lambda = 0.01)
pred <- predict(lasso.fit, s = best_lam, newx = new_ts)
rmse_lasso_no_shares = rmse(actual = test_no_shares$shares, predicted = pred)
print("The rmse on the test dataset of no_shares data is:")
```

```
## [1] "The rmse on the test dataset of no_shares data is:"
```

```r
print(rmse_lasso_no_shares)
```

```
## [1] 0.8430998
```

```r
mae_lasso_no_shares = mae(actual = test_no_shares$shares, predicted = pred)
print("The mae on the test dataset of no_shares data is:")
```

```
## [1] "The mae on the test dataset of no_shares data is:"
```

```r
print(mae_lasso_no_shares)
```

```
## [1] 0.6186547
```

Feature importance from lasso

```r
coef_lasso_no_shares = coef(lasso.fit)

coef_lasso_noShares =data.frame(names = rownames(as.matrix(coef_lasso_no_shares)), values = as.matrix(co
positive_no_shares = as.character(coef_lasso_noShares[ coef_lasso_noShares$values>0, 'names'])
negative_no_shares = as.character(coef_lasso_noShares[ coef_lasso_noShares$values<0, 'names'])

print(coef_lasso_no_shares)
```

```
## 50 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)                4.015929e+06
## n_tokens_title             9.766898e-03
## n_tokens_content           .
## n_unique_tokens           -2.485803e-01
## n_non_stop_words          -4.015921e+06
## n_non_stop_unique_tokens   .
## num_hrefs                 -2.236315e-03
## num_self_hrefs             2.245700e-02
## num_imgs                   1.659209e-03
## num_videos                 4.062347e-02
## average_token_length      -1.842235e-01
## num_keywords              -6.576801e-03
## kw_min_min1                .
## kw_min_min2               -1.127508e-01
## kw_min_min3                1.433679e-01
## kw_avg_min                 2.630430e-02
## kw_avg_max                -7.867022e-08
## kw_min_avg                -3.724099e-03
## kw_max_avg                 .
## kw_avg_avg                 1.173410e-04
## LDA_00                     1.881641e-01
## LDA_01                    -1.073657e-01
## LDA_02                    -9.494395e-02
## LDA_03                     .
```

```
## global_subjectivity         5.507072e-01
## global_sentiment_polarity   .
## global_rate_positive_words  .
## global_rate_negative_words  .
## rate_positive_words         .
## rate_negative_words         5.294835e-02
## avg_positive_polarity       .
## min_positive_polarity       .
## max_positive_polarity      -6.799897e-02
## avg_negative_polarity       .
## min_negative_polarity       .
## max_negative_polarity       .
## title_subjectivity          .
## title_sentiment_polarity    .
## channelentertainment       -9.976551e-02
## channellifestyle            7.868440e-02
## channelmiscellaneous        1.575901e-02
## channelsocmed               1.274311e-01
## channeltech                 1.719365e-01
## channelworld               -1.230282e-01
## weekdayMonday               .
## weekdaySaturday             3.591497e-01
## weekdaySunday               2.010165e-01
## weekdayThursday            -3.855278e-02
## weekdayTuesday              .
## weekdayWednesday            .
```

III) Model Building for no_content

Train/test split

```
set.seed(116)
inTrain_no_content = createDataPartition(y= no_content$shares, p=0.75, list = F)
train_no_content = no_content[inTrain_no_content, ]
test_no_content = no_content[-inTrain_no_content, ]
```

random Forest

The best hyperparameters found for this model on `no_content` dataset are: splitrule =“maxstat”, mtry = 9, and min.node.size = 2

tuning for random forest

```
set.seed(820)

# Train control with random search
rs_control <- trainControl(method = "cv",
                           number = 3,
```

```
                           search = "random",
                           verboseIter = TRUE
                           )

# Training
cvranger <- train(form = shares ~ .,
                  data = train_no_content,
                  method = "ranger",
                  tuneLength = 30,
                  trControl = rs_control
                 )
```

In this part, we are going to train the random forest on the train dataset of `no_content` data with the best hyperparameters and 500 trees.

```
set.seed(820)

ranger.fit = ranger(data = train_no_content,shares~., importance ="impurity", splitrule ="maxstat", mtry
pred <- predict(ranger.fit, data=test_no_content)$predictions

rmse_rf_no_content = rmse(actual = test_no_content$shares, predicted = pred)
print("The rmse on the test dataset of no_content data is:")
```

```
## [1] "The rmse on the test dataset of no_content data is:"
```

```
print(rmse_rf_no_content)
```

```
## [1] 1.097029
```

```
mae_rf_no_content = mae(actual = test_no_content$shares, predicted = pred)
print("The mae on the test dataset of no_content data is:")
```

```
## [1] "The mae on the test dataset of no_content data is:"
```

```
print(mae_rf_no_content)
```

```
## [1] 0.8450576
```

feature importance from random forest

```
imp = data.frame(features = names(importance(ranger.fit)), impurity= importance(ranger.fit))
imp = arrange(imp, desc(impurity))
knitr:: kable(imp)
```

features	impurity
weekday	24.325095
kw_avg_min	23.561396

features	impurity
kw_avg_avg	23.390372
title_subjectivity	22.329562
kw_max_avg	21.889552
kw_max_min	21.865304
LDA_01	21.798203
n_tokens_title	21.172697
LDA_00	21.149735
kw_avg_max	21.029881
LDA_03	20.563172
LDA_02	19.749982
title_sentiment_polarity	19.559742
kw_min_max	19.487999
num_keywords	16.894362
num_imgs	13.529261
channel	12.694568
kw_min_avg	9.982127
num_videos	8.468817

Boosting

prepare data for boosting

```
labels = train_no_content$shares
new_tr <- model.matrix(~.+0,data = train_no_content[,-18])
new_ts <- model.matrix(~.+0,data = test_no_content[,-18])
dim(new_tr)
```

```
## [1] 887  30
```

```
dim(new_ts)
```

```
## [1] 294  30
```

```
trainObject = xgb.DMatrix(new_tr, label= labels)
testObject = xgb.DMatrix(new_ts)
```

tuning parameters for boosting

The best setting of parameters are: `max_depth`: 9, `eta` : 0.03783421, `gamma`: 0.4373984, `subsample`: 0.8991138, `colsample_bytree` : 0.6359289, `min_child_weight`: 26, `nrounds`: 118 The best seed number was 3326.

```
set.seed(820)
best_param = list()
best_seednumber = 1234
best_logloss = Inf
best_logloss_index = 0
start = Sys.time()
```

```
for (iter in 1:50) {
    param <- list(objective = "reg:squarederror",
          eval_metric = "rmse",
          max_depth = sample(6:10, 1),
          eta = runif(1, .01, .3),
          gamma = runif(1, 0.0, 1),
          subsample = runif(1, .6, .9),
          colsample_bytree = runif(1, .5, .8),
          min_child_weight = sample(1:40, 1)
          )
    cv.nround = 1000
    cv.nfold = 3
    seed.number = sample.int(10000, 1)[[1]]
    set.seed(seed.number)
    mdcv <- xgb.cv(data=trainObject, params = param, nthread=6,
                   nfold=cv.nfold, nrounds=cv.nround,
                   verbose = T, early.stop.round=8, maximize=FALSE, print_every_n = 20)

    min_logloss = min(mdcv$evaluation_log[, test_rmse_mean])
    min_logloss_index = which.min(mdcv$evaluation_log[, test_rmse_mean])

    if (min_logloss < best_logloss) {
        best_logloss = min_logloss
        best_logloss_index = min_logloss_index
        best_seednumber = seed.number
        best_param = param
    }
}
finish = Sys.time()
```

model fitting boosting

In this part, we are going to train the boosting model on the train dataset of `no_content` data with the best hyperparameters.

```
set.seed(3326)
nround = 118
best_param= list(
          max_depth = 9,
          eta = 0.03783421,
          gamma = 0.4373984,
          subsample = 0.8991138,
          colsample_bytree = 0.6359289,
          min_child_weight = 26)
xgb.fit <- xgb.train(data=trainObject, params=best_param, nrounds=nround, nthread=6)
pred <- predict(xgb.fit, newdata=testObject)
res = data.frame(pred = pred, actual= test_no_content$shares)
```

```
rmse_xgb_no_content = rmse(actual = test_no_content$shares, predicted = pred)
print("The rmse on the test dataset of no_content data is:")
```

```
## [1] "The rmse on the test dataset of no_content data is:"
```

```
print(rmse_xgb_no_content)
```

```
## [1] 1.119927
```

```
mae_xgb_no_content = mae(actual = test_no_content$shares, predicted = pred)
print("The mae on the test dataset of no_content data is:")
```

```
## [1] "The mae on the test dataset of no_content data is:"
```

```
print(mae_xgb_no_content)
```

```
## [1] 0.8495623
```

feature importance

```
knitr::kable(xgb.importance(model = xgb.fit))
```

Feature	Gain	Cover	Frequency
kw_avg_min	0.1134847	0.1187865	0.1074481
LDA_02	0.1094301	0.0936501	0.1001221
kw_avg_avg	0.0927689	0.0855525	0.0793651
kw_max_avg	0.0924445	0.0981647	0.0989011
LDA_01	0.0922560	0.1005111	0.0860806
LDA_00	0.0719674	0.0619468	0.0805861
LDA_03	0.0718424	0.0527222	0.0750916
kw_min_max	0.0592096	0.0395325	0.0592186
kw_avg_max	0.0555356	0.0564919	0.0622711
title_subjectivity	0.0539062	0.0519987	0.0549451
title_sentiment_polarity	0.0472196	0.0749969	0.0506716
kw_max_min	0.0471893	0.0462433	0.0525031
n_tokens_title	0.0186446	0.0167705	0.0213675
channelentertainment	0.0173124	0.0286206	0.0122100
num_imgs	0.0168955	0.0156434	0.0158730
weekdaySunday	0.0095034	0.0284166	0.0109890
weekdayMonday	0.0093407	0.0115045	0.0103785
weekdayWednesday	0.0085888	0.0072560	0.0079365
num_keywords	0.0033773	0.0012644	0.0036630
kw_min_avg0	0.0021335	0.0023614	0.0030525
weekdayTuesday	0.0018956	0.0014534	0.0018315
weekdayThursday	0.0017515	0.0034563	0.0018315
weekdaySaturday	0.0012014	0.0017990	0.0012210
channelmiscellaneous	0.0009610	0.0003843	0.0012210
num_videos	0.0007807	0.0001481	0.0006105
kw_min_avg1	0.0003592	0.0003242	0.0006105

lasso

The best lambda was 0.03162278.

```
set.seed(820)
lambda_seq <- 10^seq(2, -2, by = -.1)
cv_output <- cv.glmnet(new_tr, labels,
            alpha = 1, lambda = lambda_seq, type.measure = "mse", nfold = 10)

# identifying best lamda
best_lam <- cv_output$lambda.min
```

In this part, we are going to train the lasso model on the train dataset of `no_content` data with the best lambda.

```
best_lam  = 0.03162278
lasso.fit = glmnet(new_tr, labels,  alpha = 1, lambda = 0.03162278)
pred <- predict(lasso.fit, s = best_lam, newx = new_ts)
rmse_lasso_no_content = rmse(actual = test_no_content$shares, predicted = pred)
print("The rmse on the test dataset of no_content data is:")
```

```
## [1] "The rmse on the test dataset of no_content data is:"
```

```
print(rmse_lasso_no_content)
```

```
## [1] 1.114848
```

```
mae_lasso_no_content = mae(actual = test_no_content$shares, predicted = pred)
print("The mae on the test dataset of no_content data is:")
```

```
## [1] "The mae on the test dataset of no_content data is:"
```

```
print(mae_lasso_no_content)
```

```
## [1] 0.8620914
```

Feature importance

```
coef_lasso_no_content = coef(lasso.fit)
coef_lasso_noContent =data.frame(names = rownames(as.matrix(coef_lasso_no_content)), values = as.matrix
positive_no_content = as.character(coef_lasso_noContent[ coef_lasso_noContent$values>0, 'names'])
negative_no_content = as.character(coef_lasso_noContent[ coef_lasso_noContent$values<0, 'names'])

print(coef_lasso_no_content)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)               7.073919e+00
```

```
## n_tokens_title            6.619506e-03
## num_imgs                  .
## num_videos                .
## num_keywords              .
## kw_max_min                1.583499e-02
## kw_avg_min                9.010745e-03
## kw_min_max                .
## kw_avg_max                .
## kw_min_avg0               4.348011e-02
## kw_min_avg1              -2.995006e-15
## kw_max_avg                .
## kw_avg_avg                9.102959e-05
## LDA_00                    8.410036e-02
## LDA_01                   -1.078070e-01
## LDA_02                    .
## LDA_03                    .
## title_subjectivity        .
## title_sentiment_polarity  .
## channelentertainment     -2.113612e-01
## channellifestyle          4.795474e-01
## channelmiscellaneous      .
## channelsocmed             6.141972e-02
## channeltech               .
## channelworld              .
## weekdayMonday             3.065070e-03
## weekdaySaturday           3.284460e-02
## weekdaySunday             2.263522e-01
## weekdayThursday           .
## weekdayTuesday            .
## weekdayWednesday         -8.945620e-02
```

References

[1] https://mashable.com/

[2] Fernandes, Kelwin. (2015). A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News.

[3] https://www.kaggle.com/shivamsarawagi/classificationendexam

[4] https://rstudio-pubs-static.s3.amazonaws.com/157053_f1c97373ae8e4240b9686d2bc6bbc08c.html

[5] https://openscholarship.wustl.edu/cgi/viewcontent.cgi?article=2097&context=art_sci_etds

[6] https://www.kaggle.com/srikaranelakurthy/predicting-shares