Mashable: Online News Popularity

Gabriel Gonzales, Sachin Chavan, Shane Weinstock

MSDS 6372-404 Project II

12/08/2019

Introduction

Through this dataset we aimed to predict the popularity of a news article. The dataset provides a number of variables and an abundance of observations (nearly 40,000) that we are able to evaluate. The task of evaluating popularity becomes extremely complex very quickly; so using a number of variables, our intent was to create a model that would accurately provide us with more insight on what essentially makes a news article become popular. Popularity in this situation was determined by the number of shares that were observed from each news article found and hosted on Mashable.com.

Data Description

The data utilized for this project can be found on UCI's Center for Machine Learning and Intelligent Systems. The original dataset contains 39644 observations with 61 attributes to determine the popularity of a news article. To better define what "popular" versus "non-popular", it was decided to take the mean of the attribute 'shares' (defined below). The value of mean 'shares' and the cutoff to become a popular news article was 1400 'shares'. For this project we will be utilizing 53 variables contained in two of the seven data channels within the given dataset. The two selected data channels were World and Technology. These were selected due to the number of observations contained within them (being the two largest data channels in the dataset). From there we will also be taking a sample of the observations due to hardware restrictions on our personal machines.

1. URL of the article (non-predictive)
2. Days between the article publication and the dataset acquisition (non-predictive)
3. Number of words in the title
4. Number of words in the content
5. Rate of unique words in the content
6. Rate of non-stop words in the content
7. Rate of unique non-stop words in the content
8. Number of links
9. Number of links to other articles published by Mashable
10. num_imgs: Number of images
11. num_videos: Number of videos
12. Average length of the words in the content
13. Number of keywords in the metadata
14. Is data channel 'Lifestyle'?
15. Is data channel 'Entertainment'?
16. Is data channel 'Business'?
17. Is data channel 'Social Media'?
18. Is data channel 'Tech'?
19. Is data channel 'World'?
20. Worst keyword (min. shares)
21. Worst keyword (max. shares)
22. Worst keyword (avg. shares)
23. Best keyword (min. shares)
24. Best keyword (max. shares)
25. Best keyword (avg. shares)
26. Avg. keyword (min. shares)
27. Avg. keyword (max. shares)
28. Avg. keyword (avg. shares)
29. Min. shares of referenced articles in Mashable
30. Max. shares of referenced articles in Mashable
31. Avg. shares of referenced articles in Mashable
32. Was the article published on a Monday?
33. Was the article published on a Tuesday?
34. Was the article published on a Wednesday?
35. Was the article published on a Thursday?
36. Was the article published on a Friday?
37. Was the article published on a Saturday?
38. Was the article published on a Sunday?
39. Was the article published on the weekend?
40. Closeness to LDA topic 0
41. Closeness to LDA topic 1
42. Closeness to LDA topic 2

43. Closeness to LDA topic 3
44. Closeness to LDA topic 4
45. Text subjectivity
46. Text sentiment polarity
47. Rate of positive words in the content
48. Rate of negative words in the content
49. Rate of positive words among non-neutral tokens
50. Rate of negative words among non-neutral tokens
51. Avg. polarity of positive words
52. Min. polarity of positive words
53. Max. polarity of positive words
54. Avg. polarity of negative words
55. Min. polarity of negative words
56. Max. polarity of negative words
57. Title subjectivity
58. Title polarity
59. Absolute subjectivity level
60. Absolute polarity level
61. Number of shares (target)

More information can be found at UCI Datasets: Online News Popularity.

Exploratory Data Analysis (EDA)

As we began to dive into the variables of what creates a popular news article, we wanted to explore the attributes that we were given. It was clear that there were several attributes that were included that observed the same objective from a different angle. To absolve some of these issues we used the Data Explorer package in R. Here the syntax ‘introduction’ can be found where all of our data is expressed. This is also called again after we obtain our reduced sample. At this time we realized that there was a large quantity of negative numbers contained in the dataset. Seeing as many of them were correlated as well, a decision was made to remove any attribute which held fifty percent or more negative values (this is to say any attribute that contained 19,822 of 39,644 negative observations). This aided in simplifying the remaining data for future accuracy and observations as well as transformations and selection methods.
At first we wanted to ensure that we were not missing any values throughout the dataset. Fortunately in this dataset, nothing was missing and everything had a value assigned in each observation for each attribute. This can be observed in Figure 1 with a representational (0%) across all attributes. It was important to establish this because if there were missing values the accuracy of our tests and how the sample represents the data, the statistical power, the introduction of bias into the models, and also the estimation of parameters. Of the original 61 attributes, 8 were removed due to uneven negative observations. In terms of our attributes there are 53 that were selected.

Diving deeper into the continuous variables, a scatterplot matrix (Figure 2.) was used to help identify multicollinearity between variables. This is a very important step for logistic regression as it is one of the primary assumptions which must be met in order to have a working model. Also, it is important that we do not use variables which are highly correlated to one another as they will greatly affect the fit of our models. From the scatterplot, we determined several correlations existed between certain variables and therefore, were subsequently removed from the future models. The next crucial step in our data exploration was to observe the distributions of continuous variables in order to determine if a transformation would be required. Our variables have been expressed in the form of histograms in Figures 3a and 3b. To better examine the variable we also developed some boxplots in Figures 4a and 4b. There are also Q-Q plots and density plots located in Figures 5a,5b and Figures 6a, 6b, respectively. The last step that was conducted in the EDA for this dataset was a correlation plot in Figure 7. This was performed to establish if multicollinearity was an issue amongst the continuous variables and also to establish the correlation of the attributes to the response (popular)..

Regression Analysis: Logistic Regression Models Introduction

In the regression analysis, it was observed that our response attribute is shares. This was not categorical, so instead (as specified above) a new attribute was created, popular, in order to establish a binary categorical attribute for the response. The objective with this logistic regression model was to predict whether or not a news article would be popular given the attributes in the dataset. The predictors that were selected were those that are highly correlated with the response, popular. This response is intended to provide an accurate threshold of combined attributes which will yield a popular or not popular news article. The size of the data was reduced for these analyses due to limited computational power of our local machines. The sample was selected from Technology which contains 7346 observations and World which contains 8427 observations. A seed was also placed to ensure that we received the same random sample and the model could be reproduced. To handle the unbalanced dataset we used SMOTE, a package in R. SMOTE is an algorithm that combats unbalanced classification problems by generating new examples of the minority class. SMOTE can also be set with a seed so that the previously randomized balancing is now reproducible as well. Several variable selection methods were attempted in order to build successful models, however backwards selection produced the best model.

Model Selection

Objective 1

Building Logistic Regression Model Using World and Second on Technology data channels.

In the process of our EDA, there are several continuous variables that had multiple peaks, had left-skewed distributions, and some normal distributions. While using the backwards selection method, and only using the World data channel, the following continuous variables were discovered to be significant: global_subjectivity, num_self_hrefs, rate_positive_words, average_token_length, num_hrefs, num_imgs, n_tokens_title, n_unique_tokens, LDA_04, kw_min_max, kw_max_max, kw_avg_max, kw_max_avg, kw_avg_avg. From these variables an AIC of 12,559. (Figure 8.) As the data had been split between World and Technology, we then ran a logistic regression for the Technology data channel as well. The continuous variables that were found significant in that data channel were as follows: global_subjectivity, num_self_hrefs, rate_positive_words, num_hrefs, n_tokens_title, n_unique_tokens, rate_negative_words, keyword_max_avg, keword_avg_avg. The AIC for the technology data channel model was 12,834.(Figure 9.)

Building the Logistic Regression Model: Final Model

After analyzing the model returned from the backwards selection method, we found that a number of variables that are included in the dataset were observing the same attributes of the data. To address this, we ran PCA to better attain what variables we should proceed with in order to build our model. The following model was constructed after addressing more acutely the collinearity within the dataset and also the redundancy in attributes. From this, we aimed to build a logistic regression model based on certain parameters and also utilized a more complex selection method, LASSO: number of words in the title, the number of words in the article, number hyperlinks, images or videos etc to predict whether online article will become popular or not. The final model contains the following parameters: n_non_stop_words, average_token_length, n_non_stop_unique_tokens, n_unique_tokens, global_subjectivity, avg_positive_polarity, rate_positive_words, kw_avg_avg, global_rate_positive_words, LDA_02, self_reference_avg_shares. This model returned an AIC of 12555.

Objective 2

Building the Logistic Regression Model: Final Model

For this model we conducted another logistic regression model around variables we observed in the EDA and previous models. We felt the following variables would provide us better insight as to how these would predict our target response, popular. The variables are as follows: rate positive words, rate negative words, number of images and title subjectivity. We placed an interaction term on rate positive words by utilizing rate negative words. Although at surface value these variables would seem to contradict one another and be highly correlated, they actually measure in the form of a count for the number of each type of word in the context of a post as defined by our source at UCI. This model returned an AIC of 7936.3 with the same amount of data as our previous models.

Checking Assumptions

Progressing with the final model from above, the collinearity of the model needed to be evaluated. This was accomplished by utilizing the VIF function (established in the car package). The VIF values for these variables in our model are located in Figure 10. After addressing these variables, it is safe to conclude that there is no multicollinearity in the final model. Following this, the residuals were plotted in Figure 11. At this point influential points that may have existed in the final model through the Cook's D plot in Figure 12, where no influential points were identified. Lastly, a Goodness of Fit test was run where the Hosmer/Lemeshow Goodness of Fit test was utilized. This was conducted to test the following hypothesis:

Ho: Our final logistic model is a good fit for the data.

Ha: Our final logistic model is not a good fit for the data.

From this test one will be able to observe whether a model best fits our given data and therefore will determine the prediction power and accuracy of the model selected. The model was run on a test given alpha of 0.05, and returned a value of 2.416e-08. As this model succeeded, we wanted to check the accuracy in which it would predict the response. In order to evaluate this a confusion matrix was constructed (found in figure). We measured the accuracy for the two data channels were Technology returned 56.2% and World returned 60.83%. This model functions, however it really doesn't suffice for determining the popularity of a news article posted at Mashable.com.

Parameter Interpretation

By using the regression coefficients of the final model, we were able to provide some interpretations of the continuous variables. These can also be noted in Table 1a and Table 1b found in the appendix.

- The regression coefficient for the rate of non-stop words in the content of an article is 2.0093. A 95% confidence interval for this regression coefficient is (0.8419404, 3.178715). The odds ratio for popularity is Exp(2.0093)=7.458 This means that the odds of a popular post are 745.8% higher when the rate of non-stop words in the article content increase by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (2.32086, 24.0158).
- The regression coefficient for the average length of the words in the content of an article is -0.58675. A 95% confidence interval for this regression coefficient is (-0.7863616, -0.3876529). The odds ratio for this variable is Exp(-0.58675)=0.5561 This means that the odds of a popular article are 55.61% higher as the average length of the words in the content increases by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (0.45549, 0.67864).
- The regression coefficient for the rate of unique non-stop words in the content of an article is -3.688. A 95% confidence interval for this regression coefficient is (-4.94001, -2.44128). The odds ratio for this variable is Exp(-3.688)=0.02501 This means that the odds of a popular article are 2.501% lower as the rate of unique non-stop words in the content increases by one, (holding all variables constant). A 95% confidence interval for this odds ratio is (0.00715, 0.08704).
- The regression coefficient for the rate of unique words in the content of an article is 2.8139. A 95% confidence interval for this regression coefficient is (1.55547, 4.076201). The odds ratio for this variable is Exp(2.8139)=16.676 This means that the odds of a popular article are 1667.6% higher as the rate of unique words in the content of an article increases by one, (holding all variables constant). A 95% confidence interval for this odds ratio is (4.7373, 58.9212).
- The regression coefficient for the text subjectivity of an article is 1.1242. A 95% confidence interval for this regression coefficient is (0.511449, 1.738362). The odds ratio for this variable is Exp(1.1242)=3.0779 This means that the odds of an article becoming popular are 307.79% higher as the text subjectivity of an article increases by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (1.6677, 5.688).

- The regression coefficient for the average polarity of positive words of an article is 0.4902. A 95% confidence interval for this regression coefficient is (-0.119467, 1.10093). The odds ratio for this variable is Exp(0.4902)=1.6328 This means that the odds of an article becoming popular are 163.28% higher as the average polarity of positive words in an article increases by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (0.88739, 3.0069).
- The regression coefficient for the rate of positive words among non-neutral tokens of an article is 0.4981. A 95% confidence interval for this regression coefficient is (0.128479, 0.86823). The odds ratio for this variable is Exp(0.4981)=1.6456 This means that the odds of an article becoming popular increase by 164.56% as the rate of positive words among non-neutral tokens within an article increase by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (1.13709, 2.3827).
- The regression coefficient for average number of keywords in relation to the average number of shares of an article is 0.00024. A 95% confidence interval for this regression coefficient is (0.0001798, 0.0003058). The odds ratio for this variable is Exp(0.00024)=1.000242 This means that the odds of an article becoming popular is 100% more likely as the average number of keywords in relation to the average number of shares increases by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (1.0001, 1.0003).
- The regression coefficient for the rate of positive words in the content of an article is 6.3874. A 95% confidence interval for this regression coefficient is (2.40805, 11.28207). The odds ratio for this variable is Exp(6.3874)=933.317 This means that the odds of n article becoming popular are 93,317% greater as the rate of positive words in the content of an article increases by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (11.28207, 79385.96).
- The regression coefficient for the closeness to LDA topic 2 of an article is -0.89815. A 95% confidence interval for this regression coefficient is (-1.11062, -0.686216). The odds ratio for this variable is Exp(-0.89815)=0.4073 This means that the odds of an article becoming popular are 40.73% more likely as the closeness to the LDA topic 2 (defined in the dataset article) increases by one unit, (holding all variables constant). A 95% confidence interval for this odds ratio is (0.32935, 0.50347).
- The regression coefficient for the average shares of referenced articles in Mashable in relation to an article is 0.000001. A 95% confidence interval for this regression coefficient is (0.000005, 0.0000164). The odds ratio for this variable is Exp(0.000001)=1.00001. This means that the odds of an article becoming popular are 100% more likely given the average shares of referenced articles in Mashable increases by one, (holding all variables constant). A 95% confidence interval for this odds ratio is (1.000005906, 1.00001640).

Logistic Regression Models: Conclusion

The first model that was constructed provided insight to what attributes would most likely point us towards a popular or not popular news article. No attribute was removed specifically in the first model as it allowed for the clarity of which attributes would be most significant, even if they were observing like data. Due to the size of the data, we were forced to use a backwards selection method, which presented a decently successful model, but not the best model. From there it was important to address the number of attributes by their correlations and collinearity before entering the next model. This was conducted with LASSO and PCA. From this new set of attributes a new model was run where LASSO became the new variable selection method. This is how we arrived at the final model containing: number of non stop words, average token length, number of non stop unique tokens, number of unique tokens, global subjectivity, average positive polarity, rate of positive words, the average of keywords, the global rate of positive words, LDA2, and the self referenced average number of shares. All of these variables had real world weight in determining whether a news article posted to Mashable.com would be popular or not. This study is an observational study and therefore inferences can only be drawn on the 39,644 news articles included in the dataset.

PCA

We used PCA for feature selection. For this dataset first 19 components contributes total 80% of variation (Fig-PCA-1). We select variables which contributes above expected average (cut-off) to all these 19 components.(Fig-PCA-2)

LDA

We attempted to perform an LDA, however due to normality constraints, LDA was not successful. We had too many variables given the amount of computational power of our personal machines. Even after transformations were made to normalize the data and reductions of the dataset were made, we felt we had violated too many of the assumptions for this to be a reliable metric. LDA is supposed to determine group means and a computation for all of the data in regards to the probability of belonging to various or different groups. The assumption that we knew was violated was the equality of covariances among the predictors across all levels of the response.

Building kNN Model

As a supervised machine learning algorithm, we were able to bypass some of the assumptions required by other algorithms. It takes a set of predictors and the binary label given by the dataset, in this case popular or not popular, and evaluates the remaining attributes based on their closeness (as similar things are likely to be closer to one another). In our model we defined k=5, and discovered that this was our optimal model. To take that a step further, we established the distance between the query and examples contained in our dataset by selected the 5 specific examples closest to the query. This was beneficial to us as the models we implemented through regression did not hold very good accuracy and we couldn’t meet additional assumptions as discussed above. This model returned an accuracy of 71.65% with a specificity of 82.24% and sensitivity was 61.06%.

Building Random Forest Model

This model is a tree based algorithm involving the construction of several trees and then proceeding to combine the output of those trees to improve the generalization capabilities of the model. The popularity data contains a variable number of shares which we derived our own variable from, popular, in order to create binary classification and a target. This denotes whether an article will be popular or not as discussed earlier in this paper as well. The dataset is then split in half for training and test. The Random Forest algorithm is run on the training set, given a number of trees to grow, in our instanced model, 500 and the number of variables sampled as candidates were split at 5. No observations were excluded in this model, only variables or attributes that were deemed to be collinear from earlier EDA. This was the only method that was able to process the entire dataset, giving us the most conclusive prediction model. The model returned an accuracy of 57.31% and holds a 95% confidence interval of (0.5682, 0.578). It's sensitivity returned 67.96% and specificity was 46.95%. The model also generated an OOB estimate error rate of 46%.These metrics can be found in Figure 13.

If we observe the model output, we find that of the variables included, the average number of keywords across the average number of shares continues to be the highest contributing variable to an article's popularity on Mashable.com. On the opposing end, it's intriguing to see that average polarity of negative words contributed the least to the prediction of an article's popularity, providing next to no informative gain.

Conclusion

From this study we were able to develop three classification models. First on the two data channels, World and Technology, which were the two largest categories of data. From there we proceeded to reduce the next model with PCA, as we had a large number of attributes that were depicting the same components of the news articles captured. The final model was constructed for logistic regression and we then compared this model to a kNN instance of the same data. It wasn't until we arrived at constructing a Random Forest that we were able to utilize the all observations in our dataset for our analysis, after again having removed correlated variables. Although we were able to produce several models, there were tradeoffs to each of them. Some were more statistically significant while others were much more predictive. The highest prediction values that we were able to obtain were through kNN, although the rest of the metrics were not as favorable. Upon completion of all of the models run above, we would favor the Random Forest, as it included the greatest number of observations and the majority of the attributes contained in the dataset.

For future considerations, we would have liked to have the capability to handle the large amount of observations within this dataset. There are extremely interesting data channels that are contained and an organization of data that could present very intriguing models. However, our personal machines were simply not up to the task of running them. Of course with our timeline and given resources we managed to develop significant models, however with the data provided we feel it can go a lot further. It would be interesting to see what the other data channels can offer and what additional sampling may provide.

Appendix

Plots and Tables

Exploratory Data Analysis (EDA)

Figure 1. Removal of Missing Values

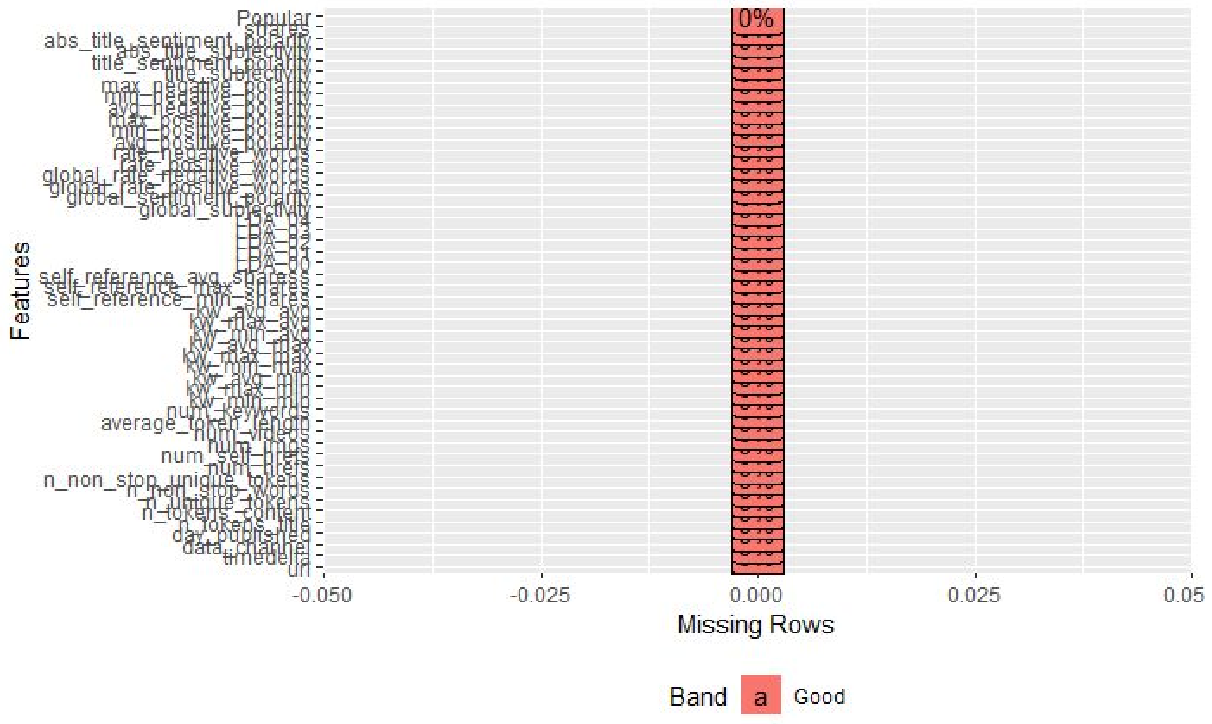


Figure 2. Scatterplot Matrix

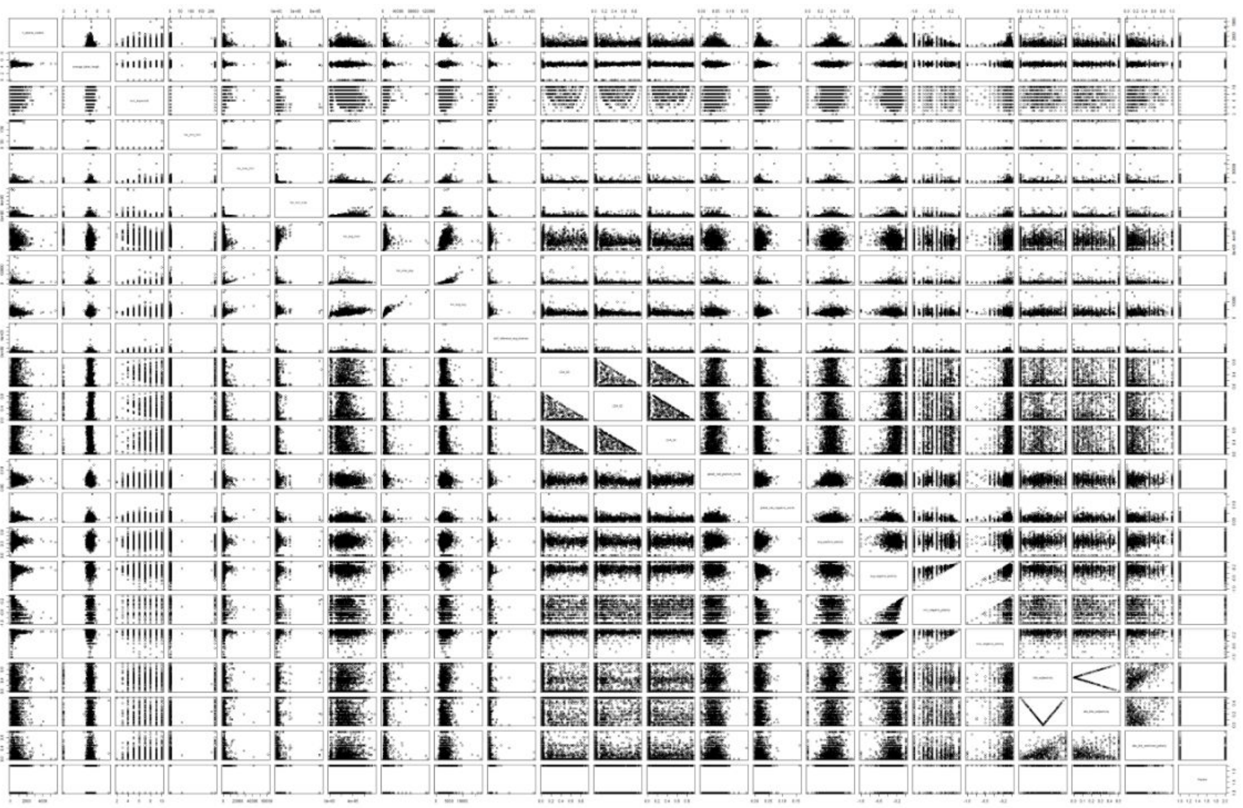


Fig 3a. Histograms

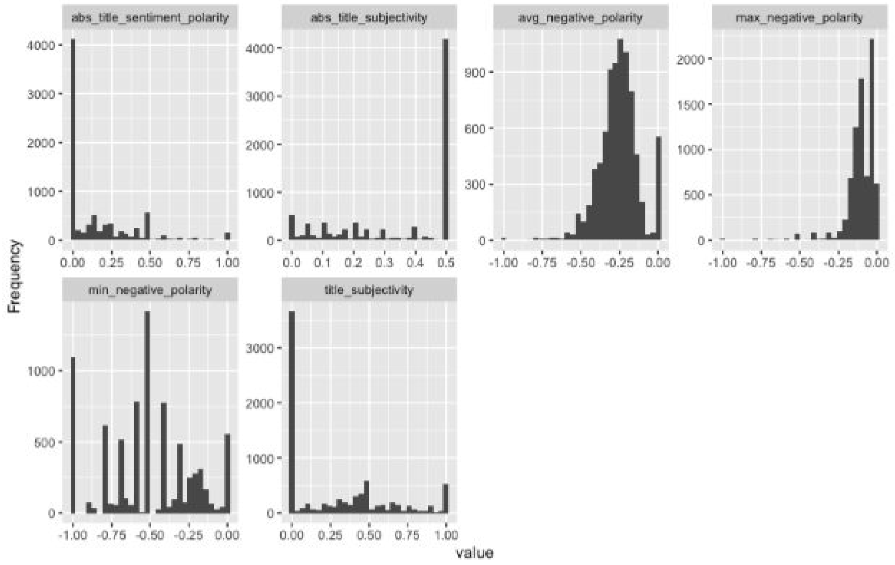


Fig 3b. Histograms

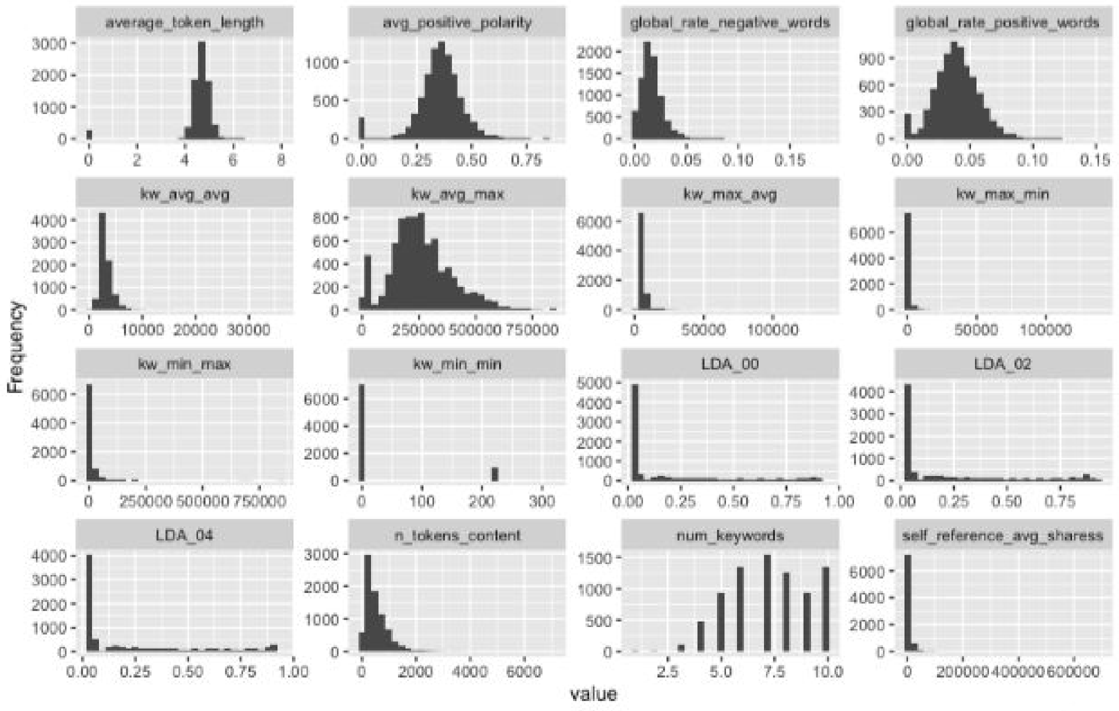


Figure 4a Boxplots

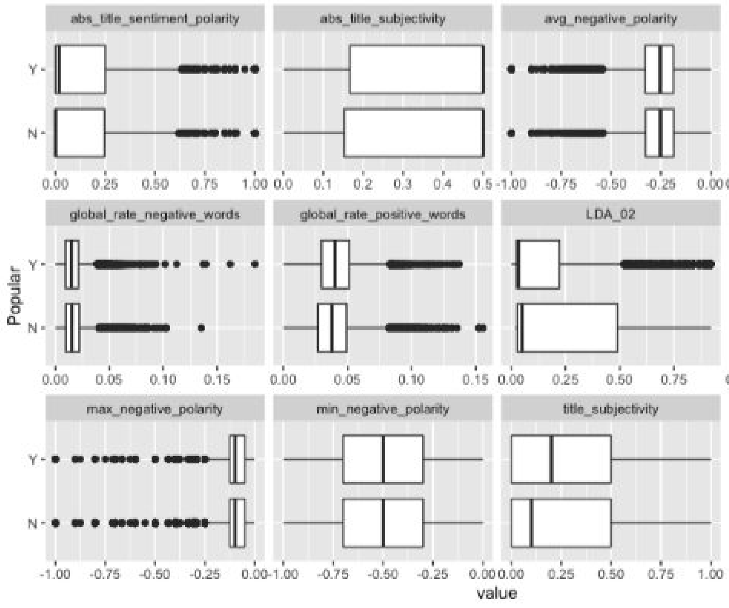


Figure 4b Boxplots

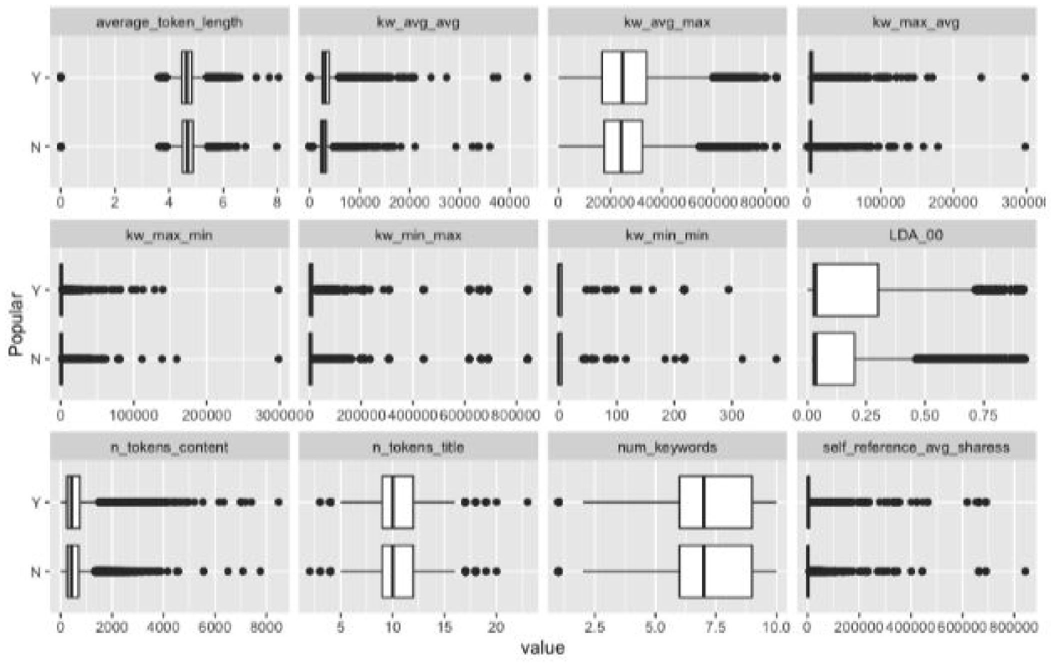


Figure 5a. Q-Q plots

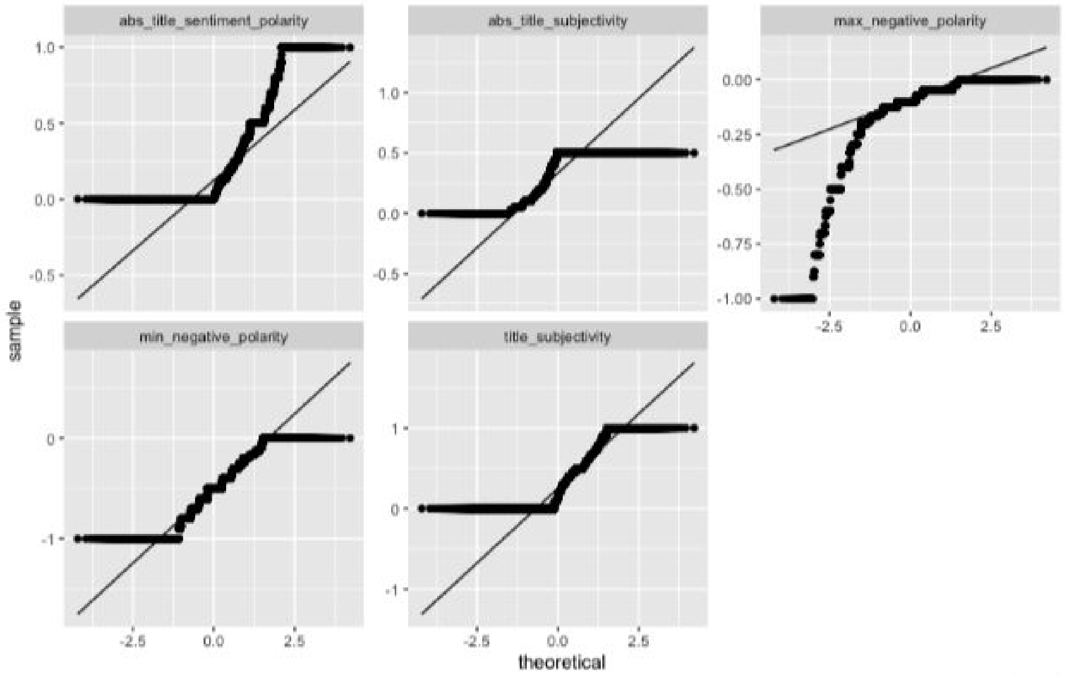


Figure 5b. Q-Q plots

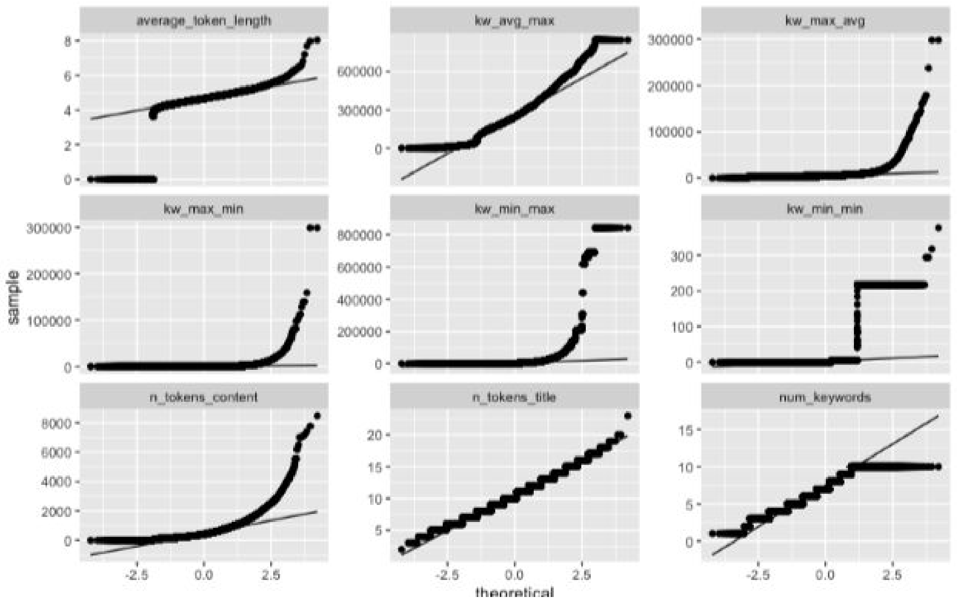


Figure 6a. Density plots

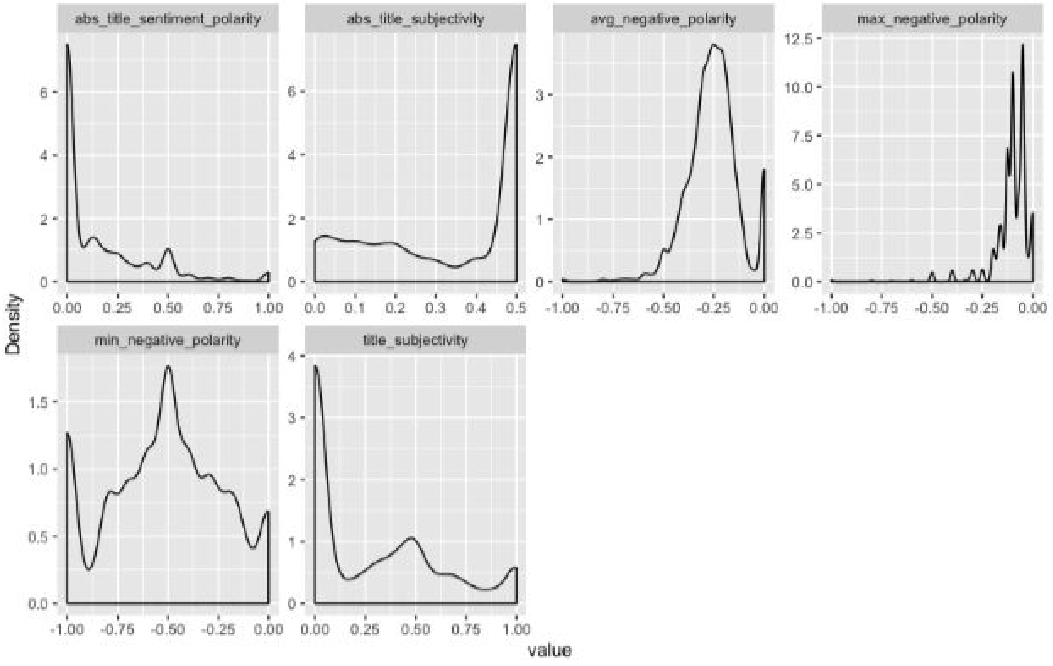


Figure 6b. Density plots

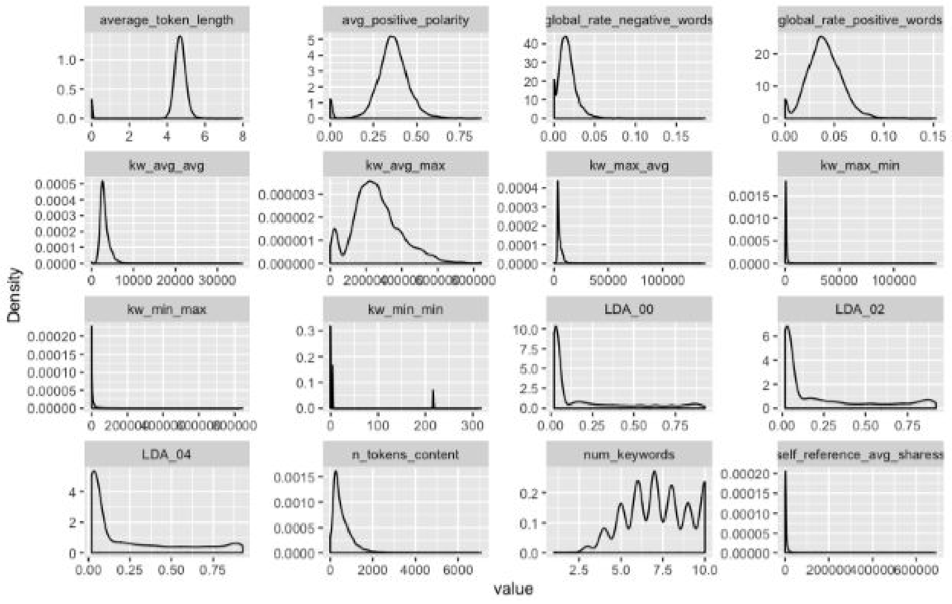
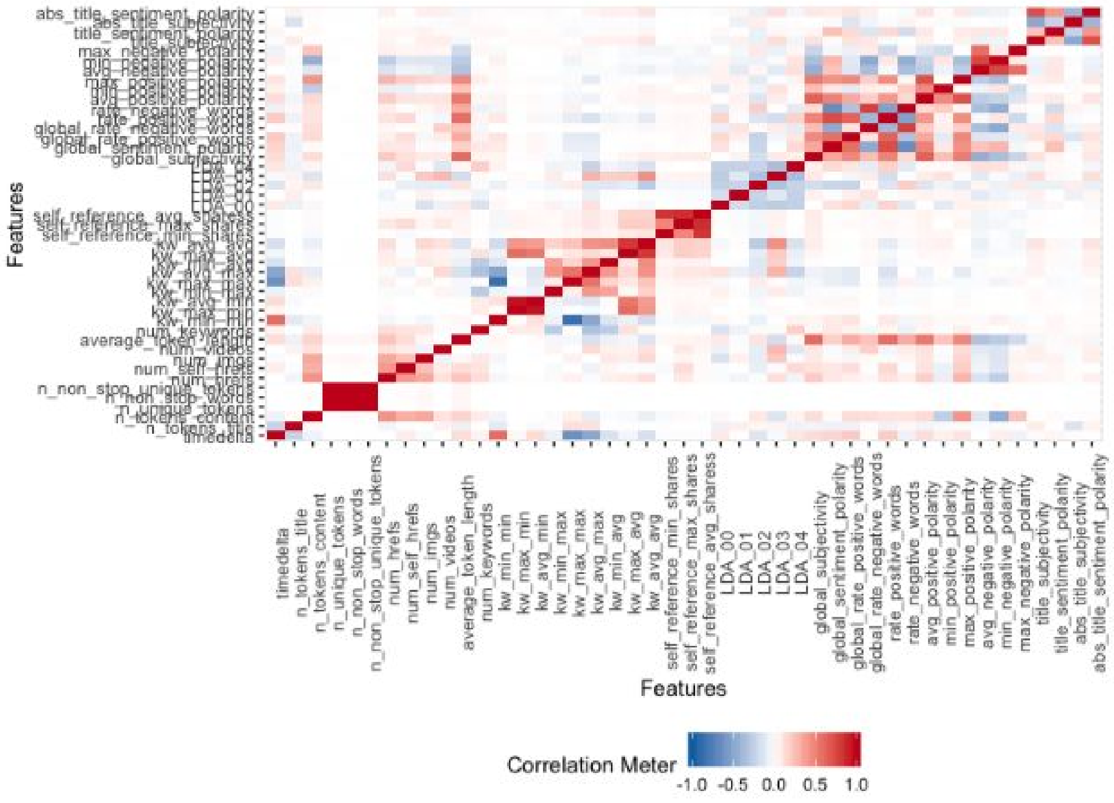
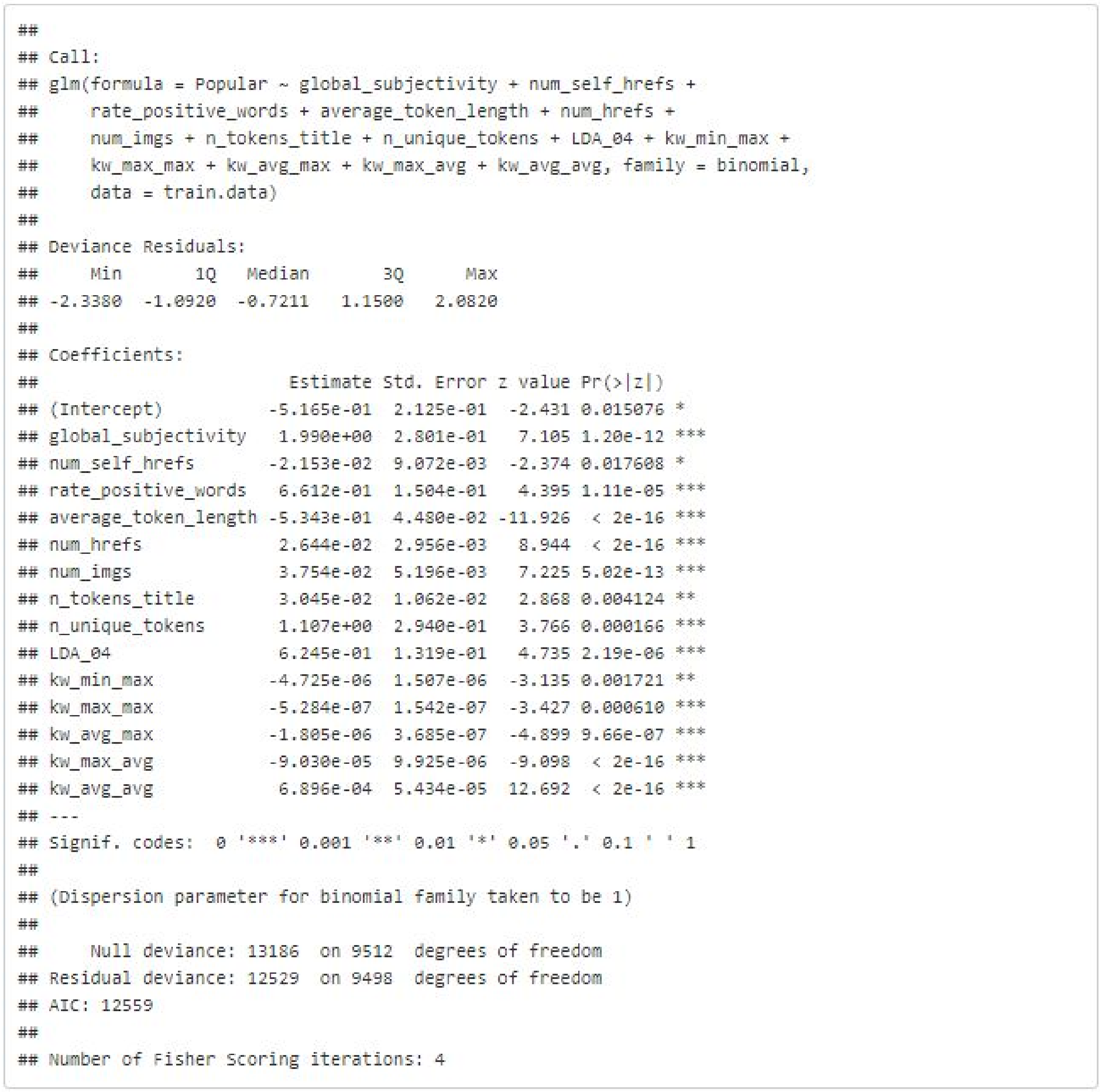


Figure 7. Correlation Plot



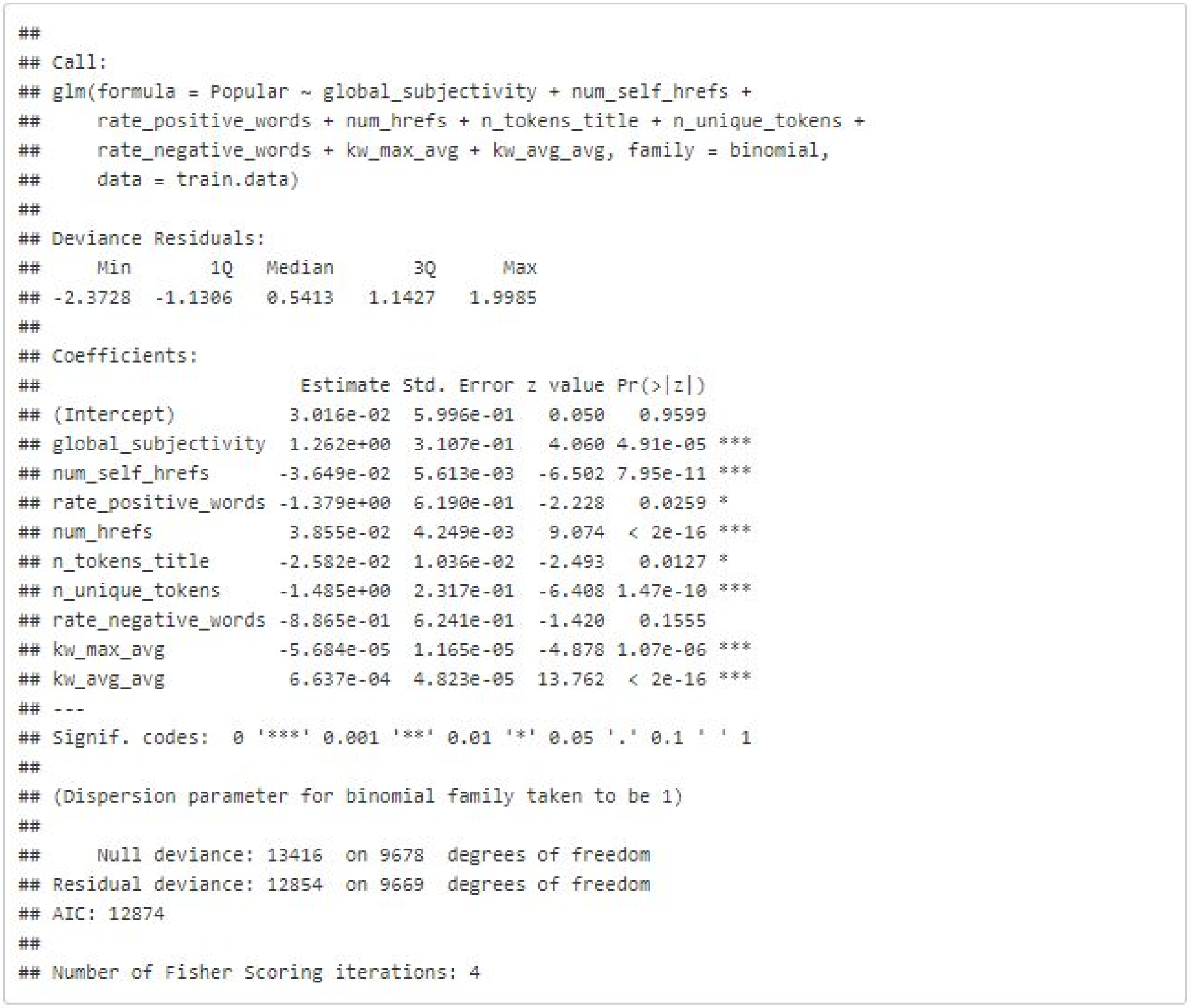
Objective 1 - Logistic Regression

Figure 8. World Data Channel



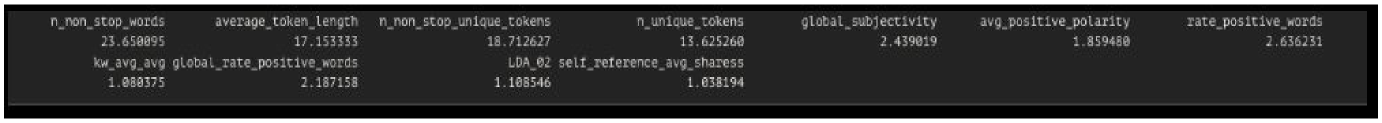
```
##
## Call:
## glm(formula = Popular ~ global_subjectivity + num_self_hrefs +
##     rate_positive_words + average_token_length + num_hrefs +
##     num_imgs + n_tokens_title + n_unique_tokens + LDA_04 + kw_min_max +
##     kw_max_max + kw_avg_max + kw_max_avg + kw_avg_avg, family = binomial,
##     data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3380  -1.0920  -0.7211   1.1500   2.0820
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.165e-01  2.125e-01  -2.431 0.015076 *
## global_subjectivity   1.990e+00  2.801e-01   7.105 1.20e-12 ***
## num_self_hrefs       -2.153e-02  9.072e-03  -2.374 0.017608 *
## rate_positive_words   6.612e-01  1.504e-01   4.395 1.11e-05 ***
## average_token_length -5.343e-01  4.480e-02 -11.926  < 2e-16 ***
## num_hrefs             2.644e-02  2.956e-03   8.944  < 2e-16 ***
## num_imgs              3.754e-02  5.196e-03   7.225 5.02e-13 ***
## n_tokens_title        3.045e-02  1.062e-02   2.868 0.004124 **
## n_unique_tokens       1.107e+00  2.940e-01   3.766 0.000166 ***
## LDA_04                6.245e-01  1.319e-01   4.735 2.19e-06 ***
## kw_min_max           -4.725e-06  1.507e-06  -3.135 0.001721 **
## kw_max_max           -5.284e-07  1.542e-07  -3.427 0.000610 ***
## kw_avg_max           -1.805e-06  3.685e-07  -4.899 9.66e-07 ***
## kw_max_avg           -9.030e-05  9.925e-06  -9.098  < 2e-16 ***
## kw_avg_avg            6.896e-04  5.434e-05  12.692  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13186  on 9512  degrees of freedom
## Residual deviance: 12529  on 9498  degrees of freedom
## AIC: 12559
##
## Number of Fisher Scoring iterations: 4
```

Figure 9. Technology Data Channel



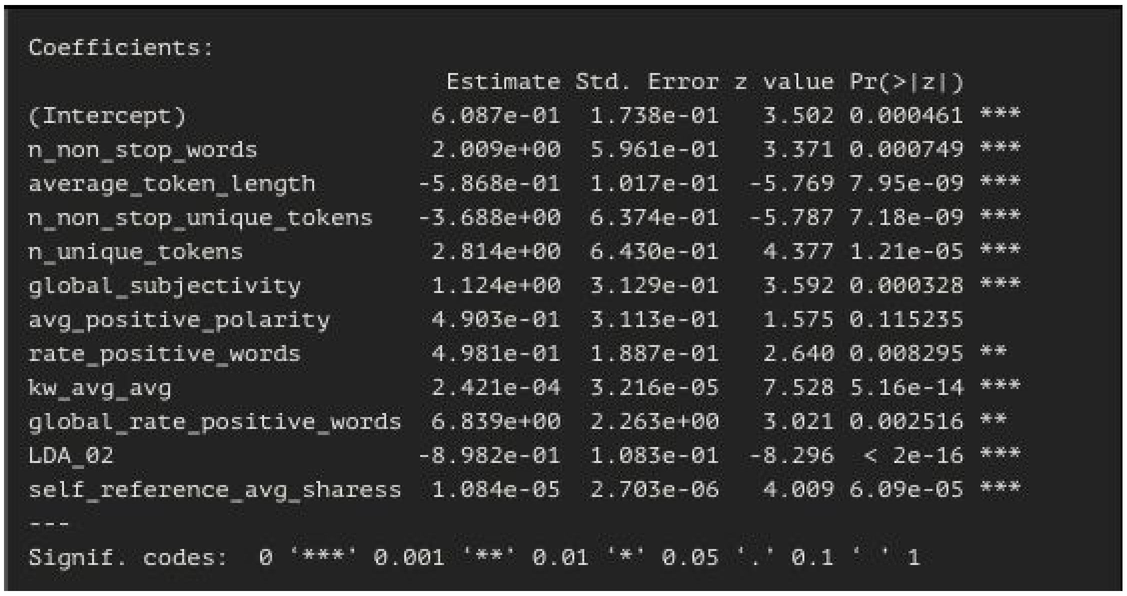
```
##
## Call:
## glm(formula = Popular ~ global_subjectivity + num_self_hrefs +
##     rate_positive_words + num_hrefs + n_tokens_title + n_unique_tokens +
##     rate_negative_words + kw_max_avg + kw_avg_avg, family = binomial,
##     data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3728  -1.1306   0.5413   1.1427   1.9985
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          3.016e-02  5.996e-01   0.050   0.9599
## global_subjectivity  1.262e+00  3.107e-01   4.060 4.91e-05 ***
## num_self_hrefs      -3.649e-02  5.613e-03  -6.502 7.95e-11 ***
## rate_positive_words -1.379e+00  6.190e-01  -2.228   0.0259 *
## num_hrefs            3.855e-02  4.249e-03   9.074  < 2e-16 ***
## n_tokens_title      -2.582e-02  1.036e-02  -2.493   0.0127 *
## n_unique_tokens     -1.485e+00  2.317e-01  -6.408 1.47e-10 ***
## rate_negative_words -8.865e-01  6.241e-01  -1.420   0.1555
## kw_max_avg          -5.684e-05  1.165e-05  -4.878 1.07e-06 ***
## kw_avg_avg           6.637e-04  4.823e-05  13.762  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13416  on 9678  degrees of freedom
## Residual deviance: 12854  on 9669  degrees of freedom
## AIC: 12874
##
## Number of Fisher Scoring iterations: 4
```

Figure 10. VIF Values



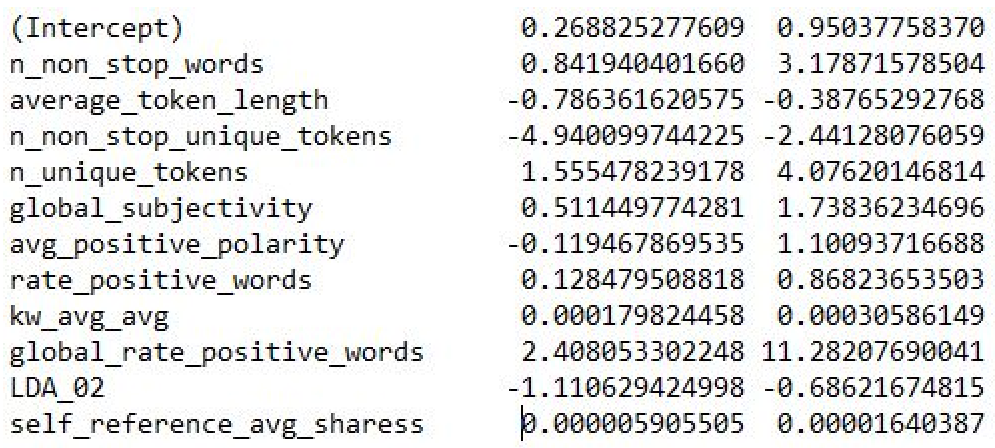
```
   n_non_stop_words        average_token_length n_non_stop_unique_tokens             n_unique_tokens         global_subjectivity        avg_positive_polarity         rate_positive_words
          23.650095                   17.153333                18.712627                   13.625260                    2.439019                     1.859480                    2.636231
         kw_avg_avg global_rate_positive_words                    LDA_02 self_reference_avg_sharess
           1.080375                    2.187158                 1.108546                    1.038194
```

Table 1a.



```
Coefficients:
                             Estimate Std. Error z value Pr(>|z|)    
(Intercept)                 6.087e-01  1.738e-01   3.502 0.000461 ***
n_non_stop_words            2.009e+00  5.961e-01   3.371 0.000749 ***
average_token_length       -5.868e-01  1.017e-01  -5.769 7.95e-09 ***
n_non_stop_unique_tokens   -3.688e+00  6.374e-01  -5.787 7.18e-09 ***
n_unique_tokens             2.814e+00  6.430e-01   4.377 1.21e-05 ***
global_subjectivity         1.124e+00  3.129e-01   3.592 0.000328 ***
avg_positive_polarity       4.903e-01  3.113e-01   1.575 0.115235    
rate_positive_words         4.981e-01  1.887e-01   2.640 0.008295 ** 
kw_avg_avg                  2.421e-04  3.216e-05   7.528 5.16e-14 ***
global_rate_positive_words  6.839e+00  2.263e+00   3.021 0.002516 ** 
LDA_02                     -8.982e-01  1.083e-01  -8.296  < 2e-16 ***
self_reference_avg_sharess  1.084e-05  2.703e-06   4.009 6.09e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Table 1b.



```
(Intercept)                    0.268825277609  0.95037758370
n_non_stop_words               0.841940401660  3.17871578504
average_token_length          -0.786361620575 -0.38765292768
n_non_stop_unique_tokens      -4.940099744225 -2.44128076059
n_unique_tokens                1.555478239178  4.07620146814
global_subjectivity            0.511449774281  1.73836234696
avg_positive_polarity         -0.119467869535  1.10093716688
rate_positive_words            0.128479508818  0.86823653503
kw_avg_avg                     0.000179824458  0.00030586149
global_rate_positive_words     2.408053302248 11.28207690041
LDA_02                        -1.110629424998 -0.68621674815
self_reference_avg_sharess     0.000005905505  0.00001640387
```

Figure 11. Residual Plots

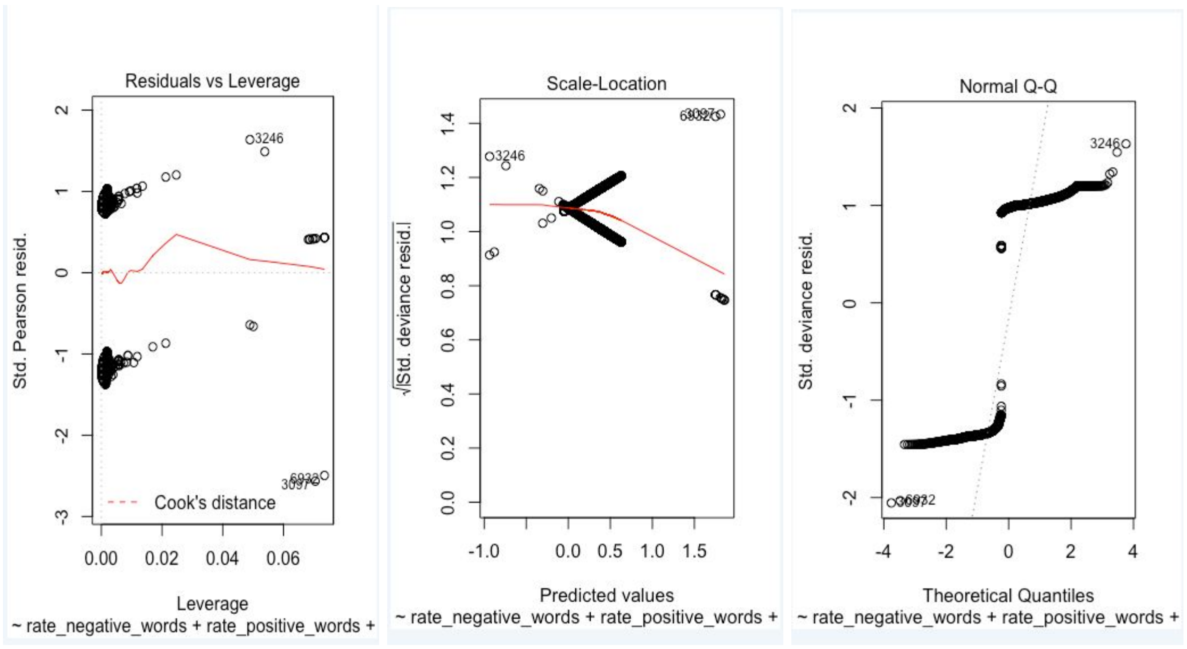


Figure 12. Cook's D

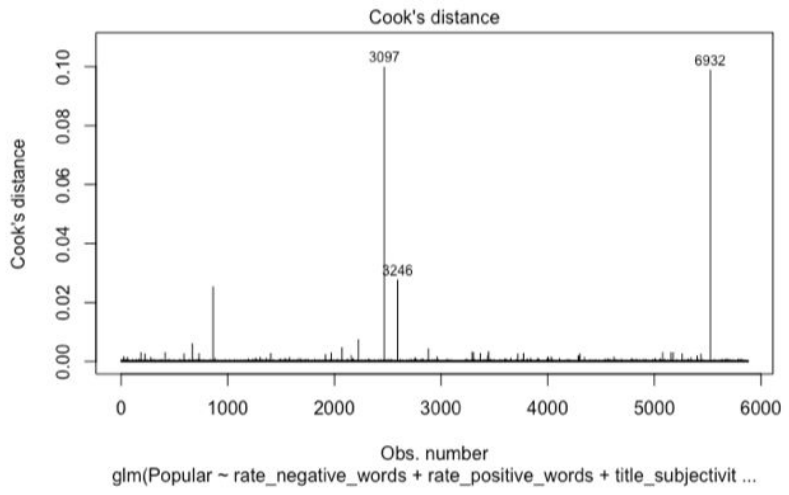
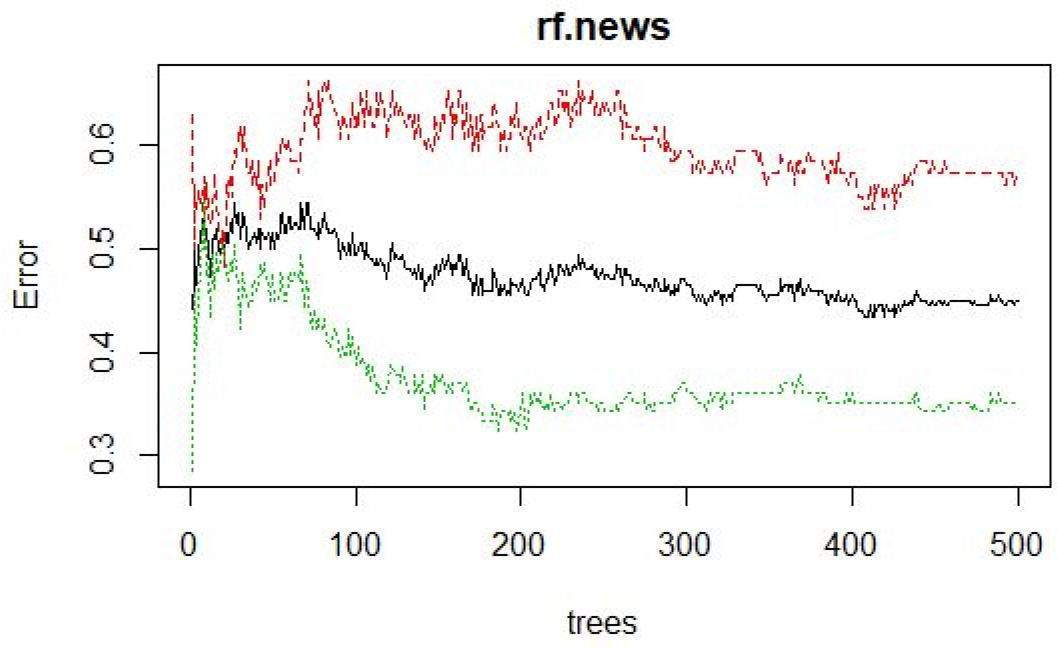


Figure 13. Random Forest Plot



Objective 2: Random Forest

Random Forest Output

```
Call:
 randomForest(formula = Popular ~ . - Popular, data = reducednews,      mtry = 5, importance =
T, ntree = 500, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 46%
Confusion matrix:
   Y  N class.error
Y 73 38   0.3423423
N 54 35   0.6067416
```

```
fit.pred     Y     N
       Y 13219 10606
       N  6232  9386
Confusion Matrix and Statistics


fit.pred     Y     N
       Y 13219 10606
       N  6232  9386

               Accuracy : 0.5731
                 95% CI : (0.5682, 0.578)
    No Information Rate : 0.5069
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1486

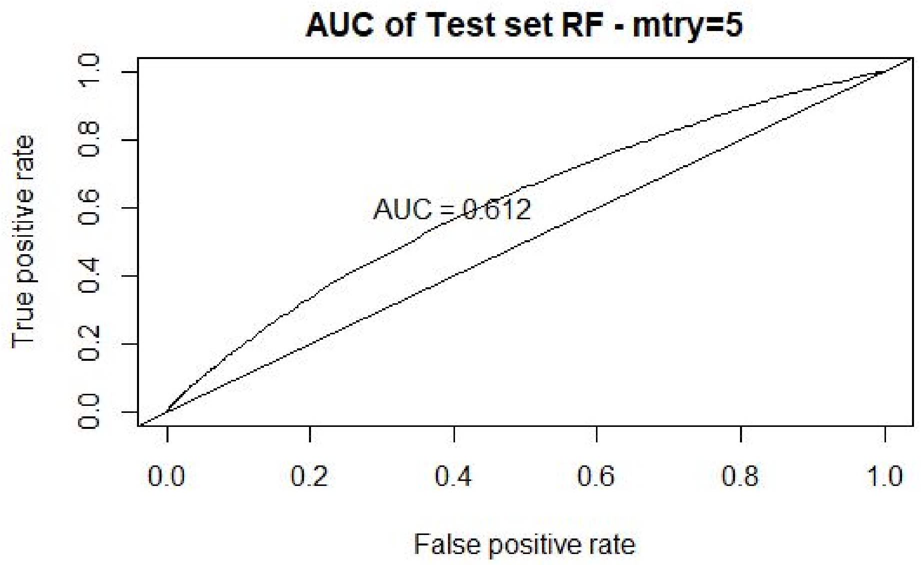
 Mcnemar's Test P-Value : < 2.2e-16

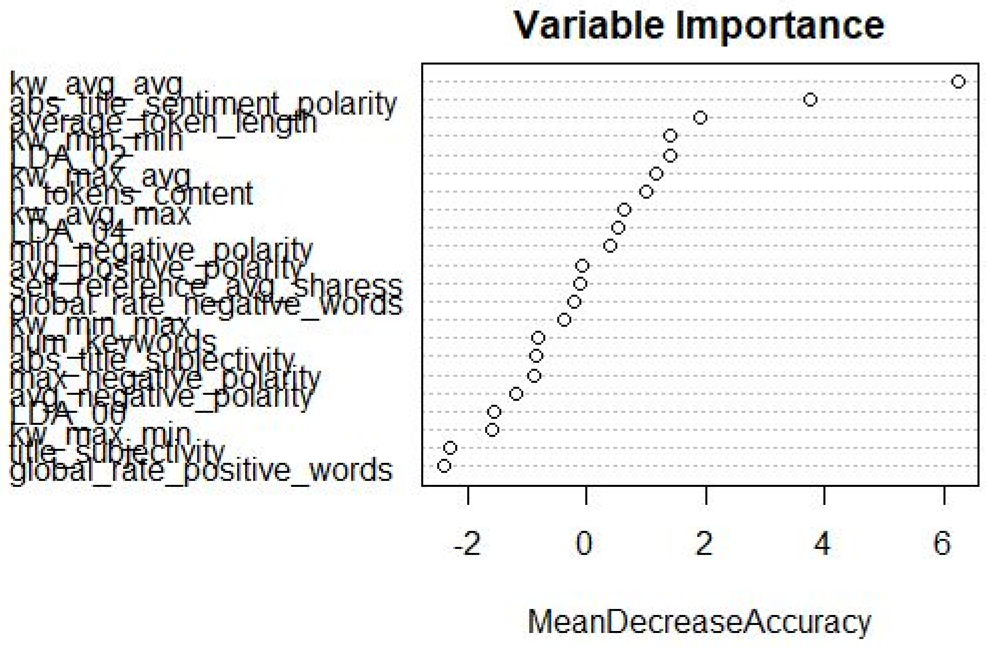
            Sensitivity : 0.6796
            Specificity : 0.4695
         Pos Pred Value : 0.5548
         Neg Pred Value : 0.6010
             Prevalence : 0.4931
         Detection Rate : 0.3351
   Detection Prevalence : 0.6040
      Balanced Accuracy : 0.5745

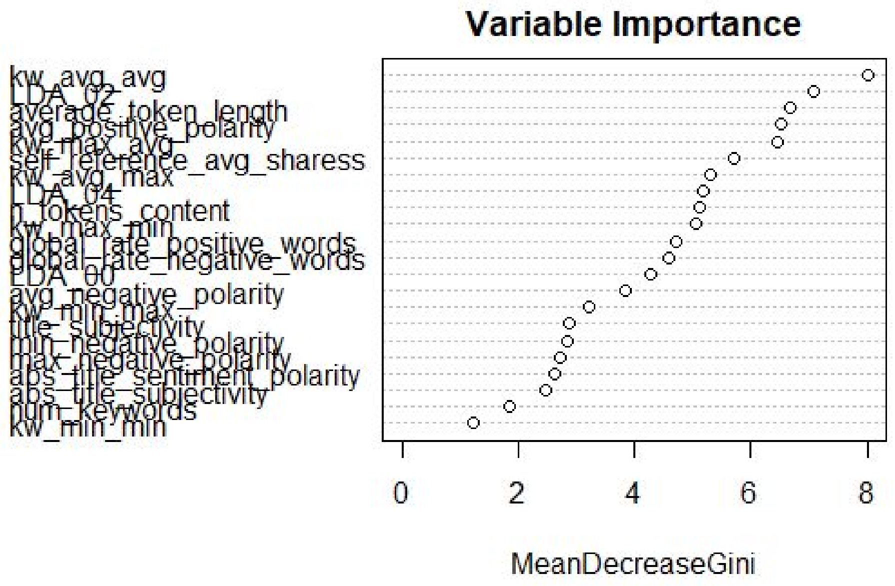
       'Positive' Class : Y

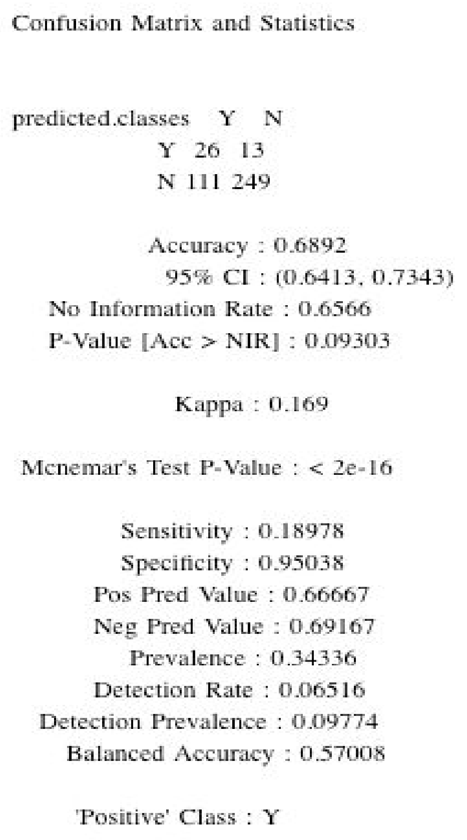

fit.pred     Y     N
       Y 13219 10606
       N  6232  9386
                                 Y          N MeanDecreaseAccuracy MeanDecreaseGini
n_tokens_content           0.4206742 -0.1413449           0.06677981         5.324460
average_token_length       1.9083714  1.9265045           2.68545618         6.906170
num_keywords              -0.8587508 -0.3579960          -1.01451540         1.938127
kw_min_min                 0.3938783 -1.0153229          -0.22547908         1.090064
kw_max_min                -0.5458004 -1.9281653          -1.71742672         4.974100
kw_min_max                 0.8131276 -0.1736117           0.48389764         3.396389
kw_avg_max                -3.0447231  2.4237552          -0.55417988         5.423542
```

```
kw_max_avg                -0.5483006  1.3744503          0.41424969        6.439017
kw_avg_avg                 2.1206196  5.7645738          4.93072494        7.971264
self_reference_avg_sharess  -0.1083007  0.6223889         0.24137150        5.468806
LDA_00                  -1.1140516 -2.2053562          -2.39426830        4.668529
LDA_02                   0.5461153  2.1549617           1.76501347        6.444353
LDA_04                  -1.4262871  0.5640949          -0.71747147        5.392215
global_rate_positive_words   -1.5826864 -0.8781719         -1.62310820        4.548526
global_rate_negative_words   -0.1628330 -1.8705474         -1.41474809        4.455819
avg_positive_polarity        1.6348476  1.9494387          2.57542923        6.327153
avg_negative_polarity       -2.9535198 -1.0388052          -2.97750904        3.984322
min_negative_polarity       -0.8131846  0.3063952          -0.37833849        2.655639
max_negative_polarity        1.7768591 -1.8945577          0.26213934        2.765568
title_subjectivity          -0.6326095  1.3616665          0.43670128        3.075586
abs_title_subjectivity       0.3874003 -0.1911154          0.20041286        2.541511
abs_title_sentiment_polarity -1.8160125 -0.7122361         -1.96913619        2.547158
```




Variable Importance
kw_avg_avg
abs_title_sentiment_polarity
average_token_length
kw_min_min
LDA_02
kw_max_avg
n_tokens_content
kw_avg_max
LDA_04
min_negative_polarity
avg_positive_polarity
self_reference_avg_sharess
global_rate_negative_words
kw_min_max
num_keywords
abs_title_subjectivity
max_negative_polarity
avg_negative_polarity
LDA_00
kw_max_min
title_subjectivity
global_rate_positive_words
-2
0
2
4
6
MeanDecreaseAccuracy


Variable Importance
kw_avg_avg
LDA_02
average_token_length
avg_positive_polarity
kw_max_avg
self_reference_avg_sharess
kw_avg_max
LDA_04
n_tokens_content
kw_max_min
global_rate_positive_words
global_rate_negative_words
LDA_00
avg_negative_polarity
kw_min_max
title_subjectivity
min_negative_polarity
max_negative_polarity
abs_title_sentiment_polarity
abs_title_subjectivity
num_keywords
kw_min_min
0
2
4
6
8
MeanDecreaseGini



```
Confusion Matrix and Statistics


predicted.classes   Y   N
                Y  26  13
                N 111 249

               Accuracy : 0.6892
                 95% CI : (0.6413, 0.7343)
    No Information Rate : 0.6566
    P-Value [Acc > NIR] : 0.09303

                  Kappa : 0.169

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.18978
            Specificity : 0.95038
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.69167
             Prevalence : 0.34336
         Detection Rate : 0.06516
   Detection Prevalence : 0.09774
      Balanced Accuracy : 0.57008

       'Positive' Class : Y
```

Principal component Analysis

Fig-PCA-1

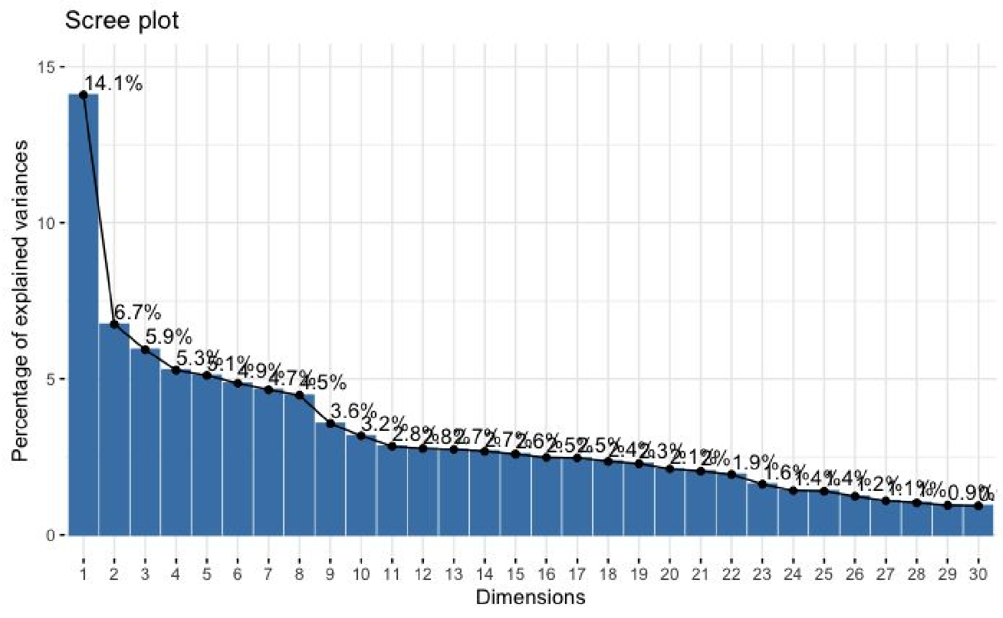
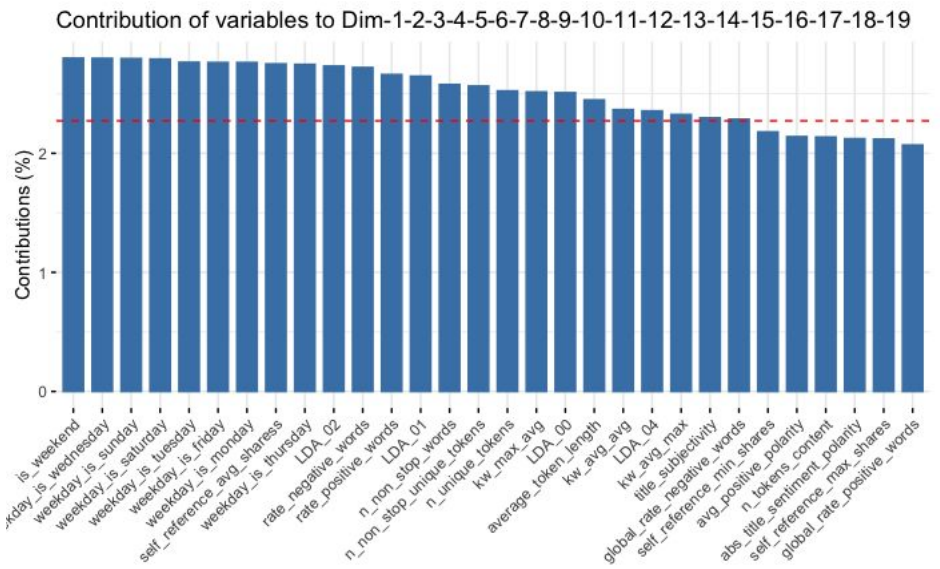


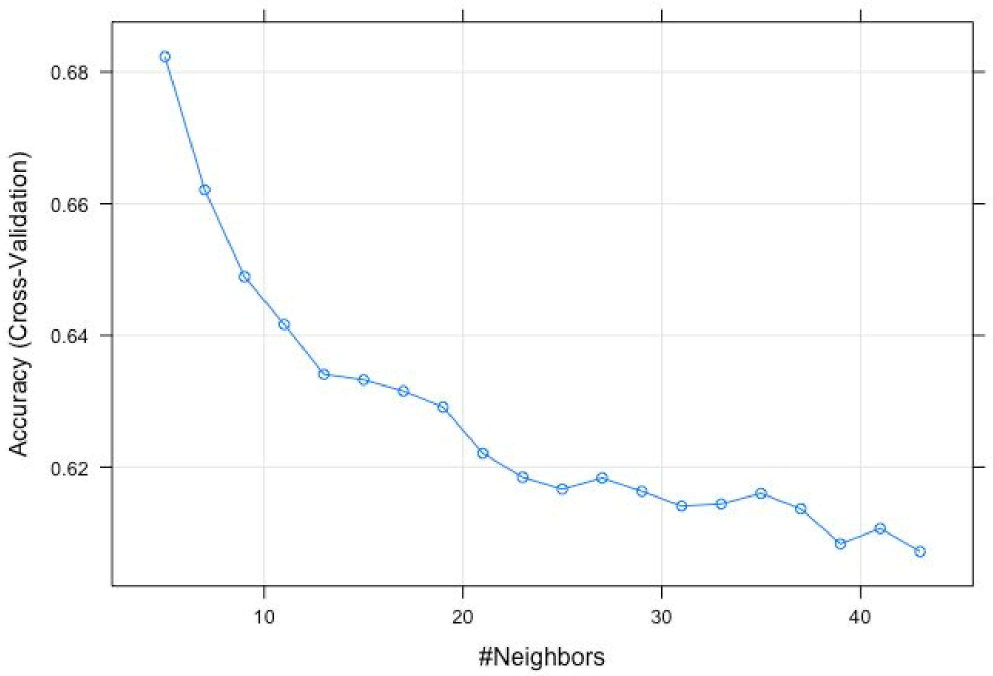
Fig-PCA-2



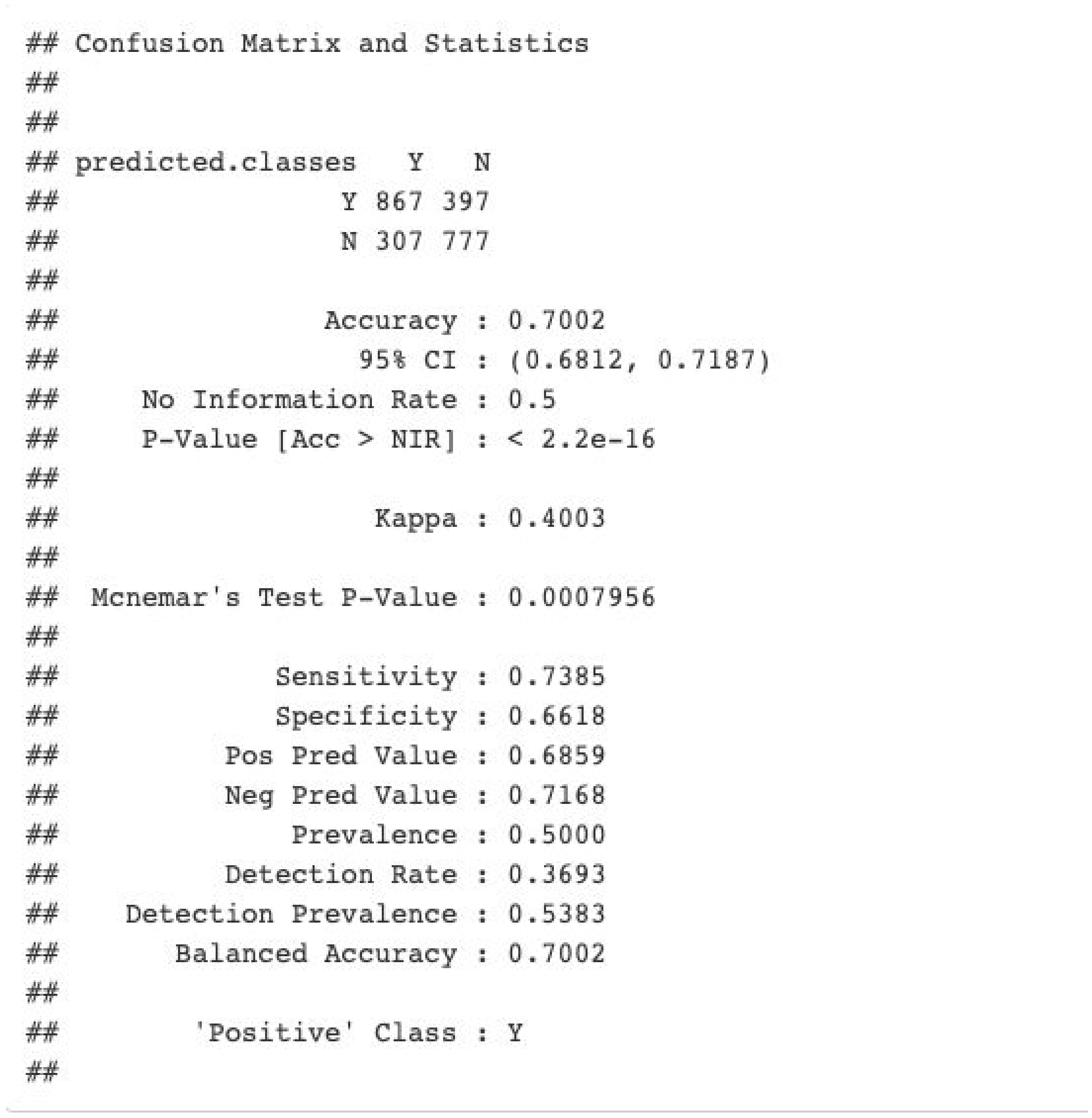
Individual contribution to principal components

kNN Model

Optimal value(k=5)



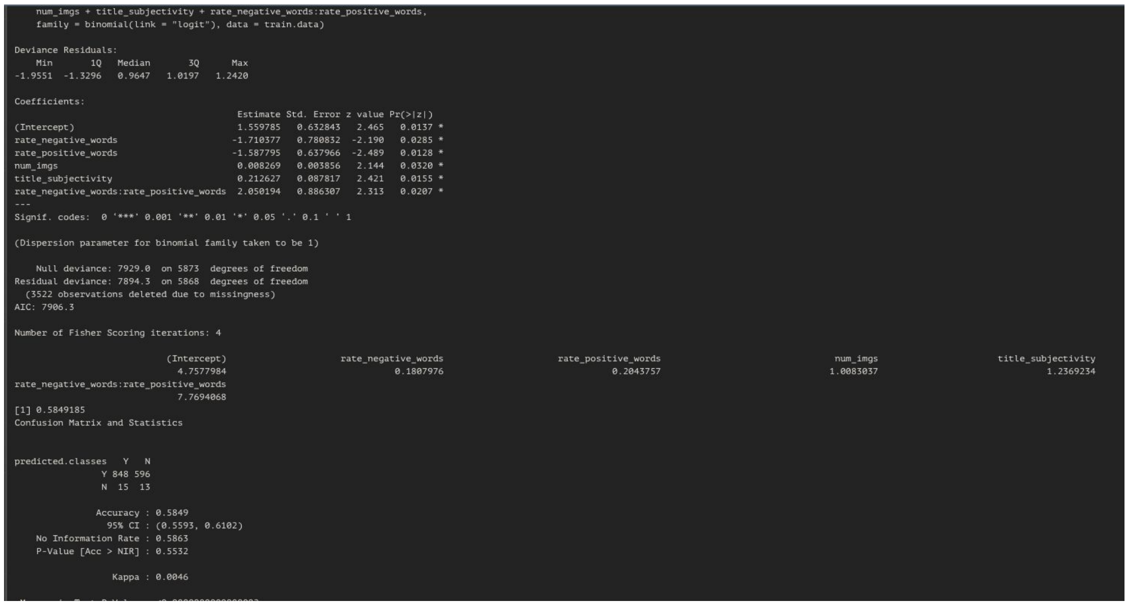
Confusion Matrix



```
## Confusion Matrix and Statistics
##
##
## predicted.classes   Y   N
##                 Y 867 397
##                 N 307 777
##
##                Accuracy : 0.7002
##                  95% CI : (0.6812, 0.7187)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4003
##
##  Mcnemar's Test P-Value : 0.0007956
##
##             Sensitivity : 0.7385
##             Specificity : 0.6618
##          Pos Pred Value : 0.6859
##          Neg Pred Value : 0.7168
##              Prevalence : 0.5000
##          Detection Rate : 0.3693
##    Detection Prevalence : 0.5383
##       Balanced Accuracy : 0.7002
##
##        'Positive' Class : Y
##
```

Objective II Logistic regression(Interactions,square terms)

Output for datachannel World



```
    num_imgs + title_subjectivity + rate_negative_words:rate_positive_words,
    family = binomial(link = "logit"), data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9551  -1.3296   0.9647   1.0197   1.2420

Coefficients:
                                         Estimate Std. Error z value Pr(>|z|)
(Intercept)                              1.559785   0.632843   2.465   0.0137 *
rate_negative_words                     -1.710377   0.780832  -2.190   0.0285 *
rate_positive_words                     -1.587795   0.637966  -2.489   0.0128 *
num_imgs                                 0.008269   0.003856   2.144   0.0320 *
title_subjectivity                       0.212627   0.087817   2.421   0.0155 *
rate_negative_words:rate_positive_words  2.050194   0.886307   2.313   0.0207 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7929.0  on 5873  degrees of freedom
Residual deviance: 7894.3  on 5868  degrees of freedom
  (3522 observations deleted due to missingness)
AIC: 7906.3

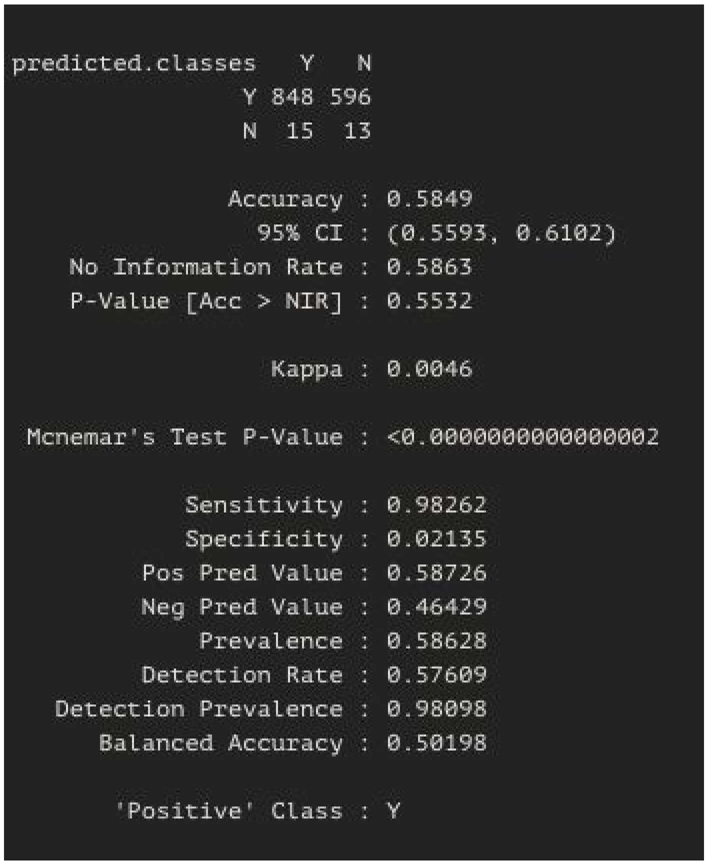
Number of Fisher Scoring iterations: 4

                            (Intercept)                     rate_negative_words                     rate_positive_words                                num_imgs                      title_subjectivity
                              4.7577984                               0.1807976                               0.2043757                               1.0083037                               1.2369234
rate_negative_words:rate_positive_words
                              7.7694068
[1] 0.5849185
Confusion Matrix and Statistics


predicted.classes   Y   N
                Y 848 596
                N  15  13

               Accuracy : 0.5849
                 95% CI : (0.5593, 0.6102)
    No Information Rate : 0.5863
    P-Value [Acc > NIR] : 0.5532

                  Kappa : 0.0046
```



```
predicted.classes   Y   N
                Y 848 596
                N  15  13

               Accuracy : 0.5849
                 95% CI : (0.5593, 0.6102)
    No Information Rate : 0.5863
    P-Value [Acc > NIR] : 0.5532

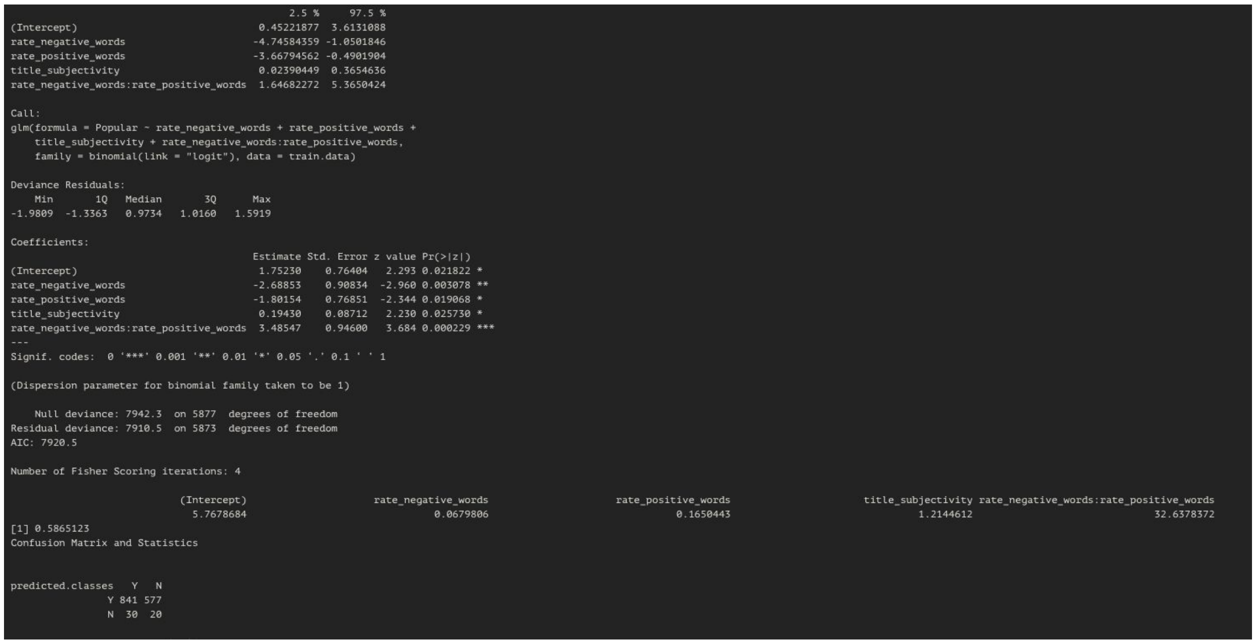
                  Kappa : 0.0046

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.98262
            Specificity : 0.02135
         Pos Pred Value : 0.58726
         Neg Pred Value : 0.46429
             Prevalence : 0.58628
         Detection Rate : 0.57609
   Detection Prevalence : 0.98098
      Balanced Accuracy : 0.50198

       'Positive' Class : Y
```

Output for datachannel Technology



```
                                             2.5 %     97.5 %
(Intercept)                             0.45221877  3.6131088
rate_negative_words                    -4.74584359 -1.0501846
rate_positive_words                    -3.66794562 -0.4901904
title_subjectivity                      0.02390449  0.3654636
rate_negative_words:rate_positive_words  1.64682272  5.3650424

Call:
glm(formula = Popular ~ rate_negative_words + rate_positive_words +
    title_subjectivity + rate_negative_words:rate_positive_words,
    family = binomial(link = "logit"), data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9809  -1.3363   0.9734   1.0160   1.5919

Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                              1.75230    0.76404   2.293 0.021822 *
rate_negative_words                     -2.68853    0.90834  -2.960 0.003078 **
rate_positive_words                     -1.80154    0.76851  -2.344 0.019068 *
title_subjectivity                       0.19430    0.08712   2.230 0.025730 *
rate_negative_words:rate_positive_words  3.48547    0.94600   3.684 0.000229 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

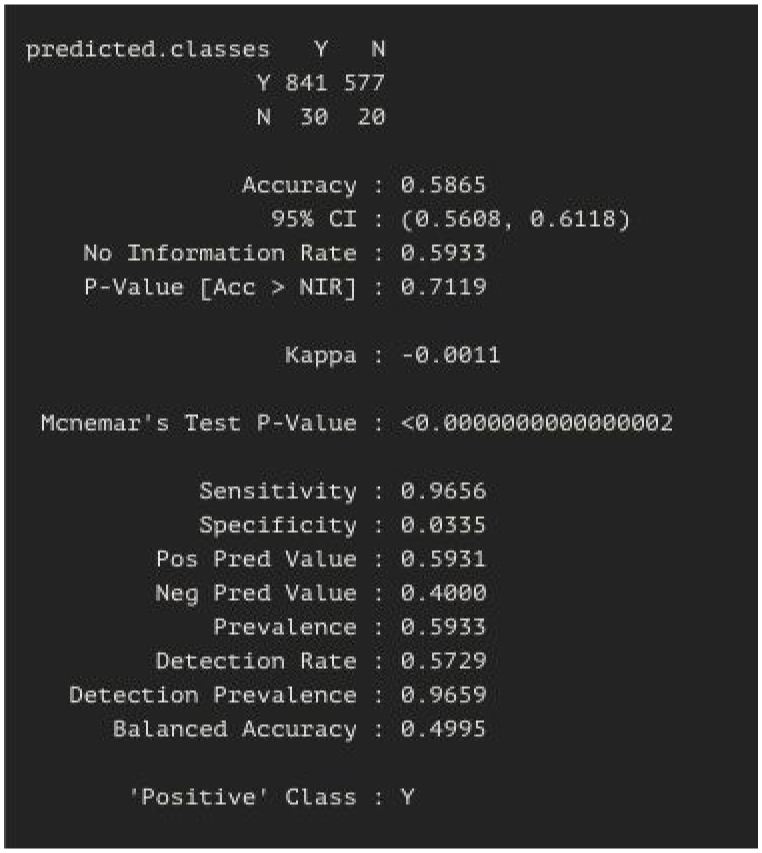
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7942.3  on 5877  degrees of freedom
Residual deviance: 7910.5  on 5873  degrees of freedom
AIC: 7920.5

Number of Fisher Scoring iterations: 4

                            (Intercept)                     rate_negative_words                     rate_positive_words                      title_subjectivity rate_negative_words:rate_positive_words
                              5.7678684                               0.0679806                               0.1650443                               1.2144612                              32.6378372
[1] 0.5865123
Confusion Matrix and Statistics


predicted.classes   Y   N
                Y 841 577
                N  30  20
```



```
predicted.classes   Y   N
                Y 841 577
                N  30  20
```

```
               Accuracy : 0.5865
                 95% CI : (0.5608, 0.6118)
    No Information Rate : 0.5933
    P-Value [Acc > NIR] : 0.7119

                  Kappa : -0.0011

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.9656
            Specificity : 0.0335
         Pos Pred Value : 0.5931
         Neg Pred Value : 0.4000
             Prevalence : 0.5933
         Detection Rate : 0.5729
   Detection Prevalence : 0.9659
      Balanced Accuracy : 0.4995

       'Positive' Class : Y
```

Building Random Forest Model

Code

~~~
```{r  setup, include=FALSE}
knitr::opts_knit$set(out.format =TRUE)
knitr::knit_theme$set("edit-vim")
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(corrplot)
library(ggthemes)
library(gridExtra)
library(caret)
library(MASS)
library(DMwR) # SMOTE function to handle unbalanced response
library(ResourceSelection)
library(plotROC)
library(broom)
library(gridExtra)
library(DataExplorer)
library(FactoMineR)
library(factoextra)
```
## Add Binary Response

```{r}
super3_alt_ds         <- super3_ds
median_shares         <- median(super3_alt_ds$shares)
super3_alt_ds$Popular <- ifelse(super3_alt_ds$shares > median_shares , 'Y', 'N')
super3_alt_ds$Popular <- as.factor(super3_alt_ds$Popular)
```

## Structure of the dataset

```{r}
str(super3_alt_ds)

str(super3_ds)
~~~

~~~
non_perdictor_cols <-
c('url','timedelta','data_channel_is_lifestyle','data_channel_is_entertainment',
'data_channel_is_bus','data_channel_is_socmed','data_channel_is_tech','data_channel_is_worl
d','shares')
```
## Remove columns with negative numbers
```{r}
list_cols_cnt=1
clist <- array()
for (i in 3:(ncol(super3_alt_ds)-1))
   if ( length(which(super3_alt_ds[,i] <0)) > 0) {
        print(paste(i,names(super3_alt_ds)[i],length(which(super3_alt_ds[,i] < 0))))
        clist[list_cols_cnt] <-  i
        list_cols_cnt <- list_cols_cnt + 1
   }
}
super3_alt_ds <- super3_alt_ds[,-clist]
```

## Function Definitions
```{r}
datachannel_subsets_ds <- function(data_channel) {
  ds_index <- which(colnames(super3_alt_ds)==data_channel)
  dc_subset <- super3_alt_ds %>% filter(super3_alt_ds[,ds_index]==1)
  dc_subset <- dc_subset[,-which(colnames(super3_alt_ds) %in% non_perdictor_cols)]
}
linear_assumptions <- function(model,dataset) {
  # Select only numeric predictors
  probabilities <- predict(model, type = "response")
  mydata <- dataset %>% dplyr::select_if(is.numeric)
  predictors <- colnames(mydata)
  # Bind the logit and tidying the data for plot
  mydata <- mydata %>% mutate(logit = log(probabilities/(1-probabilities))) %>% gather(key =
"predictors", value = "predictor.value", -logit)
  model_preds <- names(model$coefficients)[-1]
  myplots <- list()
  for(i in 1:length(model_preds)) {
     myplots[[i]] <- mydata %>%
     filter(predictors==model_preds[i]) %>%
     ggplot(aes(x=predictor.value, y=logit))+
     geom_point(size = 0.5, alpha = 0.5) +
     xlab(model_preds[i])+
     geom_smooth(method = "lm") +
~~~

~~~~
    theme_bw()
  }
  n <- length(myplots)
  nCol <- floor(sqrt(n))
  do.call("grid.arrange", c(myplots, ncol=nCol))
}
pca_variable_selection  <- function (dataset){
  res.pca <- PCA(dataset, scale.unit = TRUE, ncp = 50, graph = TRUE)
  pca_features <- fviz_contrib(res.pca, choice = "var", axes = 1:2, top = 30)
  eigens <-as.data.frame(res.pca$eig) %>% mutate(eigen=as.integer(eigenvalue)) %>%
filter(eigen >0)
  features <- pca_features$data %>% arrange(desc(contrib))
  result <- c(features,eigens)
  return(result)
}
```
# Objective 1 - Basic Logistic regression
## Part-I DataChannel World create SMOTE subset function to handle unbalanced response
variable
```{r}
subset_ds <- datachannel_subsets_ds('data_channel_is_world')
set.seed(130)
# Unbalanced Dataset
count(subset_ds,Popular)
smote_world_ds <- SMOTE(Popular ~ ., subset_ds, perc.over = 150,perc.under=200)
# Balanced Dataset
count(smote_world_ds,Popular)
```
  Build a model and make predictions using stepwise regression <br>
```{r}
predictors1 <- c('weekday_is_saturday', 'weekday_is_friday', 'weekday_is_wednesday',
'weekday_is_tuesday',
             'self_reference_min_shares', 'n_tokens_content',  'n_tokens_title',  'num_hrefs',
             'num_self_hrefs', 'num_imgs', 'num_videos', 'average_token_length', 'num_keywords',
             'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02', 'LDA_04', 'global_subjectivity',
'global_rate_positive_words',
             'rate_positive_words', 'abs_title_subjectivity', 'title_subjectivity')
set.seed(123)
training.samples <- smote_world_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <- smote_world_ds[training.samples,]
test.data   <- smote_world_ds[-training.samples,]
~~~~

```
pca_result <- pca_variable_selection(train.data[,-c(21:28,45)])
formula    <- as.formula(paste('Popular',paste(pca_result$name[1:length(pca_result$eigen-3)],
collapse = " + "),sep = " ~ "))
#formula    <- as.formula(paste('Popular',paste(predictors2, collapse = " + "),sep = " ~ "))
pca_features <- fviz_contrib(res.pca, choice = "var", axes = 1:2, top = 30)
#model<-glm(Popular~average_token_length+n_non_stop_unique_tokens+n_unique_tokens+gl
obal_subjectivity+
#kw_avg_avg+global_rate_positive_words+LDA_02, data=train.data,
family=binomial(link="logit")) %>% stepAIC(trace = FALSE,direction="both")
model      <- glm(formula, data=train.data, family=binomial(link="logit")) %>% stepAIC(trace =
FALSE,direction="both")
#model      <- glm(Popular~num_videos+num_videos*num_imgs, data=train.data,
family=binomial(link="logit")) %>% stepAIC(trace = FALSE,direction="both")
```
```{r    Model Summary}
summary(model)
  exp(coef(model))
exp(cbind(OR = coef(model), confint(model)))
## CIs using profiled log-likelihood
confint(model)
confint.default(model)
```
```{r   Predictions and Confusion matrix}
probabilities     <- model %>% predict(test.data, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "Y", "N")
# Model accuracy
mean(predicted.classes==test.data$Popular)
cm_table <-table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
## Confunsion Matrix
CM <- confusionMatrix(cm_table)
CM
```
## Checking on Assumptions
```{r    Goodness of fit Test}
trainingPopular <- if_else(train.data$Popular=='Y',1,0)
hoslem.test(model$y, fitted(model),g = 10)
```
```{r R-square}
ll.null <-model$null.deviance/-2
ll.proposed <- model$deviance/-2
(ll.null-ll.proposed)/ll.null
```
```{r   Relationship between predictor and Logit}
```

```
linear_assumptions(model,train.data)
```
~~~
```{r Influential points Analysis}
plot(model, which = 4, id.n = 3)
# Extract model results
model.data <- augment(model) %>%
  mutate(index = 1:n())
model.data %>%
  filter(abs(.std.resid) > 3)
ggplot(model.data, aes(index, .std.resid)) +
  geom_point(aes(color = Popular), alpha = .5) +
  theme_bw()
~~~
## Part-II DataChannel Technology
```{r    create SMOTE subset function to handle unbalanced response variable}
subset_ds <- datachannel_subsets_ds('data_channel_is_tech')
set.seed(130)
# Unbalanced Dataset
count(subset_ds,Popular)
smote_tech_ds <- SMOTE(Popular ~ ., subset_ds, perc.over = 150,perc.under=200)
# Balanced Dataset
count(smote_world_ds,Popular)
```
```{r    Build a model and make predictions using stepwise regression <br>}
set.seed(123)
training.samples <- smote_tech_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <- smote_tech_ds[training.samples,]
test.data   <- smote_tech_ds[-training.samples,]
pca_result <- pca_variable_selection(train.data[,-c(21:28,45)])
formula    <- as.formula(paste('Popular',paste(pca_result$name[1:length(pca_result$eigen)],
collapse = " + "),sep = " ~ "))
model      <- glm(formula, data=train.data, family=binomial(link="logit")) %>% stepAIC(trace =
FALSE)
```
```{r    Model Summary}
summary(model)
summary(model)
exp(coef(model))
exp(cbind(OR = coef(model), confint(model)))
## CIs using profiled log-likelihood
confint(model)
confint(model)
confint.default(model)
```

~~~
```
```{r Predictions and Confusion matrix}
probabilities     <- model %>% predict(test.data, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "Y", "N")
# Model accuracy
mean(predicted.classes==test.data$Popular)
cm_table <-table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
## Confunsion Matrix
CM <- confusionMatrix(cm_table)
CM
```
## Checking on Assumptions
```{r    Goodness of fit Test}
trainingPopular <- if_else(train.data$Popular=='Y',1,0)
hoslem.test(trainingPopular, fitted(model),g = 10)
```
```{r R-square}
ll.null <-model$null.deviance/-2
ll.proposed <- model$deviance/-2
(ll.null-ll.proposed)/ll.null
```
```{r Relationship between predictor and Logit}
linear_assumptions(model,train.data)
```
```{r Influential points Analysis}
plot(model, which = 4, id.n = 3)
# Extract model results
model.data <- augment(model) %>% mutate(index = 1:n())
model.data %>%
  filter(abs(.std.resid) > 3)
ggplot(model.data, aes(index, .std.resid)) +
  geom_point(aes(color = Popular), alpha = .5) +
  theme_bw()
```
## Objective 2 - More adavanced models
```{r  eval = FALSE, echo = FALSE}
library(MASS)
# Linear Discriminant Analysis
## Assumptions
set.seed(123)
training.samples <- smote_world_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
lda_predictors <- c('average_token_length','avg_positive_polarity','rate_positive_words'
                    ,'kw_avg_max','Popular')
~~~

```
predictors <- c('self_reference_min_shares', 'n_tokens_content',  'n_tokens_title',  'num_hrefs',
              'num_self_hrefs', 'num_imgs', 'num_videos', 'average_token_length', 'num_keywords',
               'LDA_00', 'LDA_01', 'LDA_02', 'LDA_04', 'global_subjectivity',
'global_rate_positive_words',
              'rate_positive_words', 'abs_title_subjectivity', 'title_subjectivity')
```
```{r  eval = FALSE, echo = FALSE}
str(smote_world_ds)
set.seed(123)
training.samples <- smote_world_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
train.data <- smote_world_ds[training.samples, lda_predictors]
test.data <- smote_world_ds[-training.samples, lda_predictors]
# Estimate preprocessing parameters
train.preproc.param <- train.data %>%  preProcess(method = c("center", "scale"))
test.preproc.param  <- test.data %>%  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformed <- train.preproc.param %>% predict(train.data)
test.transformed <- test.preproc.param %>% predict(test.data)
# Fit the model
model <- lda(Popular~., data = train.data)
# Make predictions
predictions <- model %>% predict(test.data)
# Model accuracy
mean(predictions$class==test.data$Popular)
plot(model)
# Model accuracy
mean(predictions==test.data$Popular)
cm_table <-table(predictions$class, test.data$Popular)[c(2,1),c(2,1)]
## Confunsion Matrix
CM <- confusionMatrix(cm_table)
CM
```
```{r QDA eval = FALSE, echo = FALSE}
predictors <- c('weekday_is_saturday', 'weekday_is_friday', 'weekday_is_wednesday',
'weekday_is_tuesday',
              'self_reference_min_shares', 'n_tokens_content',  'n_tokens_title',  'num_hrefs',
              'num_self_hrefs', 'num_imgs', 'num_videos', 'average_token_length', 'num_keywords',
              'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02', 'LDA_04', 'global_subjectivity',
'global_rate_positive_words',
              'rate_positive_words', 'abs_title_subjectivity', 'title_subjectivity')
qda_ds <- datachannel_subsets_ds('data_channel_is_world')
set.seed(123)
training.samples <- qda_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
```

~~~
#train.data <- qda_ds[training.samples, -c(which(colnames(qda_ds) %in% collinear_vars),45)]
#test.data  <- qda_ds[-training.samples,-c(which(colnames(qda_ds) %in% collinear_vars),45)]
train.data <- qda_ds[training.samples, c(predictors,'Popular')]
test.data  <- qda_ds[-training.samples,c(predictors,'Popular')]
# Estimate preprocessing parameters
train.preproc.param <- train.data %>%  preProcess(method = c("center", "scale"))
test.preproc.param <- test.data %>%  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformed <- train.preproc.param %>% predict(train.data)
test.transformed <- test.preproc.param %>% predict(test.data)

# Fit the model
model <- qda(Popular~., data =train.data)
model
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions$class == test.transformed$Popular)
cm_table <-table(predictions$class, test.data$Popular)[c(2,1),c(2,1)]
## Confunsion Matrix
CM <- confusionMatrix(cm_table)
CM
```
```{r eval = FALSE, echo = FALSE}
# Random Forest
set.seed(123)
predictors <- c('weekday_is_saturday', 'weekday_is_friday', 'weekday_is_wednesday',
'weekday_is_tuesday',
            'self_reference_min_shares', 'n_tokens_content',  'n_tokens_title',  'num_hrefs',
            'num_self_hrefs', 'num_imgs', 'num_videos', 'average_token_length', 'num_keywords',
            'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02', 'LDA_04', 'global_subjectivity',
'global_rate_positive_words',
            'rate_positive_words', 'abs_title_subjectivity', 'title_subjectivity','Popular')
set.seed(130)
rf_ds <- datachannel_subsets_ds('data_channel_is_world')
rf_sample <- sample(1:nrow(rf_ds), 2000, replace = FALSE, prob = NULL)
smote_rf_world_ds <- SMOTE(Popular ~ ., rf_ds, perc.over = 150,perc.under=200)
#training.samples <- rf_ds[rf_sample,]$Popular %>% createDataPartition(p = 0.8, list = FALSE)
#train.data  <- rf_ds[rf_sample,][training.samples, ]
#test.data   <- rf_ds[rf_sample,][-training.samples, ]
set.seed(123)
training.samples <- rf_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <- rf_ds[ training.samples,predictors ]
~~~

```
test.data   <- rf_ds[-training.samples, predictors]
# Fit the model on the training set
set.seed(123)
model <- train(
  Popular ~., data = train.data, method = "rf",
  trControl = trainControl("cv", number = 10),
  importance = TRUE )
# Best tuning parameter
model$bestTune

# Make predictions on the test data
predicted.classes <- model %>% predict(test.data)
mean(predicted.classes == test.data$Popular)
rf_table <- table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
CM <- confusionMatrix(rf_table)
CM
caret::varImp(model)

ggplot(caret::varImp(model)) +
geom_bar(stat = 'identity', fill = 'steelblue', color = 'black') +
ylab("Feature Importance - Knn Classification ")+
scale_y_continuous(limits = c(0, 105), expand = c(0, 0)) +
theme_light()
```
```{r kNN Data Channel World}
knn_ds <- datachannel_subsets_ds('data_channel_is_world')
set.seed(130)
smote_knn_world_ds <- SMOTE(Popular ~ ., knn_ds, perc.over = 150,perc.under=200)

count(smote_knn_world_ds,Popular)

set.seed(123)
training.samples <- smote_knn_world_ds$Popular %>% createDataPartition(p = 0.8, list =
FALSE)
train.data  <- smote_knn_world_ds[ training.samples, ]
test.data   <- smote_knn_world_ds[-training.samples, ]

# Fit the model on the training set
set.seed(123)
model <- train( Popular ~., data = train.data, method = "knn",
               trControl = trainControl("cv", number = 10),
               preProcess = c("center","scale"),
               tuneLength = 20
```

```
            )
#Plot model accuracy vs different values of k
plot(model)

model$bestTune

predicted.classes <- model %>% predict(test.data)
head(predicted.classes)

# Compute model accuracy rate
mean(predicted.classes == test.data$Popular)

knn_table <- table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]

CM <- confusionMatrix(knn_table)
CM

ggplot(caret::varImp(model)) +
geom_bar(stat = 'identity', fill = 'steelblue', color = 'black') +
ylab("Feature Importance - Knn Classification ")+
scale_y_continuous(limits = c(0, 105), expand = c(0, 0)) +
theme_light()
```
```{r kNN Data Channel Technology}
knn_ds <- datachannel_subsets_ds('data_channel_is_tech')
set.seed(130)
smote_knn_tech_ds <- SMOTE(Popular ~ ., knn_ds, perc.over = 150,perc.under=200)

count(smote_knn_tech_ds,Popular)

set.seed(123)
training.samples <- smote_knn_tech_ds$Popular %>% createDataPartition(p = 0.8, list =
FALSE)
train.data  <- smote_knn_tech_ds[ training.samples, ]
test.data   <- smote_knn_tech_ds[-training.samples, ]

# Fit the model on the training set
set.seed(123)
model <- train( Popular ~., data = train.data, method = "knn",
              trControl = trainControl("cv", number = 10),
              preProcess = c("center","scale"),
              tuneLength = 20
            )
```

```
#Plot model accuracy vs different values of k
plot(model)

model$bestTune

predicted.classes <- model %>% predict(test.data)
head(predicted.classes)

# Compute model accuracy rate
mean(predicted.classes == test.data$Popular)

knn_table <- table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]

CM <- confusionMatrix(knn_table)
CM

summary(model)
#exp(coef(model))
#exp(cbind(OR = coef(model), confint(model)))
```
```{r LASSO Data Chanel World}
library(glmnet)
# Split the data into training and test set
lasso_world_ds <- datachannel_subsets_ds('data_channel_is_world')
set.seed(123)
training.samples <- lasso_world_ds$Popular %>%  createDataPartition(p = 0.8, list = FALSE)
train.data      <- lasso_world_ds[training.samples, ]
test.data       <- lasso_world_ds[-training.samples, ]
# Dumy code categorical predictor variables
x <- model.matrix(Popular~., train.data)[,-1]
# Convert the outcome (class) to a numerical variable
y <- ifelse(train.data$Popular == 'Y', 1, 0)
# Find the best lambda using cross-validation
set.seed(123)
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial",nlambda = 300,type.measure = "class")
# Fit the final model on the training data
model <- glmnet(x, y, alpha = 1, family = "binomial", lambda = cv.lasso$lambda.min)
# Display regression coefficients
model$beta[,1]
# Make predictions on the test data
x.test <- model.matrix(Popular ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, 'Y', 'N')
```

~~~
# Model accuracy
observed.classes <- test.data$Popular
mean(predicted.classes == observed.classes)
lasso_table <- table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
CM <- confusionMatrix(lasso_table)
CM
summary(model)
```
```{r Data Channel Technology}
library(glmnet)
lasso_tech_ds <- datachannel_subsets_ds('data_channel_is_world')

# Split the data into training and test set

set.seed(123)
training.samples <- lasso_tech_ds$Popular %>%  createDataPartition(p = 0.8, list = FALSE)
train.data      <- lasso_tech_ds[training.samples, ]
test.data       <- lasso_tech_ds[-training.samples, ]

# Dumy code categorical predictor variables
x <- model.matrix(Popular~., train.data)[,-1]
# Convert the outcome (class) to a numerical variable
y <- ifelse(train.data$Popular == 'Y', 1, 0)

# Find the best lambda using cross-validation
set.seed(123)
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial",nlambda = 300,type.measure = "class")
# Fit the final model on the training data
model <- glmnet(x, y, alpha = 1, family = "binomial", lambda = cv.lasso$lambda.min)
# Display regression coefficients

model$beta[,1]

# Make predictions on the test data
x.test <- model.matrix(Popular ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, 'Y', 'N')
# Model accuracy
observed.classes <- test.data$Popular
mean(predicted.classes == observed.classes)
lasso_table <- table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
CM <- confusionMatrix(lasso_table)
CM
~~~

```
summary(model)
```
~~~
```{r Objective II Logistic Model with interactions}
set.seed(123)
training.samples <- smote_world_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <-
tech_ds[training.samples,c('num_imgs','rate_negative_words','rate_positive_words','Popular','titl
e_subjectivity')]
test.data <- tech_ds[-training.samples,
c('num_imgs','rate_negative_words','rate_positive_words','Popular','title_subjectivity')]

model  <-
glm(Popular~rate_negative_words+rate_positive_words+num_imgs+num_imgs*rate_positive_w
ords+title_subjectivity+rate_positive_words*rate_negative_words,
                    data=train.data,family=binomial(link="logit")) %>% stepAIC(trace =
FALSE,direction="backward")

confint(model)
summary(model)
exp(coef(model))

probabilities <- model %>% predict(test.data, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "Y", "N")
# Model accuracy
mean(predicted.classes == test.data$Popular)

cm_table <-table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
## Confunsion Matrix
CM <- confusionMatrix(cm_table)
CM
~~~
```{r Technology}
tech_ds <- datachannel_subsets_ds('data_channel_is_tech')
training.samples <- tech_ds$Popular %>% createDataPartition(p = 0.8, list = FALSE)
train.data<-tech_ds[training.samples,c('num_imgs','rate_negative_words','rate_positive_words','
Popular','title_subjectivity')]
test.data<- tech_ds[-training.samples,
c('num_imgs','rate_negative_words','rate_positive_words','Popular','title_subjectivity')]
mode<-glm(Popular~rate_negative_words+rate_positive_words+num_imgs+num_imgs*rate_po
sitive_words+title_subjectivity+rate_positive_words*rate_negative_words,
                    data=train.data,family=binomial(link="logit")) %>% stepAIC(trace =
FALSE,direction="backward")
confint(model)
```

```
summary(model)
exp(coef(model))

probabilities <- model %>% predict(test.data, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "Y", "N")
# Model accuracy
mean(predicted.classes == test.data$Popular)
cm_table <-table(predicted.classes, test.data$Popular)[c(2,1),c(2,1)]
## Confunsion Matrix
CM <- confusionMatrix(cm_table)
CM
```
```{r removing correlated variables}
newsdata4
<-newsdata_alt[,-c(1,2,7,8,9,10,11,12,13,18,20,22,25,26,29,31,33,34,37,38,40,41,46,49)]
str(newsdata4)
set.seed(111)
newsvector <- sample(1:39644,39643)
reducednews <- newsdata4[newsvector,-c(1,2,3)]
reducednews
introduce(reducednews)
#write.csv(newsdata4, "PopularityData2.csv")
```
Code Chunk Random Forest
``` {r data cleanup}
data_channel <- c('Other','Lifestyle','Entertainment','Business','Social
medial','Technology','World')
newsdata$data_channel <-
data_channel[newsdata$data_channel_is_lifestyle*1+newsdata$data_channel_is_entertainmen
t*2+
newsdata$data_channel_is_bus*3+newsdata$data_channel_is_socmed*4+newsdata$data_cha
nnel_is_tech*5+newsdata$data_channel_is_world*6+1]
newsdata$data_channel <- as.factor(newsdata$data_channel)
day_published <- c('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday')
newsdata$day_published <-
day_published[newsdata$weekday_is_monday*1+newsdata$weekday_is_tuesday*2+
newsdata$weekday_is_wednesday*3+newsdata$weekday_is_thursday*4+newsdata$weekday_
is_friday*5+                    newsdata$weekday_is_saturday*6+newsdata$weekday_is_sunday*7]
newsdata$day_published <- as.factor(newsdata$day_published)
newsdata$is_weekend <- ifelse(newsdata$is_weekend >0 , 'Y', 'N')
newsdata$is_weekend <- as.factor(newsdata$is_weekend)
newsdata_alt <- newsdata[,-c(14:19,32:39)]
newsdata_alt <- newsdata_alt[,c(1,2,48,49,3:47)]
```

```
str(newsdata_alt)
```
~~~
```{r binary response setup on shares}
median_shares <- median(newsdata_alt$shares)
newsdata_alt$Popular <- ifelse(newsdata_alt$shares > median_shares , 'Y', 'N')
newsdata_alt$Popular <- as.factor(newsdata_alt$Popular)
str(newsdata_alt$Popular)
#newsdata_alt$date_published <- as.Date(substr(newsdata_alt$url,21,30),"%Y/%m/%d")
newsdata_alt <- newsdata_alt[,c(1:50)]
str(newsdata_alt)
~~~

```{r removing correlated variables}
newsdata4 <-
newsdata_alt[,-c(1,2,7,8,9,10,11,12,13,18,20,22,25,26,29,31,33,34,37,38,40,41,46,49)]
str(newsdata4)
set.seed(111)
newsvector <- sample(1:39644,39643)
reducednews <- newsdata4[newsvector,-c(1,2,3)]
reducednews
introduce(reducednews)
#write.csv(newsdata4, "PopularityData2.csv")
```
```{r Random Forest}
library(e1071)
library(randomForest)
library(caret)
library(tree)
rnews <- reducednews
#rnews <- read.csv("~/R/PopularityData2.csv",header=TRUE)
rnews$target<-as.factor(rnews$Popular)

#rnews$popularnew<-factor(rnews$popular)
#Simple tree fit
set.seed(2)
train=sample(1:nrow(rnews), 200)
news.test=rnews[-train,]

#Indepdent response to compare prediction performance on test set
rnews.test=rnews$popularnew[-train]

# Regular Tree
#tree.reducednews=tree(targetnew~.-Popular,reducednews,subset=train)
```

~~~~
#tree.pred=predict(tree.reducednews,news.test,type="class")
#table(tree.pred,popular.test)
#confusionMatrix(table(tree.pred,popular.test))

#Random Forest
set.seed(123)
rf.news<-randomForest(Popular~.-Popular,reducednews,subset=train,mtry=5,importance=T,ntre
e=500)

rf.news
plot(rf.news)
fit.pred<-predict(rf.news,newdata=news.test,type="response")
table(fit.pred,news.test$target)
sd <- confusionMatrix(table(fit.pred,news.test$target))
sd
cm <- confusionMatrix(table(fit.pred,news.test$target))
cm$table
fourfoldplot(cm$table)

library(ROCR)
rf.pred<-predict(rf.news,newdata=news.test,type="prob")
pred <- prediction(rf.pred[,2], news.test$Popular)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf,main="AUC of Test set RF - mtry=5")
abline(a=0, b= 1)
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))

varImpPlot (rf.news,type=1,main="Variable Importance")
varImpPlot (rf.news,type=2,main="Variable Importance")

#Variable Importance
var.imp = data.frame(importance(rf.news,
                                 type=2))
importance(rf.news)
# make row names as columns
var.imp$Variables = row.names(var.imp)
print(var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),])
```
~~~~