IEOR142 (Fall 2017) Project Report: Getting Real about Fake News

Aditya Tyagi, BS IEOR,
Class of 2019, (University of California, Berkeley)

Table of Contents

I. Motivation

An interesting recent development is the proliferation of deliberate misinformation or hoaxes via traditional print and broadcast news media or online social media – informally termed 'fake news'. Accusations of 'fake news' were prolifically used in the recent U.S. Presidential Elections and have become a frequently heard term in the broader national conversation. And so, identifying when a news story may be authentic or potentially 'fake' is of great importance to the mindful reader.

Naturally, the machine learning community has sought for ways to contribute a solution to this problem. For example, the recently concluded 'Fake News Challenge' challenged researchers to identify the 'stance' of a particular news article ("stance detection"). In particular, a popular Chrome plugin ("BS Detector") helps users identify whether a particular webpage contains information that is 'fake', 'biased', 'a conspiracy', etc. To accomplish this, it uses a manually compiled list of untrustworthy websites and then mechanically checks the target webpage for links/references to such dubious websites. However, given the sheer size of the World Wide Web, and the expected increase in such websites, such a manual/hand-labelling approach may be infeasible.

Recently, prominent social media platforms were under the spotlight in Washington for failing to detect and report large-scale Russian Government involvement in the recent US Presidential Elections through political advertising, targeted news stories, bots spreading misinformation at scale, etc. It is more likely than not that these companies are devoting a significant amount of time & efforts to developing algorithmic approaches that can monitor the spread of such information through social networks and advise accordingly.

In a nutshell, my project aims to contribute to this exciting new space by using the automated ML approaches we have learned in class. Thematically, this is a multi-class classification problem where the labels are "fake", "conspiracy", "biased", etc. and the features consist of metadata about the website, as well as the text of the headline.

To complete this project, I learnt to use various outside technologies (h2o, tidytext) as well as self-studying core NLP concepts beyond what the class required. *Textmining with R: A Tidy Approach by Julia Silge & David Robinson,* in particular, enabled me to work on the core NLP portions of this project.

II. Data

High-level overview

Broadly speaking, the data I gathered consisted of three main types:

- Text of the website headline ("BOOM! Math shows Trump would have beaten Obama in Romney-Obama Election!")
- Web traffic metadata regarding the website: user demographics, bounce rate, daily page views, etc.
- Social Media Performance: no. of likes/comments/shares on Facebook, pins on Pinterest, etc.

Methodology

The following steps were taken to assemble the final dataset:

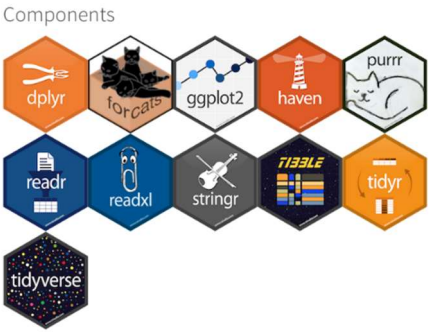
1) A base, labelled dataset of ~13000 'fake news' articles (bias, bs, conspiracy, junk science, etc.) was obtained from Kaggle; features that were present were headline text, author, site URL, and other basic information about the article.
    a. NAs were filtered out to leave us with ~6000 records.
2) To really test the discriminative power of the model, some 'legitimate' (i.e. non-fake) articles need to be added as well:
    a. I scraped ~150 politics-based articles from the CNN website using.
    b. I scraped ~100 politics-based articles from the Fox News website.
    c. I used the NY Times API to obtain ~additional 1100 articles.
    d. The above articles were added to the base dataset under the label 'legitimate'.
3) I used the SharedCount API to add ~6 columns of social media data to the dataset.
    a. These were the 'No. of Stumbles' on StumbleUpon, 'Pins' on Pinterest, 'Shares' on LinkedIn, 'Reactions', 'Comments', 'Shares' on Facebook.
    b. This information was collected at the level of each domain represented in the dataset. (i.e. Shares/Comments on FoxNews.com as opposed to FoxNews.com/a_given_article and stored in a separate table ("socialmedia").
4) I then proceeded to add ~13 columns of web traffic metadata to the dataset from Amazon Web Services' Alexa service.
    a. This included interesting digital marketing metrics such as bounce rate, average time spent on page, daily pageviews per visitor, etc.
    b. This information was also collected at level of each domain represented in the dataset and stored in a separate table ("webstats")
5) I finally inner joined the above two tables ("socialmedia", "webstats") with the base dataset to create the final data to be used for modelling.
6) We now have a rich dataset consisting of ~7000 observations spread across 26 variables.
7) However, the class representation is heavily dominated by the "BS" class (~75% of all observations), and there are only 1 or 2 obs. each of 'state news':
    a. I randomly sampled a subset of all "BS" observations and included them in the final dataset to balance things out.
    b. I deleted all observations having a class that was considered too rare.

8) Basic data cleaning was then carried out (type conversions, ID column, etc.) and then training & test sets were created using a 70:30 split (~2400 articles in training, and ~1035 articles for testing)

Tools & Technologies Used

The following range of tools & technologies were employed in the above process:

- Python (Web Scraping, API calls), Excel (Data Wrangling), R (joins, cleaning, etc.)
- BeautifulSoup4 Library
- NYTimes API, Shared Count API
- BS Detector Chrome Plugin to classify articles (used by base dataset creators)
- Various client-side API libraries written by various GitHub users



III. Analytic Models

High Level Overview & Motivation

In the modelling stage, I sought to create the most predictive model for our dependent variable. An equally important goal was to get out of my comfort zone and learn modelling techniques that I was unfamiliar with, but that could be easily understood based on the foundations we were taught in class. After some deliberation, I decided to go ahead with a Stacked Ensemble Model since these types of models (and ensemble modelling approaches) in general are known to be particularly effective in data science competitions. The three underlying models were a Gradient Boosting Machine (for the social media part of the dataset), a Random Forest (for the webstats portion), and a Multinomial Logistic Regression for the text (nlp) part of the dataset. Hence the modelling procedure, was broken down into three parallel & independent processes for each of the above data components, and then recombined in the stacking process.

Stacked Ensemble Model (Super Learner Algorithm)

The following is a brief description of the SEM algorithm:

Setting up the ensemble.

(i) Specify a list of L base algorithms (with a specific set of model parameters).

(ii) Specify a metalearning algorithm.

Training the ensemble.

(i) Train each of the L base algorithms on the training set (Level 0 data)

(ii) Perform k-fold cross-validation on each of these learners and collect the cross-validated predicted values from each of the L algorithms.

(iii) The N cross-validated predicted values from each of the L algorithms can be combined to form a new N x L matrix. This matrix, along with the original response vector, is the "level-one" data. (N = number of rows in the training set.)

(iv) Train the metalearning algorithm on the level-one data. The "ensemble model" consists of the L base learning models and the metalearning model, which can then be used to generate predictions on a test set.

Predicting on new data.

(i) To generate ensemble predictions, first generate predictions from the base learners.

(ii) Feed those predictions as features into the metalearner to generate the ensemble prediction.

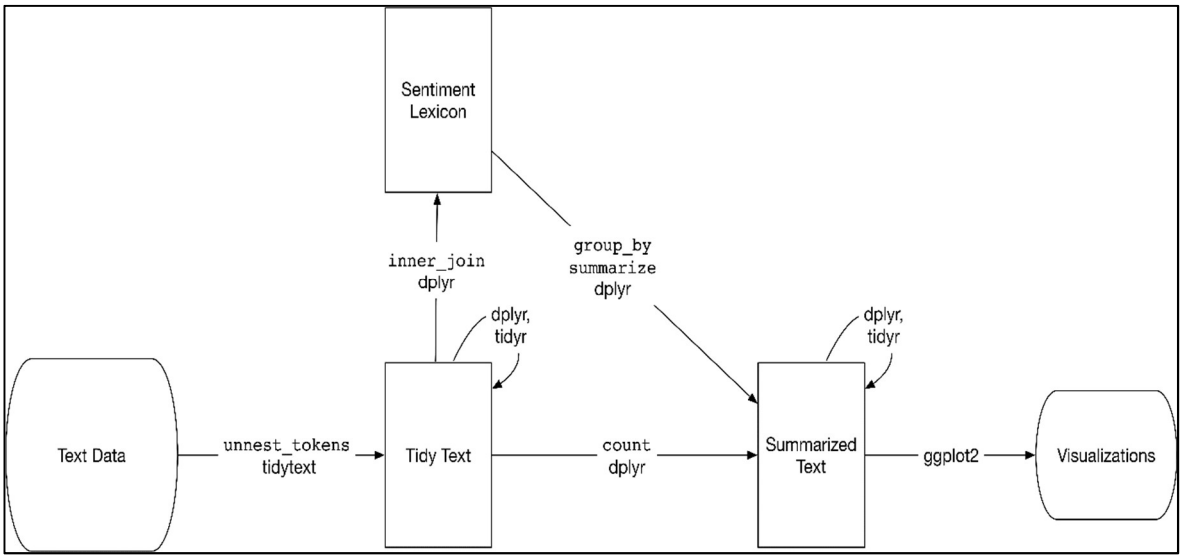
Methodology:

a) The NLP Aspect

For the text part of the dataset (article headlines), I took the following approach:

1) I removed punctuation from the headlines and converted to lower case
2) Tokenize the text into a tidytext format (one token per document per row) (see tidytext for R book). For this step, a 'document' was defined as an 'article headline', and a 'token' was simply a single word.
3) Performed a lexicon/dictionary-based dictionary sentiment analysis for each headline.
    a. Map each word present in the tokenized table to a numerical sentiment score based on the 'AFINN' lexicon.
    b. Sum up the individual scores for each word present in a headline, to get the overall sentiment of the headline.
    c. Append this 'sentiment' column to the entire dataset.
4) I then sought to identify words that were characteristic of each dependent variable class to facilitate further feature engineering.
    a. First stemmed the words present using the Porter Stemmer.
    b. For the purposes of this step, a 'document' was defined as all the headlines belonging to a particular classification ('junk science' for example), and a 'term' was a single word-stem.
    c. A tf-idf score was computed for each (document, term) pair.
    d. The top 3-5 word-stems with the highest tf-idf were identified for each class.
    e. A new set of around ~20 '*nlp indicator variables*' were added that would be 1 if a particular observation's headline contained the word-stem in question and 0 otherwise.
5) Finally, I tried out a few different models to predict the class of each observation and computed the respective out-of-sample accuracy using a test-set/holdout method. *(It is important to note that each these models were only allowed to consider the NLP features when making their predictions. This was done to encourage 'diversity of perspective'.)*
6) I then selected the model with the highest out-of-sample accuracy (A multi. log. regression)

Fig.1 : A typical flowchart for a tidytext lexicon-based sentiment analysis



b) The Social Media Aspect

For the social media aspect, I took the following approach:

1) Trained diverse models on only the 'social media' parts of the dataset.
2) Predicted the dependent variable for out-of-sample cases using each model.
3) Computed the out-of-sample accuracy for each model.
4) Picked the model with the highest out-of-sample accuracy. (A gradient boosting machine model)

c) The Web Traffic (webstats) Aspect

For the social media aspect, I took the following approach:

1) Trained diverse models on only the 'web traffic' parts of the dataset.
2) Predicted the dependent variable for out-of-sample cases using each model.
3) Computed the out-of-sample accuracy for each model.
4) Picked the model with the highest out-of-sample accuracy. (A random forest model)

Stacking it all together: The meta-learner model

To assemble the final 'stacked' model to be used for predicting the article type based on all information present, I employed the h2o framework since it automated the 'stacking' process described earlier:

1) Put the R dataframes into dedicated 'h2o frames'.
2) Defined the respective variable partitions and their underlying models using the appropriate h2o models
3) Assembled a stacked ensembled model with multinomial logistic regression as the meta-learner.
4) Use the ensemble model to make out of sample predictions and calculate relevant performance metrics.

Results & Interpretation

Overall, the results of the modelling process were excellent, producing out of sample accuracies in the neighborhood of 98%-99%. While this is definitely a positive result, such a high level of accuracy may be a cause for concern. Since we're using the test set/hold-out method, could it have been that somehow the training & test set came out quite similar to each other? Is our sample representative of the cases the model is expected to encounter when it's out in the world?

Is the model using a particularly discriminative feature that it may not have access to in the real-world? In future iterations of the study, would it make better sense to collect social media, web traffic data at the level of a single article/webpage instead of the at the level of the site URL?

Fig. 2 Most characteristic terms for each article type (above)

IV. Impact

In the era of 'fake news', such an automated approach to fake news detection has enormous value. Some key people we expect to benefit from this model are:

1) News Desks at various publishers/news corporations: Helps verify whether an incoming news story is genuine or not

2) Netizens who read get their news information online: Identify misinformation (from blogs, smaller websites, etc.)

3) Social Media Administrators at Facebook, Twitter, etc.: Locate disingenuous stories on their network and monitor their spread.

4) Campaign Consulting Firms: Assist in Web monitoring for potentially libelous information about their candidate

Some immediate real-world impact I hope will result from the project are:

- Creation of a Chrome plug-in based on our model that helps users identify whether a story they are viewing is trustworthy or not.
- Possible sharing of model with the makers of BS Detector, to improve their work

In future iterations, one may expand the scope of the project as follows:

- Increase granularity of features by gathering data at the level of a single article
- Incorporate greater depth of Natural Language Analysis by exploring the semantic relationships of words (topics present) and larger units of text (bigrams, ngrams, etc.)
- Take account of the temporal aspects of the article (day/date published)
- Explore the text of the entire article, instead of just the headline.
- Quantify the dynamic aspects of news story propagation (rate of spread, direction, etc.)

In conclusion, I'd like to thank Prof. Grigas, as well as the ever helpful GSIs Ying Cao and Meng Qi for giving me the opportunity to take such a wonderful class and learn so much in the process.

V. Appendix

A. Variable Descriptions

Variable No.	Variable Name	Variable Description
1	site_url	url of the base website
2	author	Author of the article
3	headline	headline of the article
4	language	language of the article
5	country	base website's country
6	global_rank	global rank of the website
7	US_rank	US rank of the website
8	Top1_country	country producing highest prop. of traffic at this website
9	Top2_country	country producing 2nd highest prop. Of traffic at this website
10	Top3_country	country producing 3rd highest prop. Of traffic at this website
11	Bounce_rate	percentage of visitors who navigate away from site after viewing only one page.
12	Daily_page_views_per_visitor	the average number of pageviews per visit per day
13	Daily_time_on_site	Average amt. of time people spend on this website
14	Percentage_search_visits	Percentage of traffic coming from search engine for this website
15	Num_sites_linking_in	No. of sites linking in to this website
16	gender	Predominant gender of those who use this website
17V	Educational_level	Predominant educational level of those who use this website
18	Browsing_location	Predominant browsing location of those who use this website
19	Stumble_upon	No. of 'stumbles' onto this site
20	pinterest	No. of pins onto this site
21	linkedin	No. of LinkedIn shares of this site
22	Fb_comments	No. of comments for this site
23	FB_shares	No. of shares for this site
24	FB_reactions	No. of reactions for this site
25	sentiment	Sentiment of headline
26	Various NLP features	NLP features indicating presence/absence of a characteristic stemword
27	type	Type of article (Dependent Variable)

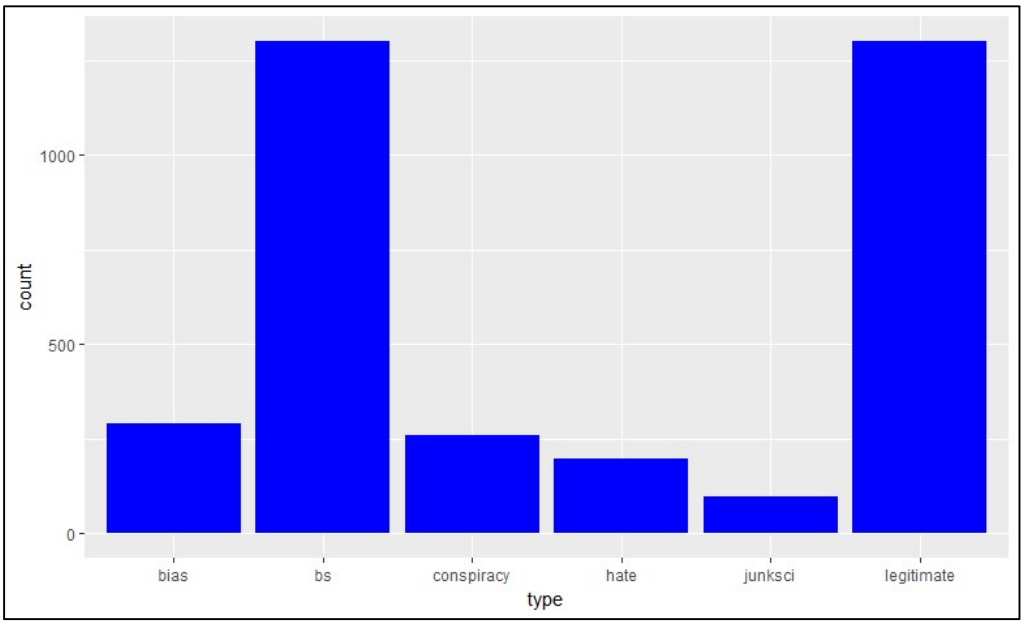
B. Various Component Model Performance

Type	Model	OOS Accuracy
Stacked Model	Stacked Ensemble Model	~1
NLP	Multinomial Logistic	0.462
	Linear Discriminant Analysis	0.459
	Support Vector Machine	0.448
	CART	0.449
	Random Forests	0.450
	Naïve Bayes	0.097
Webstats	CART	0.998
	Random Forest	~1
	Support Vector Machine	~1
Socialmedia	Random Forest	~1
	CART	~1
	Support Vector Machine	0.882
	Linear Discriminant Analysis	0.799

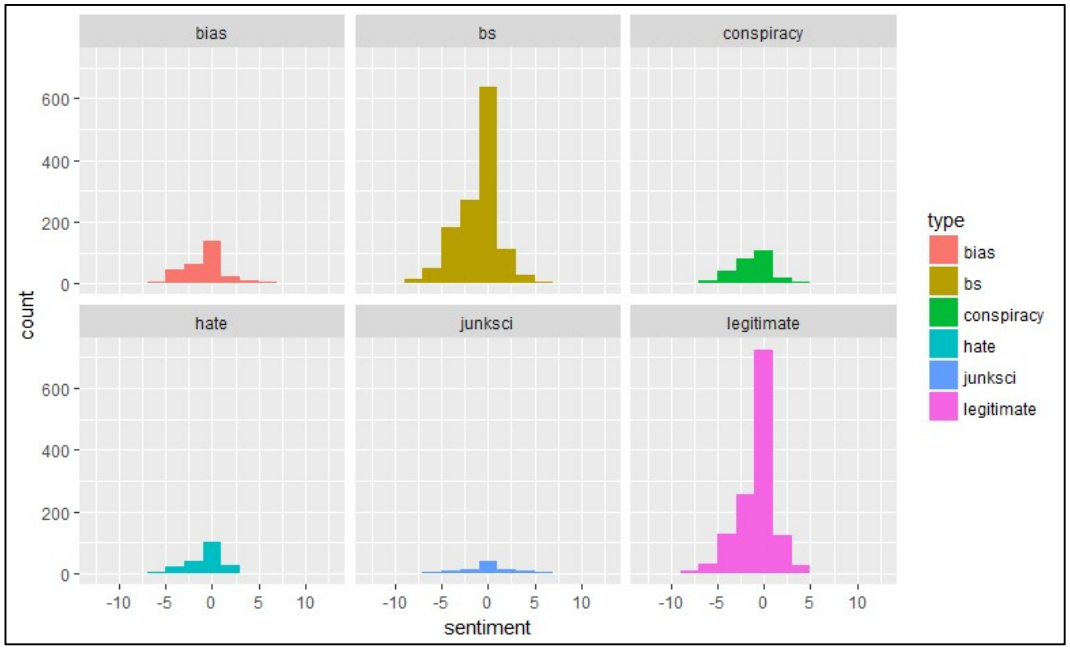
C. Confusion Matrix for Final SEM

```
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
           bias  bs conspiracy hate junksci legitimate  Error        Rate
bias         87   0          0    0       0          0 0.0000 =    0 / 87
bs            0 390          0    0       0          0 0.0000 =   0 / 390
conspiracy    0   0         78    0       0          0 0.0000 =    0 / 78
hate          0   0          0   60       0          0 0.0000 =    0 / 60
junksci       0   0          0    0      29          0 0.0000 =    0 / 29
legitimate    0   0          0    0       0        391 0.0000 =   0 / 391
Totals       87 390         78   60      29        391 0.0000 = 0 / 1,035
```

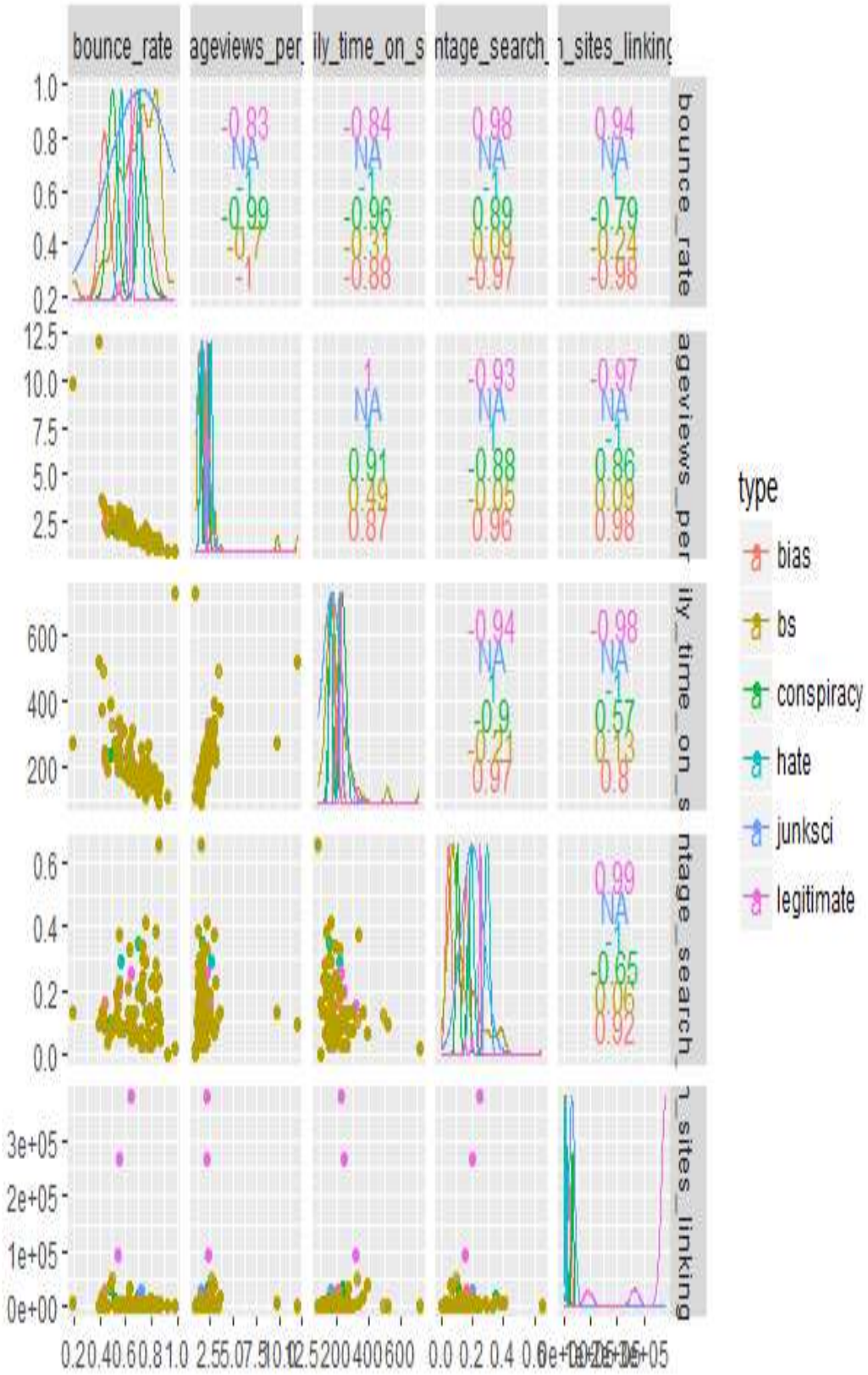
D. Class Distribution



E. Histogram of Sentiment by Type



F. Scattermatrix by type (for webstats variables)



G. Scattermatrix by type [Social Media Variables only]