Duke University
Everything Data - CS 216
Team 23: Jess Beering, Molly Carmody, Bella Hutchins, Ryan Nicholson, Catalina Sanchez-Carrion
30 April 2018

Predicting Popularity in New Media

The Problem

Social media today is one of the most prominent ways people find important information about our world regarding politics, the economy, social issues, and more. The conflation of social media and news reporting is an especially contentious issue since the 2016 U.S. election, given the impact that social media posts are reported to have had on the election results. News and media outlets aim for their posts to reach the largest number of people as possible, and thus have a high popularity among social media users. There are various factors that the popularity of an article depends on, including the day and time it is published, the subject matter, multimedia within the piece, and more. Therefore, we are interested in studying the impacts that factors like these have on the popularity of articles in order to predict how many shares an article will get, and thus, how popular it will be.

Along with deciding if an article will be popular overall, determining the most important features of a popular article is valuable. Knowing which features correlate to an article's number of shares would provide insight to social media platforms as how to craft a popular article.

Related Work

Considering the high usage and dominance of social media in society today, many similar studies have been conducted to determine the perceived popularity of a post. There is one study in particular that we feel is most important to highlight, and have referenced other related works in our References section. The study we will expand on is by HP Labs. In this study, HP Labs analyzed a number of features in order to understand what caused a certain news article to be posted on Twitter more than others. This study used four features (news category of the article, whether or not the article is subjective or objective, how many known names/entities are mentioned in the article, and what the source of the news is) in order to predict the number of times an article would be tweeted. HP labs used a Regression to do so, and ultimately found that the feature that most impacted the number of times an article was tweeted is the source of the news.

Our approach was somewhat similar to that of HP Labs because both of us used news category and the presence of known names (what we call 'trends' in our study) as features for determining popularity. Additionally, we both used a regression model. One difference between our study and HP's study is that, in addition to measuring known names and entities, we ensured that these known entities/names were trending on Google during the time that the articles were published, as we believed this would give us the most accurate measurement of what topics viewers wanted to read about. Additionally, HP's lab failed to incorporate the readability of articles as a feature, which (as we will later explain), we found to be our most important feature. Finally, while HP Labs only used a Regression

model, we used a Naive Bayes model and a Random Forest model so that we could compare the accuracies of these models against each other.

Our Approach

Our approach was to use different classifier models to evaluate which article features most affected the number of shares. We used three classifiers: Naive Bayes, Regression and Random Forest. The classifiers were trained using a subset of the original data where the number of shares and attributes were known.

Our Goals

We established 3 main goals:

1. To explore which attributes would best predict the popularity/number of shares an article receives.
2. To explore the accuracy and strength of three different predictive models: Naive Bayes, Regression and Random Forest.
3. To assess the importance/relevance of the number of Google Trending Topics that an article contains and the correlation to number of shares.

Data Sources

We used data from the following sources:

1. UCI Online News Popularity Data Set: a dataset that contains over 60 article attributes for over 39,000 Mashable News articles
2. Google Trending Topics (per month for 2013 and 2014): list of top Google searches for given month and year of various categories

Factors/Features

Below is a list of factors we included in our classifier:

- Week day posted
- Number of visual aides
- Category (lifestyle, entertainment, business, social media, technology, world)
- Trending Words Count (context and title)
- Automated Readability Index (ARI)

Process

Tools

Below is a list of Python libraries we worked with:

- Python Scikit-learn
- Python Pandas
- Python Numpy
- Python csv
- Python matplotlib

- Python newspaper3k

Data Cleaning

The original UCI dataset was relatively clean and contained over 60 article features. We focused on a subset of these features that we deemed important to popularity. Therefore, we created a more concise csv file that only contained the factors of interest (listed above); this was utilized for the predictive models. Also, some urls in the file were "dead," meaning that they led to an article that was no longer posted on the website. This problem was encountered when running the newspaper3k python package in order to abstract the article content from the url. "Dead" urls were deleted in all csv/data files.

Adding More Features

1. Trending Word Counts in the Content and and in the Title

The interaction between social media and news is ever increasing. We wanted to determine how articles having trending words would affect their popularity and how much more accurately this feature could predicate popularity in comparison to other features. In order to create this feature, we counted the number of trending words (from the article's publication month) that appeared in an articles' content or title.

To collect the trending words, we used Google Trending Topics for each of the 24 months (2013-2014). This included two parts: 1. The top ~50 trending topics searches of that specific month and 2. The top 5 out of 10 of several search categories (such as books, actors, movies and others). We then added approximately 5-6 synonyms to each of the trending words. This was done to mimic how someone would search for an article on a search engine like Google. Most likely, they would type in a variety of words or phrases relating to one relevant topic or event. For instance, for news about the environment, one could search climate change while someone else could search 'C02 emissions' and another 'global warming'. Although all users in this instance are looking for an article on the 'the environment,' they use different words to get there. Ideally, an article that had the most trending words would come up first in the search engine/be the most appealing to the searcher and thus be the most popular.

Once the trending word spreadsheet of all 24 months was created, we then collected the publication date and content of each article. This task was accomplished using the newspaper3k python program, which extracted the needed information from the given url. Looping through all ~39,000 articles, we then created a csv file containing the article url, publication date, content, and title. After both csv files of the article content and of the trending words for the 24 months in 2013-2014 were created, we could count the trending words in each article. We used the countVectorizer, with vocabulary = to the trending words from that article's publication month and the text = to that article. After doing this for all the articles in the dataset, the result was a final array of each article's total number of trending words in its content and an array of each article's total number of trending words in its title. This was then copied to a csv file to be used for features.

2. Automated Readability Index

In addition, we created a third feature for the automated readability index (ARI) for the content of each article. The Automated Readability Index is a universal benchmark test for English text. Using the formula below, The test yields a score that indicates the estimated reading level of the text. For example, a

score of roughly 5 shows that an article is fit for fourth or fifth grade readers, whereas an article with a score of 14 is fit for college-level readers. We assigned each article a score and used this as a feature to see if there was a correlation to an article having a high number of shares.
Formula used: [1]

$$4.71\left(\frac{\text{characters}}{\text{words}}\right)+0.5\left(\frac{\text{words}}{\text{sentences}}\right)-21.43$$

Results and Evaluation

I. Naive Bayes

Explanation

1. *General Idea/code*

The Naive Bayes algorithm utilizes Bayes rules to predict a label given a feature set. We conducted Naive Bayes using sklearn on our dataset using three features with multiple subcategories to predict whether an article is unpopular, normal, or popular. We denoted an unpopular article with less than 957 shares with a 0, a normal article having between 957 and 2800 shares with a 1 and popular having over 2801 shares with a 2. These separation ranges were determined using the percentile function within Python's statistic library. The range of shares for the bottom 25th percentile was [1, 957], for the middle 25th to 75th percentile [957, 2800], and for the top 25th percentile [2800, 843300]. In the actual code, training data sets and test data sets are created using the feature and target values (popularity score of 0, 1, or 2). Using the training data sets and the Naïve Bayes algorithm, the probabilities of an article having a popularity score of either a 0, 1 or 2 are calculated using the feature values. Using these probabilities, the popularity values are predicted for the test data set and compared to the target test data set to determine the accuracy of the predictions.

2. *Features we used*

We used 18 features overall, with 5 main categories.
Our features were:

- Number of visual aids (images, videos),
- Article category (lifestyle, entertainment, business, social media, tech, world),
- Day of the week posted
- Trending Word Count (for the content and for the title)
- Automated Readability Score

Predictive Model Results

1. *Overall Accuracy of Naive Bayes Classifier Using All Features: 38%*

This accuracy is very low, which signifies that Naive Bayes is probably not the best prediction model to use for our data.

---

[1] "Automated Readability Index."*Wikipedia*, Wikimedia Foundation, 10 Apr. 2018, en.wikipedia.org/wiki/Automated_readability_index.

Below is a chart comparing target popularity values with the prediction popularity values from the classifier.

Target Popularity Values	Predicted Popularity Values
[1. 1. 0. 2. 0. 2. 2. 0. 2. 2.]	[2. 2. 1. 1. 0. 2. 2. 0. 1. 0.]

2. *Anything surprising*

Initially we had assigned only a popularity score of 0 or 1 to each article using the median number of shares (1400). This generated an accuracy score of 54%. However, considering the span of the number of shares, we decided that breaking the popularity score three ways based on percentiles was more sensible.

3. *Naive Bayes on individual categories*

Furthermore, we performed Naive Bayes on each separate feature category (visual aides, article category, day of week, trending word score, and readability score) to see if the accuracy score differed among them. The chart below shows that the accuracy was highest when only predicting based off visual score, trending word count and readability score, which means these features have a higher relation to the popularity prediction of an article.

Features	**Accuracy of Share/Popularity Prediction**
Visual Score	0.4942973454900246
Data Channel	0.39706624499545806
Day of Week	0.3237560138613195
Trending Word Score	0.5068129058305016
ARI score	0.5032466440130539

II. Random Forest

Explanation

1. *General Idea/code*

The Random Forest algorithm uses a "forest" or group of decision trees in order to form a prediction based off of a set of features. Our Random Forest model used the same popularity index as described in the Naive Bayes section--the only difference is that in the Random Forest model, 1 denotes unpopular, 2 denotes normal, and 3 denotes popular. The reason for using these numbers instead of the 0, 1, 2 index used in the Naive Bayes section is simply because the formula that we used in the Random Forest code to calculate its accuracy threw an error when dividing by 0.

As for the design of our Random Forest, we decided to use 1000 estimators (decision trees), a test size of .25, which is the default for Random Forest, and a random state of 42, which is also the default for

Random Forest. The reason we decided to use 1000 estimators is due to the huge size of our dataset. During "training," the Random Forest model uses decision trees based on the importance of the features, which can be described as the impact certain features have on the target. During testing, the model then uses this information to form predictions on the data.

2. Features we used

We used the same features in the Random Forest model as we did for the Naive Bayes model so that we could more easily compare the outcomes of the two. So, to reiterate, we had 18 features that were broken down into 5 main categories:

- Number of visual aids (images, videos),
- Article category (lifestyle, entertainment, business, social media, tech, world),
- Day of the week posted
- Trending Word Count (for the content and for the title)
- Automated Readability Score

Predictive Model Results

1. *Overall Accuracy of Random Forest Classifier Using All Features: 63.18%*

The accuracy score of 63.18% for our Random Forest model indicates that this model produced the most accurate predictions of all of the models we used for this project.

Below is a chart comparing target popularity values with the prediction popularity values from the classifier.

Target Popularity Values	Predicted Popularity Values
[1 1 2 2 1 1 1 1 3 1]	[1.331 2.281 2.036 2.371 1.908 2.319 2.147 1.385 2.508 1.998]

2. *Anything surprising*

Prior to creating the popularity index of not popular, normal, and popular, our Random Forest model was the most inaccurate of all of our models (by far). Without these indexes, we were attempting to use Random Forest to predict the exact popularity of the articles, which is too specific of a prediction for this type of model. Not only did this cause the program to run for so long that the kernel almost always died, it also gave us a terrible accuracy of -102% and our predictions were, on average, 3,000 shares "off." Adding the popularity index improved our accuracy immensely.

3. *Random Forest on individual categories*

Finally, we calculated the "importance" of each individual feature so that we could understand what features of an article impact its popularity most:

Features	Importance (in percentage)
ARI	49%
Trending words in content	13%
Number of images	10%
Number of videos	5%
Trending words in title	3%
Data channel	2% (for news+entertainment) or 1% (for all other sections)
Day of the week	2% (for weekdays) or 1% (for weekends)

Regression

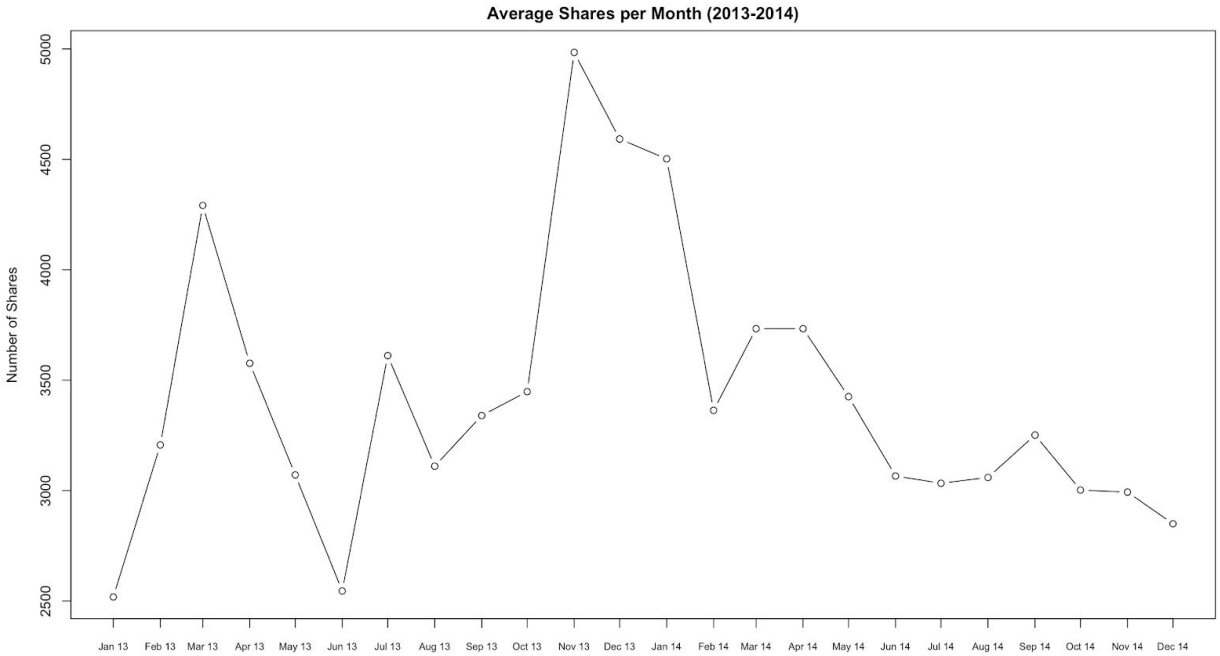
Predictive Model Results

1. *Adjusted R-Squared for our OLS multivariate linear model: 0.0613*

Performing regression analysis on our cleaned dataset revealed some interesting quirks that it contains. The analysis uncovered the extreme variability of the number of shares for our entries. Number of shares appeared to have a very strong right-skew.

We tried to normalize this variable, as it was set as our dependent, by performing a log transformation. This did normalize the distribution slightly, however due to the outliers lying above 500,000 shares, the data was still skewed to the right. These outliers are points of high leverage and negatively influenced the model. We were tempted to remove outliers, but chose not to because we wanted to preserve the true nature of the dataset.

Another quirk that influenced the linear model was the variability of shares over time. The figure below shows the polarity of number of average shares per month over the time period we were studying.

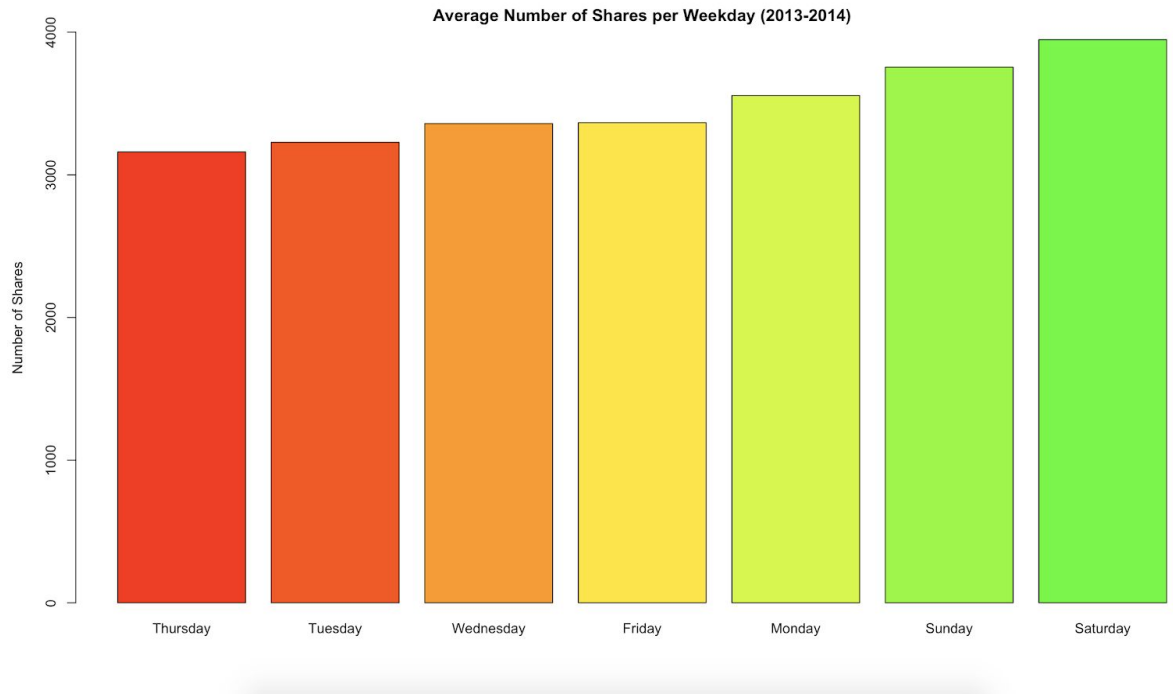


There are a couple obvious outliers (January 2013, March 2013, June 2013, November 2013). The variability of average shares per month posed an issue when trying to predict shares. Since we used our trending topic analysis in a monthly time frame, the correlation coefficient for title_trending and content_trending is greatly influenced by these outliers.

2. *Anything surprising*

The most surprising finding from regression analysis was the variability in article shares. At first, one may conclude that much of this variability can be explained by the features in the dataset. After conducting exploratory data analysis, the features can only explain 6% of the variability in article shares.

Conceptually, the high variability and extreme outliers could be due to the shares themselves. Sharing articles is an example of a positive feedback loop. The article starts with zero shares when it is published, then an individual decides to either share or not share the article. As more people share the article on social media, it is then exposed to new constituencies, who may or may not decide to share the articles themselves. Of course this is just speculation, however it has potential to explain the low R-squared value.



Another aspect of the analysis that surprised me was the average number of shares based on day of the week it was published. Initially, we hypothesized that there wouldn't be a significant difference in number of shares by weekday.

After running an ANOVA test, Saturday proved to be the day that yielded the highest shares. Although correlation does not mean causation, individuals usually have more free time on the weekend to read/share Mashable articles.

3. *Regression on individual categories*

Due to our unexpected results from our ordinary least squares regression, we decided to analyze the predictor variables individually against the number of shares. We still had to perform a log transformation on the number of shares to normalize the distribution when conducting these tests.

Automated Readability Index (ARI) was the strongest feature at predicting the number of shares. The ideal ARI range for a popular article is between 16 and 22, which indicates that most of Mashable's content is aimed towards the young-adult age group.

Closely behind the ARI is the number of trending topics in the article content. We got a positive, (fairly) linear relationship between number of trending words in the article and log(number of shares).

The number of images and videos in the article were also positively correlated with log(number of shares).

If a Mashable author wants their articles to become more popular, we advise them to publish an article on a Saturday with numerous images/videos, an ARI between 16 and 22, and include trending topics within the title and content. According to our model, these are the optimal characteristics for an article to become popular.

Conclusion/Summary of Findings

After we finished running our predictive models, we believe there could have been a few improvements to achieve better results. These improvements include, adding more trending words and synonyms, matching trending words per day rather than month, using twitter trending hashtags, and considering different news sources such as New York Times or Buzzfeed.

From analyzing the results of our predictive models, we determined the ideal article to be one with the following trends.

1) Readability Index - Between 16 and 22
2) # images in the content - more is better
3) # videos in the content - more is better
4) # trending words in the article - more is better
5) # trending words in the title - more is better
6) News Category: Lifestyle
7) Posted on: Saturday

Considering social media is the prime source of news and information for our generation, understanding article popularity is important for writers to reach the largest possible audience. Our conclusion for the features of an ideal article provide insight into how writers can achieve the most popularity possible on various social media sites.

References:

[1] Bandari, Asur, Huberman, “The Pulse of News in Social Media: Forecasting Popularity,” HP Labs.
[2] Ren, Yang, “Predicting and Evaluating the Popularity of Online News,” Stanford University, 2015.
[3] Geng, Yuan, Wang, “Predicting Popularity of Posts on Hacker News,” CS229, 2016.