**Online News Popularity**

Lynell Amanna, Bhuvana Bellala, Shirish Dhar, Jeanie Oh, Pratibha Rathore

Info 290T, Final Report

May 6, 2016

**Introduction -**

“What makes a site go viral?” Intuitively, we grasp this question to have most significance in marketing. For instance, by knowing what site is most likely to be popular, we can use that information to sell body lotion or spread the word on a fundraising event. This is not trivial as it can result in successful ad campaigns generating tremendous profit margin results. However, this line of inquiry has an even deeper meaning when it comes to culture, especially as we are in the midst of the Information Age. Traditional means of information dissemination has transitioned from centralized sources of media content production and broadcasting to decentralized patterns of content distribution where each individual can become his or her own content distributor. The consequence is that each person is empowered to effect culture in a way that was not possible before. The resulting “networked society” as coined by Manuel Castells facilitates a “participatory culture.” In other words, this change in the broadcast landscape empowers people to shape discourse and society in ways that were isolated to the few broadcasting titans like FOX, NBC or CNN.

The term “viral” is not an adequate word to describe the popularity and rapid influence of media according to Henry Jenkins because it does not capture the aspect of human agency involved in something going viral. The analogy of viruses in a medical setting implies that the spreading of something is a passive process on the part of humans which leaves out the aspect of human agency. The alternative term coined by him “spreadable media” describes more aptly the phenomena of how media spreads through society with human agency dramatically altering the cultural landscape.

Before laws are passed, before policies have to be drafted, before politicians are elected, the fountain of culture is the source of true change. Being far more significant than just identifying the next successful marketing gimmick, the question of what makes a site more likely to spread through the internet is a vital question because it helps us understand culture and the different factors that may affect it. Furthermore, knowing these factors may also lead to discoveries in human behavior. Why does a particular feature result in higher probability? It can lead to numerous avenues of further investigation to account for why each feature was predictive for an outcome. For instance, one of the features

Our dataset takes over 35,000 instances of over 60 features of various online article. The target variable is the number of “Shares” that each site had. Number of shares is a proxy for how “spreadable” or popular a particular site is given the 60 plus features for that site. In the following sections we will introduce in more detail the nature of the dataset, describe our approach in finding valid conclusions, and explain what our findings implicate about spreadable media as it relates to the sites we studied.

**Dataset -**

We obtained the dataset from the UCI Machine Learning Repository. The dataset summarizes a set of attributes about articles that were published on Mashable over a period of two years. The goal of the dataset is to predict the number of shares (“Popularity”) of an article. These articles were published on [www.mashable.com](http://www.mashable.com/). The dataset consists of a set of computed statistics based on the content of the article. This dataset was acquired in January 2015. There exists a column that stores the url of the article where the article can be viewed.

The dataset consists of a total of 61 attributes one being the label column “shares”, and another the url to the article. Therefore the total number of predictors possible are 59.

The dataset included information about what day of the week the article was published (“Is\_weekend”, “Is\_monday”, “is\_tuesday”), the sentiment of the article (“global\_sentiment\_polarity”, “title\_sentiment\_polarity”), how positive or negative the article was (“rate\_positive\_words”, “rate\_negative\_words”, “max\_positive\_polarity”, “max\_negative\_polarity”, “min\_positive\_polarity”, “min\_negative\_polarity”)

**Secondary Dataset –**

In its current form, the UCI dataset is very granular and specific to the contents in the articles. It does not account for broader trends and the demographic interests that can make an article popular. For example, some articles might enjoy great popularity simply because the topic is widely trending. Given that it’s election season, a subjective article by a well renowned columnist about current candidates could get more shares than one in the non-election season. Considering these limitations of the dataset, we wanted to enrich it by building context around each data point. We researched into Twitter, Google, and BuzzFeed trends and found a tool called pytrends that taps into the Google trends API and returns the relative popularity of a token over a certain period of time. We scraped each Mashable article for the keywords and the date it was published. Then we used pytrends to obtain the relative interest of all the keywords over the thirty-day period of when the article was published. Relative interest stands for the number of searches for a term relative to the total number of searches done on Google on that day. Once we obtained the relative popularity for a thirty-day period, we found the average popularity of each keyword and augmented the dataset. Then we only picked five keywords with the highest popularity.

Attributes added –

kw1\_popularity: Popularity of the first keyword

kw2\_popularity: Popularity of the second keyword

kw3\_popularity: Popularity of the third keyword

kw4\_popularity: Popularity of the fourth keyword

kw5\_popularity: Popularity of the fifth keyword

**Introduction to approaches -**

Once we found our dataset, we decided to make our label binary rather than continuous because we wanted to focus on creating a “cheat sheet” that can be used to make sure an article gets noticed. We plotted a histogram of the distribution of shares and found that the data is skewed to the left, with the medium being 1400 shares.



Looking at the distribution, we decided to make anything with less than 1400 shares unpopular with a label of 0 and anything with greater than 1400 shares popular with a label of 1. We chose to not use mean for this dataset because the data is heavily skewed. We also thought about adding dimensions of popularity (‘low’, ‘medium’, and ‘high’), but for the purpose of our project, we wanted to solely focus on increasing the accuracy for binary predictions since this would allow us to narrow down what makes an article popular or not popular.

After working with the dataset, we wanted to increase the breadth of our project by adding context to each datapoint. We used Google Trends API to determine relatively how popular the keywords in an article were at the time published.

After initial exploration and deciding on the label and goal of our project, each of us set out to try different models with different approaches to feature engineering. In the following section, each of us will talk about the models we tried and the conclusions we came to.

**Individual Contributions** -

We had a lot of trouble finding the correct dataset. We all went out and found datasets and did some rudimentary explorations to determine if the datasets were viable for the project. Then, once we decided on the news popularity dataset found and pre-cleaned by Shirish, Data explorations were primarily done by Pratibha and Bhuvana. Shirish, Jeanie, and Lynell did most of the work on feature engineering. These are the models each of us tried once the data was cleaned:

Random Forest - Lynell Amanna

Decision Trees - Bhuvana Bellala

Logistic Regression - Shirish Dhar

SVM - Jeanie Oh

Naive Bayes- Pratibha Rathore

Once each of us ran our models, Bhuvana ensembled the different models by using majority voting. Lynell and Shirish looked at the different models, the different features we tried and came to a viable conclusion. Bhuvana then went on to utilize extrinsic features pertaining to global trends by merging the dataset with ‘Google Trends’ and worked on improving the accuracy.

In the following section, each of us talks about the contributions we made to the project.

**Individual Contributions**

**Jeanie Oh :**

We experienced some setbacks in terms of agreeing on a dataset that was interesting enough for our liking but also rich enough to justify applying machine learning methods. All of us contributed to foraging through the internet to find something unique that we could all get excited about. We narrowed down the candidates to two : one on credit default and the other, a dataset we ultimately decided pursue, on the popularity of online news articles from the Mashable website. I contributed by cleaning both datasets and analyzing their viability. I sorted on all the features in order to see if the data made sense. With the credit default data, I thought a more interesting question to ask than “Will this person default or not?” is “Can we predict the spending pattern of cardholders based on certain features?” I ruled out this line of investigation by noticing that the spending patterns were negative in many cases. Not just by a few dollars but by hundreds of thousands of dollars. Then, we realized that this is not in US dollar but in Taiwanese currency. Still, that meant some people were overpaying their bill by thousands of dollars. Why would anyone do that? Given the little we know about credit card companies in Taiwan and how lending works, we decided to choose the data set that had to do with online articles. My contribution was ruling out and helping decide on the data by doing an initial analysis on the various data sets.

Once we settled on the data set, my contribution was to research the greater implication of our data, particularly the target variable. Why does it matter how many shares a new article gets? Is there a greater implication to our data than just advertising and marketing? Last year, I took a New Media class called “Theorizing Pop Culture & Social Media” and remember a bunch of social science students in my class asking me if the I-school can make a tool for tracking how often a link is shared. This caused me dig through my notes and read through “Spreadable Media” by Henry Jenkins and sure enough there was plenty of relevance that a widely shared article has on culture and media.

We had decided to split up the work by choosing an algorithm we learned in class, standardize among ourselves the training set, cross-validation test data and final test data sets. Next, we decided to add a column that classifies the shares as unpopular or “0” for anything below 1400 shares and popular or “1” for anything above 1400 shares. My chosen algorithm was SVM or Support Vector Machine. I thought this wouldn’t be a bad model to use since its strength is with classification and regression problems. Since we are trying to find relationships that correlate to a binary target variable, SVM seemed like a reasonable bet.

The first thing I did was double-check for any null values and feature values that made no sense. The data was pretty clean so not much effort had to be spent in this area. Next, it seemed like the data was ordered in a certain way. I realized the rows were ordered by date. Therefore, I randomized the rows to make sure the cross-validation splits would yield representative samples. Next, 61 features is a lot! Where do I begin? How many features should I run through the model at a time? Should I just make some educated guesses? I did some trial and error and realized that the more features I included in the model, the better the accuracy. Regardless of whether the feature had any “common sense” causational relationship with the target. But does it make sense to keep adding features to the model? At one point, I had added 55 features to the model yielding a 59% accuracy on the test set using cross validation. Having a good accuracy is important but practically speaking, what good is it to have predictive power with over 50 different features? Given my accuracy, it’s easier to toss a coin than point to so many features for predictive purposes.

I had to make a decision on how many features is a reasonable number of attributes to point to and manage for the purposes of our study. How about 5? I can imagine myself walking into a room and make a presentation about the top five most important things to keep in mind when predicting whether an article will go viral or maybe up to 10 but not past 15. How do I find the ideal 5-15 feature sets to run through my model? I could have done a simple regression between each individual variable and the target but that was an unsatisfactory approach in my mind because 2 features that individually have a low correlation with the target may still have a high correlation with the target if they are combined. Therefore, I needed to take the combination of features into consideration during my selection process.

After a good deal of research on the internet, I found a function in SCIKIT Learn called “rfe()” that takes a model and iteratively runs through the features by the numbers we specify. For instance, if I specified the model to be “SVM” and the grouping to be “5.” The function will iteratively run through my 60 features 5 at a time and find the group that has the greatest effect on the target variable. I used a linear kernel at first since that is the most simple. The algorithm was taking more than 3 hours to run. However, it ran fairly quickly when I switched the model to decision trees or random forests. Therefore, I gave my code to other members of my team. All of them used the code and appreciated it as a great starting point for their analysis. Next I threw out outlier rows as there were a handful of rows that were skewing the data. Then I normalized the values using Z-Score so that the wide range of nominal values would not confound relationships that can be identified with the target. I did a good deal of research on what each of the features mean and took my best guess at which ones might be the most relevant like positive\_polarity which means whether the article have positive content. Ultimately, my accuracy would not pass 61% for 10 features that I chose.

Finally, what I did was ask the other members of my group who got the best accuracy. Lynell seemed to get the best results using Random Forests. Therefore, I ran my algorithm that ranks the best features using the random forest model to see which features would emerge. I took those features and ran SVM ‘rbf’ as this gave better results than ‘linear’. This pushed my accuracy up to 62%. When I chose an additional 5 features using trial and error in addition to the 10 that Lynell gave me, that boosted my accuracy level to 63%. As I stated initially, I did not want to choose more than 15 features to gain predictive accuracy for the target as it gets practically unmanageable after a certain point. The 63% accuracy was what I ultimately presented to my group. Since my accuracy was so low compared to my group, it was not added to the ensemble.

**Pratibha Rathore**

I worked on :-

1. Finding a potential data set to work on for the project
2. Exploratory data analysis for the popularity dataset.
3. Naives Bayes Classifier and Random Forest Classifier

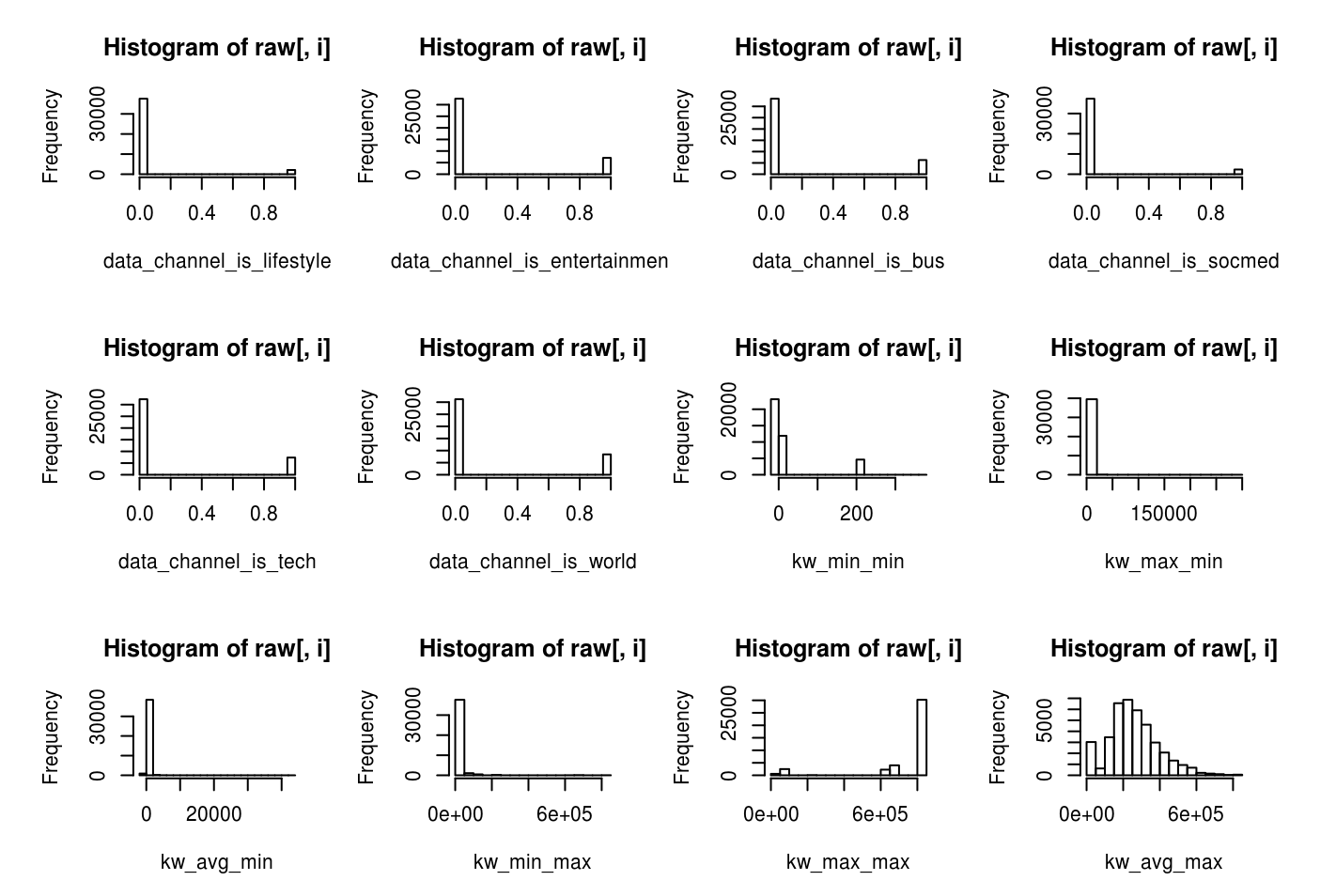
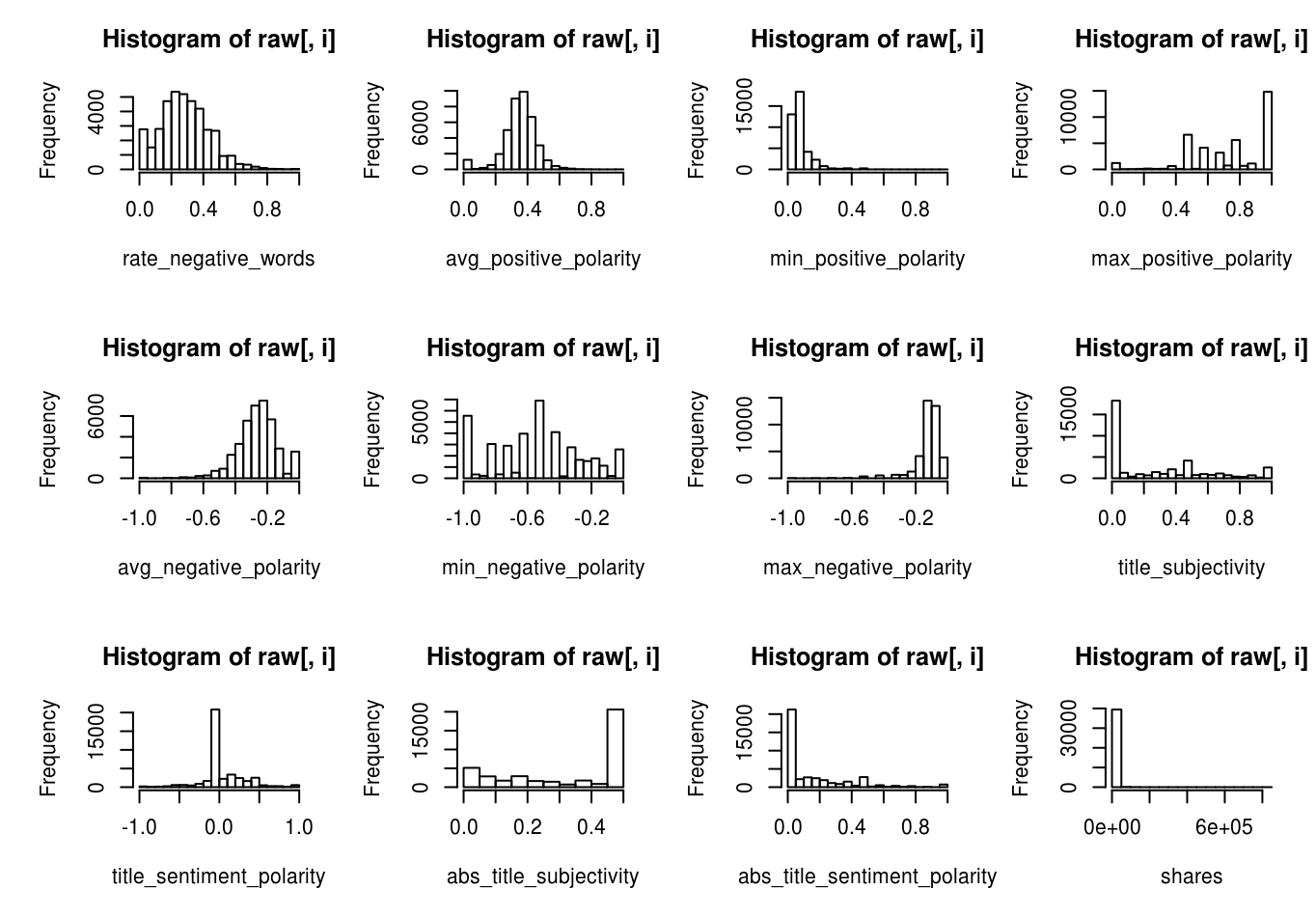
Finding a potential data set for the project:

Each of the team members worked either independently or with the group to research and identify a potential data set to work on. I found traffic violations in maryland data set available publicly and is provide by US govt under their open platform data services. The data set can be found here : <https://catalog.data.gov/dataset/traffic-violations-56dda>. The main idea was to do clustering to identify attributes which together contributes for more traffic violations. However, when discussing with the team, we decided not to go with this data set as it included attributes such as race which can have potential social and cultural biases and our data mining techniques would not be able to account for the biases. In the end, we decided to go with Online news popularity data from UCI repository.

Exploratory Data Analysis:

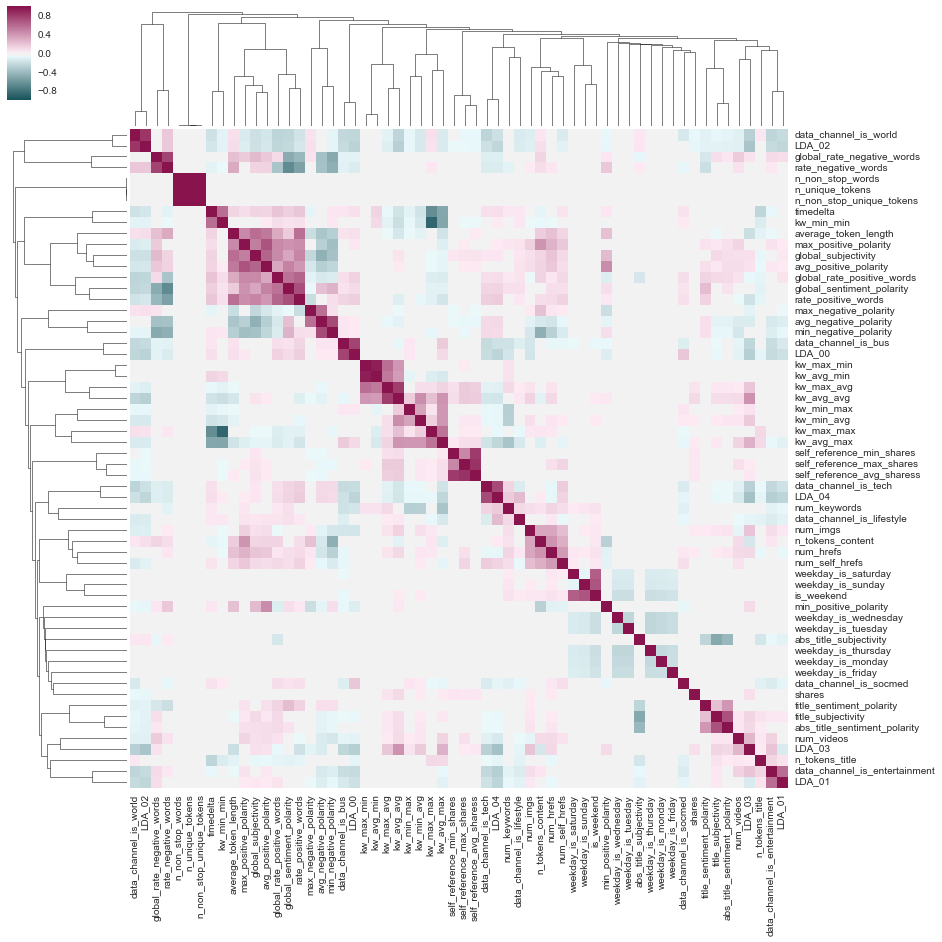
To start with, I wanted to see which attributes might have an impact on the popularity of an article.

I) Firstly, In order to see how the data for all the variables is distributed, I used ggplot library in R to plot frequency distribution histograms. The plots are shown below and the R script to produce these plots is attached in the attached project folder.



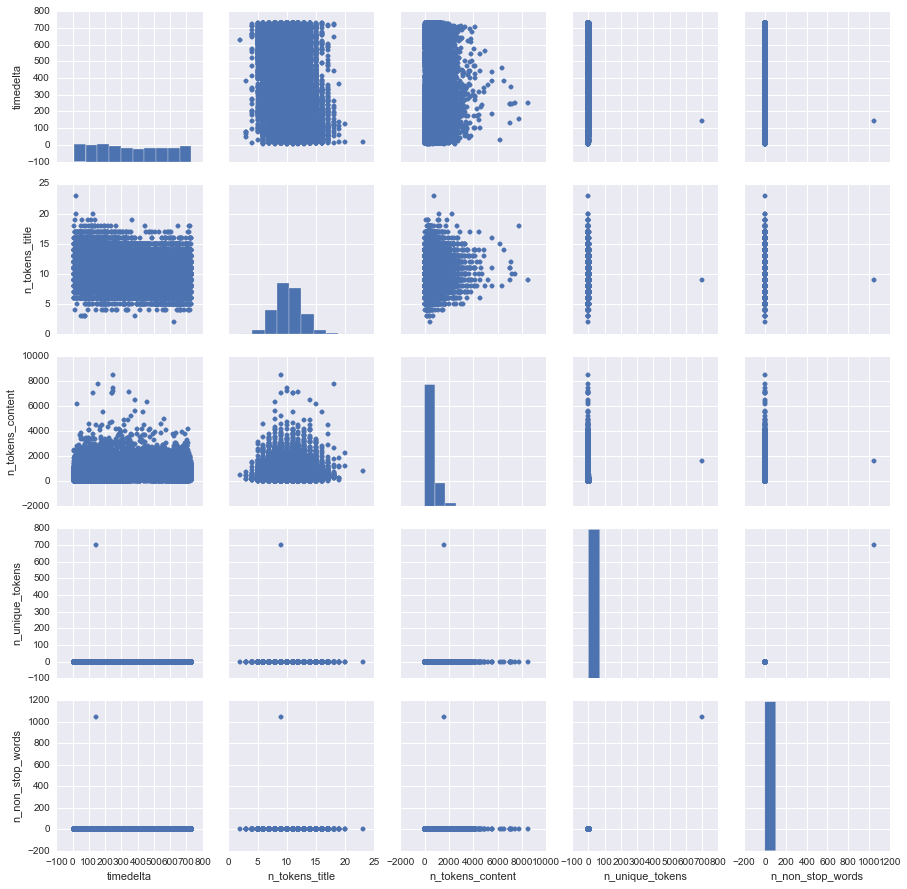
The frequency distribution histograms helped me understanding two important things about the data, first whether there are any outliers and second whether the data for any variable is skewed and needs transformation such as scaling or converting to different format. From the plots I found that, it would be better if we take log transformation of some of the variables such as num\_of\_self\_hrefs, average\_token\_length etc.

II) Secondly, to understand the association of each of the variables with other variables I plotted a ‘Cluster Grid’ also known as Correlation matrix or Heat Map using a python’s rich visualization library called ‘Seaborn’. The intention was to see if variables are independent of each other and if not how much is the dependence. The color coding shows the strength of association between any two variables as defined by the value of correlation coefficient reflected in the legend. The plot is shown below and the python script to make this plot attached in the attached project folder.



From the Cluster Grid, I found that most of the variables are not related to each other and are mostly independent.

III) Finally, in order to find potential predictors for my model I wanted to try different permutations and combinations of the features and so created pair plots again using Seaborn library. These are essentially scatter diagram between any two variables. The plot is shown below and the script is attached in the folder. Similar to the results in case of Cluster Grid, I did not find any two predictor variables that are strongly associated.



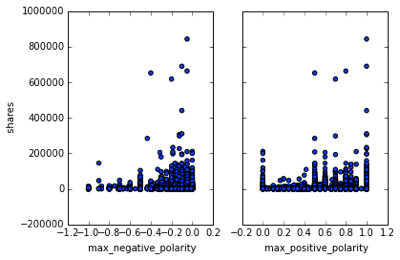
Naives Bayes Classifier:

Since, our team decided to turn the research question to a classification problem instead of a prediction one, I wanted start with using a simple Naive bayes classifier with all (except for url and timestamp) the features and found that the classifier gave me decent baseline accuracy for 61%. I used Naives bayes because it is easier to build and faster to run. Naives bayes assumes that most of the features are independent of each other and do not work well with highly correlated data and that was validated from my exploratory data analysis, where in I found that most of the variables were indeed independent of each other. It also makes sense to use a simple classification algorithm to test bunch of hypothesis and then other algorithms like Random forest can be used to make more robust model. In addition, Naives bayes do not need huge data to learn and classify, it just needs enough data to understand the probabilistic relationship of each attribute in isolation with the output variable, which goes very well with the amount of data we had.

**Bhuvana Bellala:**

Initially, we wanted to apply machine-learning techniques to college scorecard and campus crime data. I found the campus crime data, did some preliminary data cleaning and we merged the dataset with college scorecard. However, we realized that most of the crime data is “Not available”, 0 or “not reported”. Furthermore, the crime dataset does not contain information about the campus police or about any other crime stopping measures taken by the university. Without this information and the surrounding area crime rate, our conclusions would be very biased. So, Unfortunately, after spending a lot of time cleaning this data, we moved away and decided on news popularity dataset.

I helped with data visualization by plotting correlation graphs between the different features and the label (Not Popular – 0 or Popular – 1). These are some graphs that were helpful in deciding what features to focus on as we delved into optimizing our models.

max\_negative\_polarity/ max\_positive\_polarity vs. shares

Looking at these graphs, we hypothesized that the more negative an article was, the less shares it got and that the more neutral or positive an article was, the more shares it got. However, we wanted to test out our hypothesis because our dataset is biased, as there were fewer articles with max negative polarity and more articles with neutral polarity. One interesting thing about this graph is that popularity dips down between 0.0 and 0.5 positive polarity, but picks up again at around 0.5. Also most of the popular articles were either extremely positive or mildly negative.

These correlation graphs also made us aware of the biases in our dataset. One, there were more articles written in the weekdays than weekends. Two, most of the articles contained few hrefs or images. These kinds of observations helped me in normalizing the dataset and in designing my model. I chose to use decision trees because the model helps you visualize which features were helpful in making a prediction. Furthermore, if I were to try different features, then through these visualizations, I could see which features had the most impact in making a decision. As a baseline measure, I predicted the majority class (‘Popular’) for all labels and that gave an accuracy of 53.3%. Then, I divided the dataset into 80% training and 20% testing, used all the features for prediction and got an accuracy of 64.9%. I wanted to improve on this accuracy, so I tried normalizing the features and using different subsets of the features. Furthermore, I optimized the hyper parameters and used cross-validation to make the model more robust. I ended up with a final accuracy of 65.9%.

In the beginning, I took out outliers (any row with a column value of more than a Z-score value of 3). However, accuracy fell down to 49.5%. This is when I realized that these outliers were actually contributing to the predictive power of the model. The outliers were outliers for a reason, so I included them in my dataset and worked on normalizing some of the features. I plotted all the features to see the distribution of datasets and found that ‘timedelta’, ‘n\_tokens\_content’, and ‘n\_tokens\_title’ had the greatest variation and changed all the values to Z-score. This actually decreased the accuracy to 63.7%.

After deciding to not normalize the features, I went on to find the hyper parameters. I found that a max-depth of 5 and max\_leaf\_nodes of 10 gave the highest accuracy. I then trained the model using 5-fold cross validation, which still gave me an accuracy of only 65.1%. Now, I wanted to see if different combinations of features could increase the accuracy significantly. I tried only the sentiment based features such as “max-negative\_polarity”, “global\_subjectivity”, and so on. However, with only the combinations of sentiment based features, I could get the accuracy only up to 56.4%. This means that the content based features like “n\_tokens\_content” had significant predictive power. After trying different combinations of the features and the model given by Jeanie, the decision tree model gave an accuracy of 65.9%. The features I ended up using were “n\_tokens\_content”, “num\_imgs”, “data\_channel\_is\_bus”, “kw\_min\_min”, kw\_max\_min”, “LDA\_00”, “global\_subjectivity”, “global\_sentiment\_polairty”, “rate\_negative\_words”, “max\_negative\_polarity”, and “title\_sentiment\_polarity”.

Once everyone gave me their predictions using their best model, I used majority voting to increase the predictive power. I picked the three models with the highest accuracy, and we finally ended up with an accuracy of 68.2%.

We discussed a lot about why is there such a low accuracy and figured out that our dataset is not comprehensive. It was missing two key data values. There is no data about the demographics that read and share articles, and no data about what was happening around the world when the article was published. I went on to scrape keywords and date using the ’url’ of the article. Then, used pytrends to get a report of how relatively popular each of the keywords were at the time of publishing. After merging the keyword dataset with the popularity dataset, I ran my decision tree on the data to see if there would be an increase in accuracy. However, the new dataset with a decision tree only gave an accuracy of 64.0%.

**Shirish Dhar**

A brief overview of my contributions to this project is as follows:-

* Researched datasets and found the Online Article Popularity dataset
* Pre-processed the dataset to remove outliers and poorly recorded feature values
* Applied logistic regression model to the dataset
* Performed extensive feature engineering on the model to obtain the highest recorded accuracy out of the five recorded in this project

A lot of the initial time was spent on finding the right dataset that would be both exhaustive in terms of its quantity, but also capable of producing an exciting data mining problem. After ample research work going through repositories, I came up with the Online Article dataset from the UCI Machine Learning Repository.

After finalizing this dataset, I worked on pre-processing the dataset to locate and remove any outliers or poorly recorded feature values. At the very start, I realized that I will need to play around with the output ‘shares’ column, as having such highly skewed and dispersed values of ‘shares’ would not lead to much information gain for us. The ‘shares’ column had a minimum value of ‘1’ and a maximum of ‘850,000’. I decided to take an architectural approach and decided to split the ‘shares’ column on its median of 1400. This way, anything over 1400 shares would be deemed a ‘popular’ article, while anything below 1400 ‘shares’ would be deemed ‘unpopular’, thereby reducing the high variance. This suggestion was utilized by the entire team, and helped our team narrow down our focus to just two output possibilities- ‘popular’ and ‘unpopular’. I also eventually realized that this bifurcation would lead to a very low threshold level at the 1300-1500 interface, and worked on changing that. I then went on to remove any outliers in the features by using data visualizations and distributions. This helped increase the eventual accuracy of my model. My pre-processing techniques also involved the employment of normalization to the features. After trying several iterations of logarithmic and z-score normalization, I found out that normalizing the features using z-score provided the highest increase in accuracy. Thus, z-score was the final method of normalization that was used with my model.

Following the pre-processing, I ran different algorithms on the pre-processed data to choose one model that would take my pre-processed data and offer the best accuracy. I utilized two models - SVM and logistic regression. The accuracy I obtained with my logistic regression model was much more than with SVM, so I decided to go ahead with the logistic regression model. Following this, I worked extensively on feature engineering to obtain the optimum set of features and sub-features that would affect the output the most. My first step was to plot scatter plot visualizations in python to obtain correlations between different features and the output. I performed this to gain a high-level idea of how certain features affect the output independently. Knowing that combinations of features can also unlock hidden correlations, I went on to create an iterative loop algorithm that takes in a maximum of 20 features and checks their relevance to the popularity with different possible combinations. This helped me gain an idea of what combinations of features might give us the best prediction accuracy. All iterations, aforementioned and subsequent, were done along with 5-fold cross validation to ensure the most accurate trainings, with a 70-30% training-testing ratio. One of the biggest reasons that I was able to boost up the logistic regression accuracy to the highest number amongst all the models was by combining different features into one single feature. Think of this as combining a ‘number of siblings Person X has’ feature with a ‘number of living parents Person X has’ feature, adding one to them, to obtain an all new ‘Family Size of Person X’ feature. I believed this helped increase my accuracy because some times combining two features can unlock hidden patterns that they individually would not have unlocked. Thus, I combined different combinations of features into one new feature and checked for increases in prediction accuracy. This extra feature that I incorporate into my training set increased the accuracy of my logistic regression model by 2.4%, resulting in an eventual accuracy of 67.3%.

Once the best models were combined in an ensemble to increase the accuracy to 68.2%, I started to look for reasons regarding why the prediction accuracy ceiling of this project is below 70%. After a lot of online research on news article datasets, I found an article that was written by a UCLA Phd. student on online news datasets. The individual had observed that a lot of the success of news articles depend on the kinds of global events happening at the time of publication. This was the instance I realized that there might be a missing ‘fifth wheel’ in our dataset that has relevance to how popular an article can be. After sharing my observed findings with the team, we were able to pivot in terms of our thinking and realize that the intrinsic features of a news article, like its polarity or genre, can only go to a certain extent in terms of predicting how popular an article can be. This post-coding analysis I conducted helped the team reach a valid conclusion point, where we understood why the intrinsic features in our dataset could only help us predict the popularity of an article with a maximum accuracy of 68.2%.

**Lynell Amanna:**

Each member of the group was assigned the task of looking for an appropriate data set to work with. I was able to obtain a UCI Machine learning data set for credit card default where a possible research question would be “who was most likely to be a defaulter in the next credit payment cycle”. However after further discussion it was decided that the Online Popularity dataset had an interesting problem to work with and we decided to select that as our project dataset.

*Model Selected*– Random Forest

*Data pre-processing for Random Forest:*

Data pre-processing was divided in two parts: Feature Selection and Normalization of Select Features.

I first divided the data set into training and testing data. I used the first 25000 rows as the training set and the rest of the rows as the test set. The output column “shares” is used to determine the popularity of the post. The shares column was converted into a popularity column with binary values 1 being popular and 0 being unpopular. The articles that received 1400 or more shares were coded to 1 and those that received less than 1400 were coded to 0. The popularity column was now used as the class label.

In order to arrive at the most optimal set of features for the model, I tried various different feature combinations. I started out with selecting all possible features to be included in the model, which resulted in an accuracy of 63% I then reduced the feature set by rerunning the model each time a feature was removed.

For those features whose values were large numbers I defined a function “take\_log()” that would convert the given column values into its respective log values. For those values where taking the log would result in a positive or negative infinite value, I assigned a NaN value which was then coded to zero. Certain features such as “timedelta”, “ self\_reference\_min\_shares “, “ kW\_avg\_min”, “kW\_avg\_max”, “kW\_avg\_avg”, had large values and needed to be normalized. The task of normalizing these values resulted in increasing the accuracy by 1.5%. The normalization was performed on both the training and testing sets.

I defined a list called “predictors” which included the set of features that would be used to train the model. While trying different combinations of features I found that features such as “maximum\_negative\_polarity” or “maximum\_positive\_polarity”, “is\_weekend” , “global\_rate\_positive\_words”, “ global\_rate\_negative\_words” and “num\_hrefs”, had a greater positive impact on the accuracy while certain columns that held information about an article being published on a weekday such as “weekday\_is\_Monday”, “weekday\_is\_Tuesday”, “global\_sentiment\_polarity” resulted in bringing down the accuracy by 1% - 2%

*Model and hyper parameter tuning:*

For the random forest model one of the key parameters is the number of estimators. In order to find the optimal number of estimators I selected a range of n = {1,100} and ran a for loop to find the accuracy of the random forest for the training set for that particular value of n. Using “sklearn cross validation” library to partition the training data set into 10 folds to achieve the average accuracy of the 10 folds for each value of the estimator.

For each value of “n” I computed the accuracy using the score function and stored it in a temporary variable. When higher accuracy is found for a particular value of n that value is stored in the variable. At the end of the loop, the value of “n” that obtained the highest accuracy is stored in the temporary variable. This value of n is then used as a parameter to define the Random Forest classifier. Once the classifier was defined I trained the model on the training set data with the specific value of n and predicted outcome of the model. Tuning the hyper-parameter helped to increase the accuracy by 2%

I also found that feature combinations that worked well for other models such as Logistic regression and SVM did not necessarily perform well on the random forest model. Although I started out with the entire feature set of 60 attributes I was able to narrow down the list of predictors to 28, where further removal of features resulted in reducing the accuracy by at least 1%.

*Observations:*

It was found that Random Forests resulted in one of the highest accuracies, 67% with all the above methods including feature selection and normalization and tuning the number of estimators to find the value that resulted in the highest accuracy for that feature combination. The final model with the highest accuracy had 28 features and n = 88.

I found that the fact that an article was published on a weekend increased in some way the probability that the article would be popular. Articles that had a strong sentiment, high positivity or high negativity were also more likely to be shared. These features were found to have the highest predictive power in determining whether an article would be shared or not. While such features may be useful to writers and publishers, other aspects of a viral article need to be examined as well. Is the topic being talked about in the media currently? Is it a highly controversial topic and is more likely to be shared due to the shocking nature of its content? Is it about a famous public figure that people want to know about? All of these aspects must be taken into consideration when trying to determine if an article will be a popular one or not.

**Results -**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Naive Bayes | SVM | Decision Trees | Random Forest | Logistic Regression | Blending (Ensemble) |
| Accuracy | 61% | 63% | 65.9% | 67.2% | 67.3% | 68.2% |

Naives Bayes proved to be the model offering the least accuracy, followed by SVM. Logistic Regression was the model that offered the highest accuracy, followed by Random Forest and Decision Trees. We then decided to utilize these three models and get the best predictions using a ‘majority-vote’ ensembling model. Blending the three aforementioned models with equal weight helped us increase our prediction accuracy to 68.2%.

In terms of feature selection, three broad categories of features proved to have a very tangible effect on the popularity of an article. These were:-

|  |  |
| --- | --- |
| **Feature Category** | **Examples of features** |
| Sentiment of Article | ‘maximum positive polarity’, ‘Maximum negative polarity’, ‘rate\_positive\_words’, ‘title\_positive\_polarity’ |
| Genre of Article | ‘is\_entertainment’, ‘is\_business’, ‘is\_socmed’ |
| Day of Publishing | ‘weekday\_is\_friday’, ‘is\_weekend’ |

**Inference and Conclusion -**

We were able to find strong correlations between certain feature categories and how they affected the popularity of the article. That said, we wanted to take our project a step further and look at the broader picture in question. Through this project, we were able to pinpoint the types of features that were most relevant to the popularity of an article.

We also acknowledged that the accuracy ceiling of close to 68% indicated that there might be other factors in play that cause an article to be popular or unpopular. This is where the significance of ‘context’ comes in. Descriptive features like the sentiment of an article, its genre, and its day of publishing are all ‘intrinsic’ features of the article. The popularity of an article is based on both intrinsic and extrinsic features. Different articles are surrounded by different world events and themes around them, and these factors extrinsic to the article can go a long way in deciding their popularity. For example, there is a good probability that a political article would receive much more shares if it is surrounded by an election season or political event during its time of publishing. Thus, it is imperative to utilize and consider extrinsic context based features that elaborate on the global environment around an article durings its publishing.

We added extrinsic features to our current dataset by polling Google Trends for the keywords in the article. However, we did not really see a significant increase in accuracy. The reason for this could be that the keywords were not representative enough of the article. For example, some of the keywords for the article were “uncategorized”, “US World”, “entertainment”. These keywords were not really representative of the article. Furthermore, we only looked at the trends for the month the article was published and then averaged the relative interest in the term. However, this dataset was put together in 2015. So, the relative interest in a term might have gone up and down over time and contributed to the popularity of an article. In the future, the dataset can be further enriched by using non-stop-words and most commonly occurring tokens in the article to poll for google or twitter trends. Additionally, we could obtain demographic data about Mashable users and create a cheat sheet for popular articles based on the age and gender of the users.

**Real world implications -**

From our project, we found that features related to sentiment, type of article, and day of publishing (weekday vs weekend) had pronounced impact on the popularity. By focusing on these subset of features, journalists and Mashable publicists could potentially improve the visibility of their articles. For example, we found the global\_subjectivity of an article had a high predictive power. The more objective an article was, the more it was shared. Mashable could use this information to edit their articles to be less subjective and increase the popularity of the article. Furthermore, our analysis could be leveraged to other online newspaper agencies such as New York Times to engage more readers in an increasingly networked world.