Data Analytics and Mining Project

Elvira Gasanova
N1501156H

Nguyen Ngoc Tram Anh
U1221056H

Márcio Porto
N1500417D

Liu Shaobo
N1500783H

Lasse Vuursteen
N1500648K

Patricio Beltran
N1500934H

Nanyang Technological University

November 8, 2015

Abstract

Producing and sharing online content has become easier than ever thanks to the countless social platforms out there. But not all content shared becomes equally popular. Research shows that a very small number of items receive the most attention, while most content receives only a few views. So what are the elements that make an item more popular than others? That is the answer we set out to find in this project. Guided by previous research, our team analyzed a dataset obtained from UCI Machine Learning Repository, which consists of articles gathered over a two year period from mashable.com, in order to find elements that can lead to content popularity. We found that the factors with the greatest positive impact are subjectivity, data channel (topic) and whether an article is released on the weekend. From different classification methods used, Decision Trees and Naïve Bayes seem to be the best performing models. These results could potentially help content producers, such as online news agencies, reach larger audiences by producing content that is more likely to be read and shared by social media users.

1 Introduction

In today's world where social media has become more prevalent and connected, the popularity of an organization, a celebrity or an article is mostly determined by the number of shares and retweets. Many online users lead a decent life solely through their online activities. The success of a brand's marketing strategy is judged through its number of likes or Youtube views. Similarly, the pervasiveness of an online article is determined through its shares and likes from social media. If we can pinpoint certain factors that make an article viral, many online news agencies will appreciate the findings and formulate their articles using such factors.

2 Related Work

So we arrive at the question: "Which features of an online article determine the amount of shares it will receive?"

In “The Pulse of News in Social Media: Forecasting Popularity”, Roja Bandari, Sitaram Asur and Bernardo Huberman claim to be able to predict popularity of a news article on Twitter with 84% accuracy based on four criteria.[1] The first concerns news sources. An article from The New York Times is more likely to achieve higher popularity than an article from The Huffington Post. The category of the article is also important. In most cases, an entertainment article is more likely than a business one to achieve popularity. The third criteria suggests that the more subjective, emotional or personal an article is, the more it will resonate with readers, increasing its likelihood to become popular. The last criteria concerns name dropping. An article that includes Lady Gaga’s name is likely to become popular because of her fame.

Alexandru Tatar, Panayotis Antoniadis, Marcelo Dias de Amorim and Serge Fdida’s article titled “Ranking news articles based on popularity prediction” makes the obvious prediction that the popularity of an article can be predicted by simply looking at the number of comments it receives on social network websites such as Facebook.[2] This can also be extended to Twitter based on the number of “retweets”. An interesting finding from this article was that the authors used various modelling techniques to model their data, but found that a simple linear regression returned the best results.

From the same authors of the previous article, “A survey on predicting the popularity of web content” summarizes previous research on the topic to make four predictions.[3] The first is that popularity can be predicted by word choice and key phrases. Choosing the correct words to use in the body or title of an article can make it show up in popular recommendation lists, thus increasing its visibility. The second prediction regards sentiment analysis and suggests that articles invoking strong emotions tend to experience higher popularity (similar to the findings of Bandari et al. discussed above). The third prediction is the same one made for their “Ranking news articles based on popularity prediction” article, which was discussed above. The last prediction suggests that popularity can be predicted due to a sudden spike in the frequency of edits on a certain page on Wikipedia. For example, imagine that an earthquake hit Chile. It can be expected that soon after the event a new page will be created on Wikipedia and that the frequency of edits it will receive will be higher than average. That way we can predict that articles about the earthquake will enjoy some level of popularity in social media platforms.

Using two very popular content-sharing platforms - Digg and YouTube - the authors of “Predicting the Popularity of Online Content Predicting the Popularity of Online Content” found that by looking at how many votes, comments or “diggs” content shared online receives during a certain period of time after it is posted, they were able to predict the future popularity of that content.[4] The authors found a strong linear correlation between the popularity of content shortly after it was posted and later as more time passed. Furthermore, they observed that once that “content is exposed to a wide audience, the social network provided by the service does not affect which users will tend to look at the content.” Therefore, in a large scale, social network is not effective in predicting popularity. However, they remain important in smaller scales with smaller number of users. The authors point out in the end that they would like to see how their findings differ depending on which sections of each platform they look at (for example, YouTube’s “games” and “entertainment” sections).

Still trying to predict popularity based on early reactions on Digg, the authors of “Using a Model of Social Dynamics to Predict Popularity of News” use observations of the number of votes received by a recently posted story to estimate how interesting it is.[5] Then, they create and use a model of social dynamic and interaction on Digg to predict how popular the story will become in the next few days (measured by how many votes it receives). The results of this article also show that a more refined model could be potentially created to differentiate how interesting a story is to its fans and to the general users.

The number of comments is usually a good indication of how popular an article is, and in “Predicting the Volume of Comments on Online News Stories”, the authors try to predict how many comments an article will receive before it is even published.[6] They develop a set of features (surface, cumulative, textual, semantic and real-world) and classify articles based on whether they will generate comments and if they will receive few or many comments. The authors employed a 2-step classification where the first step was in charge of determining whether an article had any potential of receiving comments, and the second step was about rating the potential as either high or low. The findings suggest that textual and semantic features prove to be strong predictors of number of comments.

It is certain that the findings from the works discussed above have provided an enormous value to publishing agencies as everyday it is harder to remain relevant in a world where users have increasingly more choices from where to get their news. Likewise, these findings have guided our team during this project.

3 Data Acquisition and Analysis

3.1 Data Description

The dataset for the project assignment was obtained from UCI Machine Learning Repository, “Online News Popularity Data Set” [1]. Our team decided to opt for this database because it is both interesting and has the right level of complexity for a short-term project. This dataset is comprised of instances of articles from `mashable.com`. The articles were

[1]http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity

gathered over a period of two years from January 2013 till January 2015. Please refer to Table 1 for statistical measures of the dataset.

Table 1: Statistical measures of the Mashable dataset.

		Articles per day			
Number of articles	**Total days**	**Average**	**Standard Deviation**	**Min**	**Max**
39000	709	55	22.65	12	105

For ease of reference, each attribute described in this section indicates a corresponding number from the list attributes provided in Appendix. Each sample contains 61 attributes (2 non-predictive, 58 predictive, and 1 class label). The non-predictive variables are URL of the article (0) and number of days between the article publication and the dataset acquisition (1). Predictive attributes in the dataset include raw and derived values. Raw original attributes are retrieved directly from the website, while derived attributes are calculated from the original data in order to further capture the discriminating information about the article.

This paragraph summarizes the original attributes. Number of words in the title (2) ranges from 2 to 23, and number of words in the content (3) ranges from 0 to 8474. Total number of links (<href> tags in html) in the article (7) ranges from 0 to 304, and number of links referencing specifically Mashable articles (8) ranges from 0 to 116. Number of images (9) ranges from 0 to 128, and number of embedded videos (10) ranges from 0 to 91. Article metadata contains information on keywords, data channel type, total number of shares. There are maximum 10 predetermined keywords (12) tagged to each article. Some articles are tagged to a data channel type (13-18). Not all articles are tagged, and each article is only tagged to one data channel maximum. There are six types of data channels: Lifestyle (2099 articles), Entertainment (7056 articles), Business (6528 articles), Social Media (2323 articles), Tech (7346 articles), World (8427 articles). Attributes also indicate on which day of the week an article was published (31-37), specifically: Monday – 6661, Tuesday – 7389, Wednesday – 7435, Thursday – 7267, Friday – 5701, Saturday – 2453, Sunday – 2737. There is a separate attribute indicating whether an article was published on a weekend (38). Among the articles in the given dataset the number of shares (60) varies from 1 to 844300.

This paragraph explains the attributes that were derived from the original ones. Rate of unique words in the content (4): number of unique words / number of words in content. Stop words are common, non-discriminating words that do not have much information content. Some examples of stop words are: "a", "and", "from", "the", "while" etc. Stop words are not useful for prediction purposes. Therefore, the rest of the words in the content are considered non-stop words. Rate of non-stop words (5) in the content: number of non-stop words / number of words in content. Rate of unique non-stop words in the content (6): number of unique non-stop words / number of words in content. Since the number of non-stop words (5) basically includes all words in the article, the rate of unique non-stop words heavily depends on the number of words in the content (3). Average length of the words in the content (11) is the average number of characters in a word. This is a continuous attribute ranging from 0 to 8.042 characters. The values are normally distributed. As mentioned earlier, for each article there is a set of predefined keywords (12). All article keywords have been already sorted based on their average shares. From this global sorted list of keywords the worst, best and average keywords were identified. Then for the worst keyword the number of shares was analyzed to get the minimum (19), maximum (20), and average (21) shares of the articles that used the given keyword. Similarly, the minimum, maximum, and average number of shares was determined for the best (22-24) and average (25-27) keywords. Some articles have links to other articles that have been previously published on Mashable. So for each article, among its referenced articles, the one with minimum (28), maximum (29) and average (30) number of shares is identified. Five most relevant topics were determined by applying the Latent Dirichlet Allocation (LDA) algorithm to all previously published Mashable articles. Closeness of an article to each of these topics (39-43) was measured in range from 0 to 1. Pattern web mining module [2] was used for calculation of subjectivity and polarity sentiment analysis. Text subjectivity (44) reflects the extent to which the text of an article contains subjective information as opposed to objective, factual information. Text sentiment polarity (45) indicates the degree to which the content of an article has a negative (with values in [-1, 0) range), neutral (values equal 0), or positive (with values in (0, 1] range) sentiment. Distinct sets of positive, neutral and negative words were determined. Rate of positive words in the content (46): positive words / total number of words in the article. Rate of negative words in the content (47): negative words / total number of words in the article. Rate of positive words among non-neutral tokens (48): positive words / (positive + negative words). Rate of negative words among non-neutral tokens (49): negative words / (positive + negative words). Polarity score is calculated for each word. Then for each positive word average (50), minimum (51) and maximum (52) polarity is determined. Similarly, for each negative word average (53), minimum (54) and maximum (55) polarity is derived. Title subjectivity (56) reflects the extent to which the words of the title contain subjective information as opposed to objective, factual information. Title polarity (57) score indicates the level to which the title of an article has a positive or a negative sentiment to it. Absolute title subjectivity level (58) shows the distance of title subjectivity score (56) to 0.5. Absolute title polarity level (59) indicates distance of title polarity score to 0.

3.2 Data Pre-Processing

We use Weka to first have an overall statistics for each attribute. Through this, a number of samples are identified as noises. For example, one data point is removed because its rate of unique tokens (*n_unique_tokens*) is 240, while the remaining of dataset has value 0 to 1 for this attribute. Another big set of data is found to have 0 for number of tokens in the content (*n_tokens_content* = 0), which is obviously noise, because the article link was checked out, and there are words in the content. Therefore, we removed all 1181 articles with this anomaly data. Note that a few other methods of filling in missing data were considered, but if *n_tokens_content* is to be filled in, other attributes that were dependent on *n_tokens_content* would need to be re-calculated as well. Re-calculating poses a problem here because we don't have the knowledge of the formulas (for instance, *n_non_stop_words*, *average_token_length*, and *kw_min_min*) to recalculate. Aside

[2]http://www.clips.ua.ac.be/pattern

from the 1181 articles where the value of non-stop words is 0, all $n\_non\_stop\_words$ values are close to 1. The minimum value is 0.999999917 and maximum is 1. We decide to remove this attribute as it does not carry any discriminating meaning between the samples.

We then analyze the distribution of each attribute to get a feel of how they stack against each other. Most of the distributions follow either Gaussian distribution or Poisson distribution. However, there are some graphs that do not follow any known distribution.

For methods like Naive Bayes and Decision tree, it is best to discretize scattered values to reduce variance. These attributes are $n\_tokens\_content$, num_self_hrefs, num_imgs, num_videos, avg_token_length, kw_*_*, $self\_reference\_*\_*$, LDA_*, $title\_subjectivity$, and $title\_sentiment\_polarity$, where $*$ represents min, max or avg.

4 Approach

We attempt to create as accurate of a prediction of the number of shares as possible using a variety of methods and models. This is done both in a regression setting and a classification setting.

For some models, the same model and technique can be used for both classification and regression, such as linear regression. Other predictive models, such as Decision Tree Learning, are only applied to classification. Ultimately, the goal is to compare different models and their prediction scores.

4.1 Models Treated

In this section, we briefly touch on each of the models treated in the report.

4.1.1 Linear Models

Linear models can be used for both regression and classification. Linear models have the benefit of not suffering from the curse of dimensionality as much as k-NN does, for instance.

Also, linear models provide for an economic and easy interpretation of the regression results. It allows for answers to the question: How much a change in one of the variables expected to affect the variable of interest? And, perhaps even more importantly, it allows for statistical inference: Is the obtained result statistically significant?

Issues of linear models are the heavy reliance on an underlying set of assumptions. Linear models lose much of their statistical and predictive power when these assumptions are violated. For OLS for example, it is assumed that

- There is a linear relationship between the regressors and dependent variable.

- The error term and the regressors are independently distributed (exogeneity).
- The regressors themselves are independently distributed (non-multicolinearity).
- The variance of the error term is the constant and independent of the regressors (homoskedasticity).

The most common technique, ordinary least squares, relies heavily on these assumptions. There exist more robust variations of OLS that perform even when some of these assumptions are violated. We use the Ridge and Lasso variations of OLS to obtain models which are more reliable when these assumptions are violated. Notably, the Lasso regression is used to identify the most important predictors.

In both Lasso and Ridge variations of OLS, the expression

$$\min_{\beta} ||y - X\beta|| + \alpha|\beta|^2_{L_{norm}}$$

where $L_{norm} = L_1$ for Lasso and $L_{norm} = L_2$ for Ridge. Because of its absolute value norm, Lasso is more inclined to force the β coefficients to 0 than Ridge. While this generally causes Lasso to underperform against Ridge in non $k > n$ settings, it can prove to be a useful tool to obtain leaner models or identify the "most important" regressors.

To overcome the handicap of only being able to map linear relations, additional nonlinear variations of regressors are added. This is done in accordance to Occam's razor principle. If the addition of a regressor did not improve the adjusted R^2 of the model, the regressor is removed. Since dimensionality is not a problem for linear models, adding regressors should not phase a threat for the asymptotic convergence of our estimators.

4.1.2 K-Nearest Neighbors

In k-Nearest Neighbors (k-NN), an object is classified by looking at the classification of its k closest training examples, using some distance metric.

k-Nearest Neighbors is a so called non-parametric model, meaning it does not require assumptions about the underlying distribution of the data. A k-NN model doesn't need to learn from the data set, its only parameter is k and optimal dimension for data (advanced). A disadvantage of k-NN is that it suffers greatly from the curse of dimensionality, as it only relies on local information.

4.1.3 Support Vector Machines

Support Vector Machines are a class of models which can be used for classification. The support vector classifier used in this report is the so called Support Vector Classifier (SVC). It relies on the minimization problem

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - \iota^T\alpha$$
$$\text{subject to } y^T\alpha = 0 \text{ and } 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n$$

where ι is a vector of all ones, $C \in \mathbb{R} > 0$ a constant, $Q \in \mathbb{R}^{n \times n} \geq 0$ with the i, jth element equal to the kernel $K(x_i, x_j)$, $x_i \in \mathbb{R}^k$ being one of the n training vectors (observations).

The kernel is a function that maps two vectors to a real number. Different types of kernels create different decision boundaries. We compare the SVC for four different kernels:

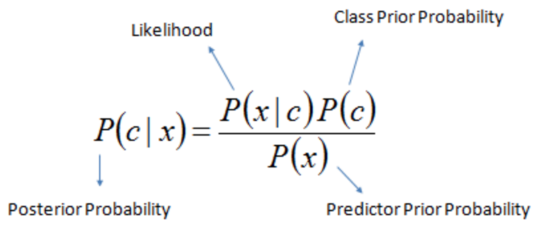
- A linear kernel: $K(x, y) : \mathbb{R}^{k \times 2} \to \mathbb{R} := x^T y$
- A polynomial kernel of degree 2: $K(x, y) : \mathbb{R}^{k \times 2} \to \mathbb{R} := (\frac{1}{k} x^T y)^2$
- A polynomial kernel of degree 3: $K(x, y) : \mathbb{R}^{k \times 2} \to \mathbb{R} := (\frac{1}{k} x^T y)^3$
- Radial basis function (rbf) kernel: $K(x, y) : \mathbb{R}^{k \times 2} \to \mathbb{R} := \exp^{(\frac{1}{k}|x-y|^T|x-y|)}$

where k is the number of features used in the model (equivalent to the dimension of x, y). Using these different kernels, we avoid relying just on the assumption of the data being linearly separable.

The C parameter plays an important role in the SVC as specified above. For a large C, the model has a lot of room to fit the values of α_i to the training data. For a small C, the error on the training data will be larger, but the model is less sensitive to noise. This trade-off will be further analyzed by comparing cross validation scores for different values of C for each of the different kernels.

4.1.4 Naive Bayes

Naive Bayes classifier is based on Bayes theorem to find the probability of the data point being assigned a class based on its attributes. Below is Bayes Theorem that is the core of Naive Bayes algorithm:


Likelihood
Class Prior Probability
$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

Having datapoint x, we calculate the probability of having class c_i provided x. For a problem with i classes, we calculate $P(c_i|x)$ for each class, and assign class c_k where $P(c_k|x)$ is the largest. Since $P(c_i|x)$ has the same denominator that is $P(x)$, to minimize computation, we can omit calculating the denominator. That is,

$$P(c|x) = P(x_1|c)P(x_2|c)...P(x_n|c)P(c)$$

There is one assumption for Bayes theorem to work, that is

$$P(x|c) = P(x_1, x_2, ..., x_n|c) = P(x_1|c)P(x_2|c)...P(x_n|c)$$

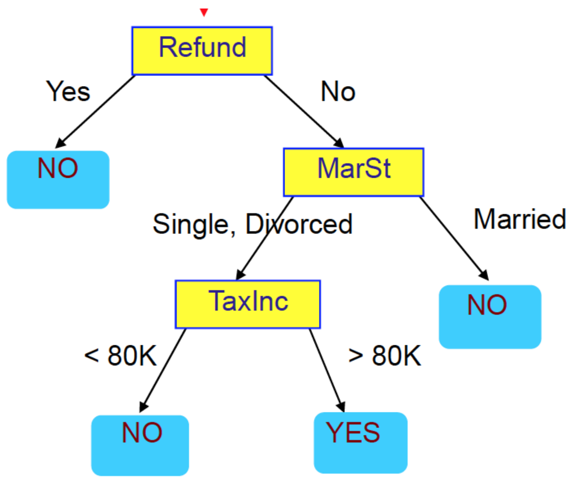
This means all attributes for a sample are assumed to be independent of each other. This is why Naive Bayes algorithm is called Naive, because it is assumed all attributes are independent, while in practice, there might be some slight correlation between attributes.

Another problem with Naive Bayes classifier is that it works best with discretized data. If the data is continuous, every value of the attribute is unique, making $P(xi_|c)$ very small and insignificant. Therefore, in our implementation, for Naive Bayes, we discretize all attributes before running the classifier.

4.1.5 Decision Tree

Decision tree builds classification model in the form of a tree structure. With a tree learned from training set, any unseen data can be walked through a tree to reach a node, where it will indicate which class the data belongs to. Below is an example of a decision tree.

Figure 1: Decision Tree Diagram



The difficult part in the Decision tree algorithm is how we build an effective tree, ie. short but capture the most information. This step is called Tree Induction step. This involves choosing which attribute to split first, and at which value to split the branch. To decide on which attribute, a general algorithm is to measure Info Gain [3] or GINI Index [4] for each attribute, at each split. Then the attribute and the split are chosen based on where the Info Gain is the maximum. We repeat the process of tree induction until we arrive with a node with pure class. This is the stopping criteria for the tree induction step.

A decision tree can be built based on the induction step as described above. However, over-fitting problem will occur if we let the tree run until there is a pure node. This is

[3]http://www.sciencedirect.com/science/article/pii/S1042814305800202

[4]http://people.revoledu.com/kardi/tutorial/DecisionTree/how-to-measure-impurity.htm

because the tree only performs well for the training data, and the state of the tree is heavily dependent on the distribution of the training data. Therefore, to reduce over-fitting problem, one can either use early stopping, ie. stop building the tree if a certain impurity is tolerated, or use a post-processing step called Tree Pruning.

Early stopping is easy to perform, if the Info Gain or Gini index reaches a certain threshold, further splitting will be stopped. However, in some cases, this method will reduce the classification rate.

Tree Pruning is a more advanced step to reduce the tree after we've fully built in. Having a complete tree, we will look at the leave nodes to "prune" it, meaning to cut short the branch where Info Gain or Accuracy rate is not compromised. In Weka, the J48 Decision Tree Algorithm uses pruning by default.

4.2 Methodology

4.2.1 Visual Exploration

We investigate the data using a multitude of graphical display options. This is done for both description purposes as to get a general idea for relations between variables.

A graph for distribution of each attribute is in the Appendix. Note that only attributes with interesting distributions are displayed.

4.2.2 Identifying Useful Predictors

As previously stated, k-NN suffers from the curse of dimensionality. In order to reduce the dimensionality, we attempt to create a leaner model with only the most significant predictors using a Lasso regression with varying amounts of punishment. The non-zero coefficients resulting from this regression are then used in the k-NN model.

The other nonlinear classifiers are less "locally focused" and therefore are unlikely to benefit from this approach.

For the OLS procedure, we investigate in the same manner if (nonlinear) variations of regressors, such as interaction variables and square variables, yield better prediction results.

4.2.3 Comparing Models

For each of the models discussed in the previous section, we will compare cross validation scores for different model parameters.

Cross validation is done by repeatedly splitting the data set in training and test set (75% and 25% of the data respectively). Classifiers are built based on training set, test errors are calculated based on classifier error of test set. Note that we don't use validation set here because we have more than enough samples (36000 samples for 56 attributes), so the test error is used to compare between models, as well as represent the actual test error.

When deciding on the amount of cross validation folds, two things are taken into consideration. The standard deviation of the cross validation scores and the time it takes to complete a particular cross validation fold.

A higher standard deviation among the cross validation scores would imply more folds are required in order to obtain a sense of significance for the score. This is important as a model might seem like it is performing well comparatively, while it is just due to luck of the draw.

On the other hand, we are also limited by computational power. Some models take a particularly long time to run (for example, SVM with non linear kernels). Because of this, it is not always possible to run a lot of cross validations. We have therefore tried to strike a good balance between computational speed and significance of the cross validation scores for each model.

5 Implementations

We approached the data analysis problem with Python initially, as it is a rising language for Data Mining, praised for its versatility and strong support for libraries. We did the entire data analysis and mining with Python at first. With ample amount of time and human resources left, we went on to explore Weka, the recommended tool from the lecturer. We noticed slightly different performance results between Python and Weka. We will discuss these differences in a later part of our report.

5.1 Python Implementation

In this section, we will briefly describe the tools used and steps taken in Python to obtain the results discussed in this report. It can be read as either documentation (snippets of code have been included), or to get a general idea of our work. We use the procedures to obtain the SVM estimates as an example, which can be extrapolated to any other model as the procedure is very similar.

For Python, there are five libraries we used extensively for this report. First, there is the `Numpy` library for mathematical operations and matrix handling. For the handling of large data frames and importing data, we used the `Pandas` library. The implementations of our classifiers and preprocessing we used are obtained from the `SciKit-Learn` library. For the linear models, the `Statsmodels` library is used as it offers more extensive regression support. For plotting and visualizations, we used the `MatPlotLib` library.

As Weka was our main tool for visualization and data cleaning, this is not implemented in Python. We simply make use of the obtained csv file. Importing the *.csv* data into Python data frames is done as follows:

Listing 1:

```
# Import csv data
raw_data = pd.read_csv('OnlineNewsPopularity_wLabels_deleteNoise.
    csv').iloc[:, 1:]       # read in csv, omit the first column of
      url
print np.median(raw_data['␣shares'])
# Remove the binarized column of shares
raw_data = raw_data.iloc[:, :-1]
# Take up to the second last column
news_data = raw_data.iloc[:, :-1]
# Take shares column for labels
news_labels = raw_data.iloc[:, -1]
```

We have now obtained our variable of interest (`news_labels`) and our variables used for prediction (`news_data`). In case of SVM, we have to binarize the label and preprocess the predictory variables, which is done as follows:

Listing 2:

```
# the median amount of shares is the boundary of our independent
    variable
print '\nBinary␣Threshold:'
binary_threshold = np.median(raw_data['␣shares'])

# Binarize the labels
binarizer = Binarizer(threshold=binary_threshold)
y_binary = binarizer.transform(news_labels).transpose().ravel()

# Preprocess X to standard normal
scalerX = preprocessing.StandardScaler().fit(news_data)
X = scalerX.transform(news_data)
```

We then introduce helper functions to aid us in cross validation for different parameters. In case of SVM, it is just the C parameter. Note that there are four different instances of SVC objects used, each with a different choice of kernel. The function returns the mean cross validation score and the mean standard deviation of the folds.

Listing 3:

```
# cv = amount of cross validation folds
def cv_mean_std_SVM(C, cv):
    lin = svm.SVC(C=C, kernel='linear', max_iter=10000)
    poly_2 = svm.SVC(C=C, kernel='poly', degree=2, max_iter
        =10000)
    poly_3 = svm.SVC(C=C, kernel='poly', degree=3, max_iter
        =10000)
    rbf = svm.SVC(C=C, kernel='rbf', max_iter=10000)
    cv_lin = cross_val_score(lin, X, y_binary, cv=cv)#, dual=
        False)
    print 'finished␣computing␣linear␣cv'
    cv_poly_2 = cross_val_score(poly_2, X, y_binary, cv=cv)#,
       dual=False)
    print 'finished␣computing␣polydeg2␣cv'
    cv_poly_3 = cross_val_score(poly_3, X, y_binary, cv=cv)#,
       dual=False)
    print 'finished␣computing␣polydeg3␣cv'
    cv_rbf = cross_val_score(rbf, X, y_binary, cv=cv)#, dual=
       False)
    print 'finished␣computing␣rbf␣cv'
    return np.mean(cv_lin), np.std(cv_lin), np.mean(cv_poly_2),
       np.std(cv_poly_2), np.mean(cv_poly_3), np.std(cv_poly_3),
       np.mean(cv_rbf), np.std(cv_rbf)
```

This is the basic setup for experimenting with different models and parameters. As stated previously, this setup is the same for each of the different models treated in Python. The above function can be used in for loops to loop over different parameter values to optimize the model. For the Python implementation of other models, please refer to the code and given documentation.

5.2 Weka Implementation

Having had a rough idea of how different classifiers fare with our data set, we continued to explore Weka.

An advantage from Weka is its user-friendliness with strong powerful data visualization tools. In fact, through Weka, we were able to visualize the distribution for each attribute, therefore identifying noises in our dataset. Of course, we went back to our Python classifiers to run the classifiers again after removing these noises.

6 Experimental Results and Analysis

In this section, we will present and discuss the results in terms of accuracy score.

6.1 Regression Models

6.1.1 Regression Results

Regressing with the large number of additional regressors results in a model which violates multiple assumptions as stipulated in the Approach section.

When testing for collinearity, we find that a lot of the NLP keyword-based regressors suffer from correlation larger than 0.80, implying there might be multicolinearity at play.

To test for heteroskedasticity (violation of homoskedasticity) we regress our original regressors against the squared residual of the original regression:

$$\hat{\epsilon}^2 = \mathbf{X}\alpha$$

where $\hat{\epsilon}$ is the estimated error obtained from the initial regression. Through this method, we find a joint coefficient significance F-score of 1.285. This translates to rejecting the hypothesis that $\alpha_i = 0$ for all $i \in 1, \ldots, k$, which translates into a P-value of 0.0211, which implies that the regressors can be used to predict the error term variance, therefore implying heteroskedasticity persists.

In an attempt to obtain a leaner, more robust model, we apply a selection process in which we force less significant and colinear regressor coefficients to 0 using an initial Lasso regression and comparing values for the mean square error (MSE) and mean absolute error (MAE).

In figure 6 in the Appendix section, we show how the choice of α or punishment factor affects the amount of regressors with nonzero coefficients.

The resulting model of these tests is significantly leaner, with just 17 regressors including a constant. It has an improved joint significance of its coefficient estimates, even using heteroskedasticity robust standard errors. The fit of the model is still quite poor, with an R^2 of only 0.0136. The regression results can be found in table 4.

Firstly, the global subjectivity of the article seems to be a major factor in predicting the amount of shares that article receives, where a more subjective article gets significantly more shares than an objective article.

Another interesting result of the regression is an expected increase of shares by 394 for articles published on the weekends, keeping other factors fixed. This could be attributed to people having more time on the weekends to read and subsequently share articles and/or the fact that there are less articles published during the weekends (only 13.2% of all articles are released on weekends), resulting in less 'competition' between articles.

Additionally, the amount of days since the article has been released influences the amount of shares significantly, both through a linear and a quadratic component. The linear component has a positive coefficient with a value of 6.939, whereas the quadratic component has a coefficient of -0.010. This implies that initially, expected amount of shares increases per day since the article has been released, but this effect slows down over time. This result is to be expected.

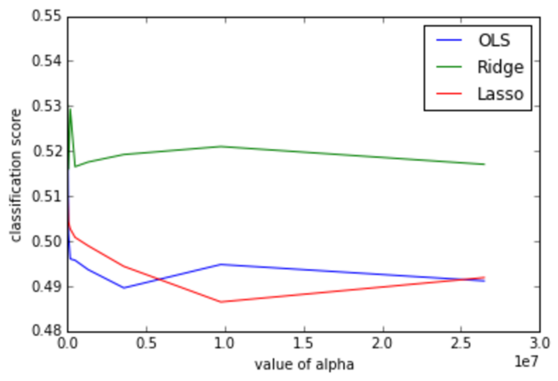
The data channel also seems to negatively impact the amount of shares for all of the obtained channels. This seems to imply that more general topics (topics that are in data channel "other") experience higher popularity.

6.1.2 Classification Results

The fit of linear models, even with the addition of nonlinear variations and interaction variables is fairly poor. The maximum cross validation score is just 0.553 for the ridge model.

The leanness of the model does not seem to be the problem. Increasing values of α does not result in improved cross validation performance as the model sheds the less important regressors. This can be seen in Figure 2 below.

Figure 2: Cross validation scores of linear models for different α values. The amount of regressors resulting from a particular choice of alpha can be found in Figure 6



.

We conclude that the relationships between the regressors in the data and the number of shares is very likely to be nonlinear (assuming these relationships are strong).

6.2 Classifier Models

In this section, we discuss parameter optimization for different models and how they stack up against each other.

6.2.1 Experimenting with parameters of k-Nearest Neighbors

As the k-NN algorithm is the most sensitive to dimensionality of the data, we have experimented with reducing the dimensionality through running a Lasso regression for an increasingly large punishment factor α. This way, we can reduce dimensionality by removing regressors based on significance (removing the least significant ones first).

Since for a reduction in data there might be a change in the optimal value for the k parameter, iterations over different values of both are required. The resulting cross validation values are plotted below in Figure 3.

The first conclusion that can be drawn is that the dimensionality exerts little influence on the predictory ability of a local method such as k-NN for our dataset. Luckily, our sample is large enough to overcome the curse of dimensionality.

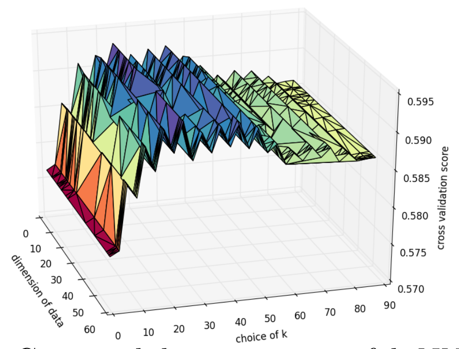


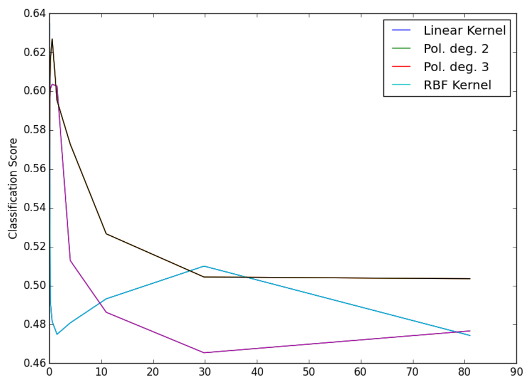
Figure 3: Cross validation scores of k-NN for different values of k and number of regressors.

The smallest dimensionality tried (10 re-
gressors) is less than 1% removed from the classification score obtained by using all regressors. The "bad" implication is that there are a lot of regressors with little or no predictory value when using a local method.

6.2.2 Experimenting with parameters for Support Vector Machines

For support vector machines, different punishment factors C give quite drastically different classification results as can be seen in Figure 4. The optimal value of C seems to be less than 0.1 for each of the different kernels. A larger value results in a rapid drop of classification score.

Figure 4: Cross validation scores of SVM



The best choice of kernel is the RBF kernel with a classification score of 62.71. The linear kernel scores the worst for any value of C, which supports the poor result of the linear models tried earlier.

6.2.3 Experimenting bins for Discretization for DT and NB

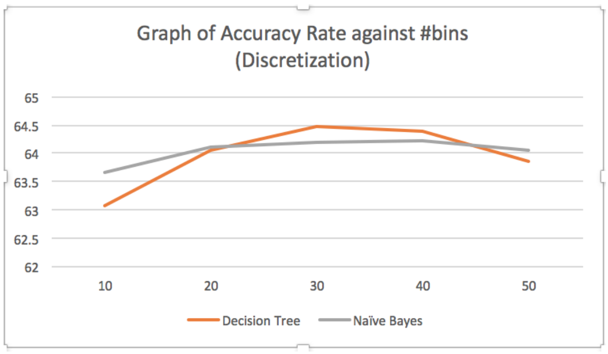
For classifiers like Decision Tree (DT) and Naive Bayes (NB), as mentioned before, it is best to discretize the data. We experimented with a number of bins between 10 and 50, in order to determine the best bins for each of these two classifiers.

Figure 5 is the graph to show the accuracy rate vs the number of bins.

We can see from the graph, when number of bins is more than 30-40, accuracy score starts to decline. Hence, we stop the experimentation at 50 bins.

Therefore, we discretize into 30 bins for Decision Tree and 40 bins for Naive Bayes.

Figure 5: DT and NB Bins Discretization



6.3 Comparison of all models

After determining the number of bins to discretize for Decision Tree and Naive Bayes, optimal k for k-NN, and optimal C for SVM, we train the classifiers with our data on both Python and Weka. Table 2 shows the results (Accuracy score in %).

Table 2: Comparison table for different models

	Python	**Weka**
Linear Model	55.32	54.25
SVM	62.71	52.73
Naïve Bayes	60.73	64.23
Decision Tree	55.94	**64.46**
kNN	58.12	57.52

Note that Naive Bayes and Decision Tree in Python performs much worse than that in Weka because we did not discretize the attributes. This shows that discretization really improves Naive Bayes and Decision Tree, where priori probability and range values are important.

Also, there is a discrepancy between Python's SVM and Weka's SVM. This is interesting because we input the same parameters as we had set in Python, but the accuracy score is only around 52%, which is 10% less than what we had got in Python. One explanation is because we did not normalize to standard normal before feeding into SVM classifier in Weka. Another possible explanation lies in the optimization in Weka that may not work well with our dataset.

Besides the 3 methods described above (Naive Bayes, Decision Tree and SVM), the remaining methods return similar results between Python and Weka, with Python performing slightly better than Weka.

7 Conclusion

The econometric results obtained from our regressions yield the conclusion that the subjectivity, data channel and whether the articles are posted during the weekend seem to be the main single determinants of amount of shares, keeping all other factors fixed.

Therefore, an author that wants to maximize his shares increases his likelihood of doing so by writing a subjective article. The data channel should be either general or other, and the article should preferably be released in the weekend.

Even though they prove useful in giving economic interpretation to some of the regressors, the classification results for the linear models are the poorest. This goes for linear regression models as well as SVM with a linear kernel.

Whether SVM is effective seems to be fully dependent on choice of kernel, where an RBF kernel seems to provide the best results compared to polynomial kernels of degrees between 1 and 3.

For classification, the best classification method according to our cross validation scores is the Decision Tree method. This is in accordance with the original paper analysing the same dataset, where the authors found the Random Forest to be the best classifier, as Random Forest is a variation based of the Decision Tree algorithm. As a close second to the Decision Tree algorithm, there is Naïve Bayes.

When it comes to the environment used for data analysis, we conclude that Weka provides a lot more and easier accessible tools right out of the box. Python on the other hand provides easy customization and is faster than Weka in run time.

References

[1] Roja Bandari, Sitaram Asur, Bernardo Huberman *The Pulse of News in Social Media: Forecasting Popularity*
`http://www.hpl.hp.com/research/scl/papers/newsprediction/pulse.pdf`

[2] Alexandru Tatar, Panayotis Antoniadis, Marcelo Dias de Amorim, and Serge Fdida *Ranking news articles based on popularity prediction.*
`http://www.tik.ee.ethz.ch/~pantonia/papers/asonam_camera_ready.pdf`

[3] Alexandru Tatar, Panayotis Antoniadis, Marcelo Dias de Amorim, and Serge Fdida *A survey on predicting the popularity of web content*
`http://www.jisajournal.com/content/5/1/8`

[4] Gabor Szabo and Bernardo A. Huberman *Predicting the Popularity of Online Content*
`http://www.hpl.hp.com/research/idl/papers/predictions/predictions.pdf`

[5] Kristina Lerman, Tad Hogg *Using a Model of Social Dynamics to Predict Popularity of News* `http://www.isi.edu/~lerman/papers/wfp0788-lerman.pdf`

[6] Manos Tsagkias, Wouter Weerkamp, Maarten de Rijke *Predicting the Volume of Comments on Online News Stories*

https://staff.fnwi.uva.nl/m.derijke/wp-content/papercite-data/pdf/tsagkias-predicting-2009.pdf

A Appendix

Figure 6: Number of regressors with non-zero coefficients, running the Lasso algorithm for different α values.

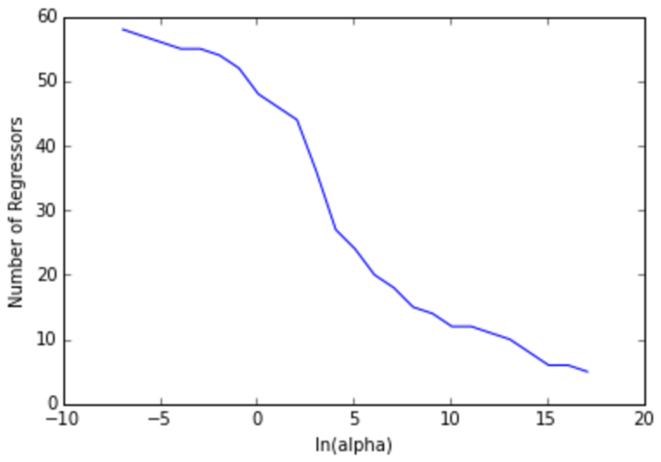


Figure 7: Distributions of Attributes

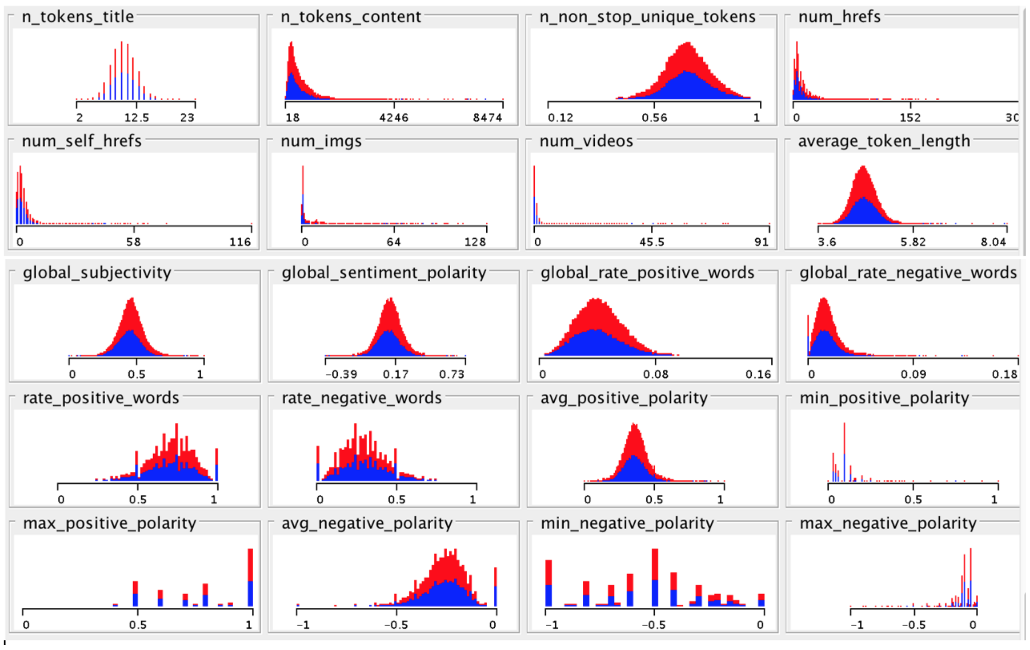


Table 3: List of attributes: 58 predictive, 2 non-predictive, 1 goal field

#	Name	Description	Data Type
0	url	URL of the article	nominal
1	timedelta	Days between the article publication and the dataset acquisition	discrete integer
2	n_tokens_title	Number of words in the title	discrete integer
3	n_tokens_content	Number of words in the content	discrete integer
4	n_unique_tokens	Rate of unique words in the content	ratio [0,1]
5	n_non_stop_words	Rate of non-stop words in the content	ratio [0,1]
6	n_non_stop_unique_tokens	Rate of unique non-stop words in the content	ratio [0,1]
7	num_hrefs	Number of links	discrete integer
8	num_self_hrefs	Number of links to other articles published by Mashable	discrete integer
9	num_imgs	Number of images	discrete integer
10	num_videos	Number of videos	discrete integer
11	average_token_length	Average length of the words in the content	continuous real number
12	num_keywords	Number of keywords in the metadata	discrete integer
13	data_channel_is_lifestyle	Is data channel 'Lifestyle'?	binary
14	data_channel_is_entertainment	Is data channel 'Entertainment'?	binary
15	data_channel_is_bus	Is data channel 'Business'?	binary
16	data_channel_is_socmed	Is data channel 'Social Media'?	binary
17	data_channel_is_tech	Is data channel 'Tech'?	binary
18	data_channel_is_world	Is data channel 'World'?	binary
19	kw_min_min	Worst keyword (min shares)	discrete integer
20	kw_max_min	Worst keyword (max shares)	discrete integer
21	kw_avg_min	Worst keyword (avg shares)	discrete integer
22	kw_min_max	Best keyword (min shares)	discrete integer
23	kw_max_max	Best keyword (max shares)	discrete integer
24	kw_avg_max	Best keyword (avg shares)	discrete integer
25	kw_min_avg	Avg. keyword (min. shares)	discrete integer
26	kw_max_avg	Avg. keyword (max. Shares)	discrete integer
27	kw_avg_avg	Avg. keyword (avg. Shares)	discrete integer
28	self_reference_min_shares	Min. shares of referenced articles in Mashable	discrete integer
29	self_reference_max_shares	Max. shares of referenced articles in Mashable	discrete integer
30	self_reference_avg_sharess	Avg. shares of referenced articles in Mashable	discrete integer
31	weekday_is_monday	Was the article published on a Monday?	binary
32	weekday_is_tuesday	Was the article published on a Tuesday?	binary
33	weekday_is_wednesday	Was the article published on a Wednesday?	binary
34	weekday_is_thursday	Was the article published on a Thursday?	binary
35	weekday_is_friday	Was the article published on a Friday?	binary
36	weekday_is_saturday	Was the article published on a Saturday?	binary
37	weekday_is_sunday	Was the article published on a Sunday?	binary
38	is_weekend	Was the article published on the weekend?	binary
39	LDA_00	Closeness to LDA topic 0	ratio [0,1]
40	LDA_01	Closeness to LDA topic 1	ratio [0,1]
41	LDA_02	Closeness to LDA topic 2	ratio [0,1]
42	LDA_03	Closeness to LDA topic 3	ratio [0,1]
43	LDA_04	Closeness to LDA topic 4	ratio [0,1]
44	global_subjectivity	Text subjectivity	ratio [0,1]
45	global_sentiment_polarity	Text sentiment polarity	ratio [-1,1]
46	global_rate_positive_words	Rate of positive words in the content	ratio [0,1]
47	global_rate_negative_words	Rate of negative words in the content	ratio [0,1]
48	rate_positive_words	Rate of positive words among non-neutral tokens	ratio [0,1]
49	rate_negative_words	Rate of negative words among non-neutral tokens	ratio [0,1]
50	avg_positive_polarity	Avg polarity of positive words	ratio [0,1]
51	min_positive_polarity	Min polarity of positive words	ratio [0,1]
52	max_positive_polarity	Max polarity of positive words	ratio [0,1]
53	avg_negative_polarity	Avg polarity of negative words	ratio [-1,0]
54	min_negative_polarity	Min polarity of negative words	ratio [-1,0]
55	max_negative_polarity	Max polarity of negative words	ratio [-1,0]
56	title_subjectivity	Title subjectivity	ratio [0,1]
57	title_sentiment_polarity	Title polarity	ratio [-1,1]
58	abs_title_subjectivity	Absolute subjectivity level	ratio [0,0.5]
59	abs_title_sentiment_polarity	Absolute polarity level	ratio [0,1]
60	shares	Number of shares (target)	discrete integer

Table 4: Table displaying results for the heteroskedasticity robust regression with highest joint significance.

	(1)	(2)
VARIABLES	shares	shares
timedelta	6.939***	6.939***
	(1.343)	(1.279)
timedelta2	-0.0100***	-0.0100***
	(0.00200)	(0.00208)
num_hrefs	43.98***	43.98***
	(5.851)	(7.166)
num_self_hrefs	-60.63***	-60.63***
	(17.01)	(16.13)
average_token_length	-561.3***	-561.3***
	(93.24)	(102.5)
data_channel_is_lifestyle	-2,133***	-2,133***
	(294.0)	(326.6)
data_channel_is_entertainment	-2,585***	-2,585***
	(205.7)	(267.8)
data_channel_is_bus	-2,316***	-2,316***
	(217.4)	(341.3)
data_channel_is_socmed	-1,968***	-1,968***
	(286.2)	(289.7)
data_channel_is_tech	-2,349***	-2,349***
	(208.6)	(284.3)
data_channel_is_world	-3,004***	-3,004***
	(210.7)	(283.5)
kw_min_min	3.180**	3.180*
	(1.383)	(1.632)
kw_max_min	0.0794***	0.0794*
	(0.0151)	(0.0456)
is_weekend	394.0**	394.0**
	(172.9)	(164.3)
global_subjectivity	3,628***	3,628***
	(663.7)	(728.4)
abs_title_subjectivity	341.7	341.7
	(308.3)	(301.2)
Constant	5,107***	5,107***
	(373.1)	(417.5)
Observations	39,644	39,644
R-squared	0.014	0.014

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1