Predicting News Sharing on Social Media

Joe Johnson & Noam Weinberger

1. Introduction

The ability to predict which online articles will be most popular would shed light on how media companies can attract subscribers, how important information is spread, how advertisements can be targeted to be most effective, and how public opinion is formed. In this project, we use features of online articles to predict the number of times they will be shared on social media. We approach this task in two ways: regression and classification. In regression, we try to predict the exact number of times an article will be shared. In classification, we try to predict whether an article will be popular, where a popular article is one that has been shared more than a chosen threshold.

Previous research has been done to predict online article popularity. Fernandes, Vinagre, and Cortez constructed the dataset we use, which is described below. They approached this as a binary classification task, labeling articles as popular if they were shared more than the median number of shares, or as unpopular otherwise. They found that random forests was the most effective method for this prediction, reaching an accuracy of 67%.

Keneshloo, Wang, Han, and Ramakrishnan approached the task as a regression problem predicting popularity of an article after 24 hours, though using a different dataset. Their results were very accurate, however their features included information about the articles' popularity after 30 minutes. In our work, we use only features known before publication.

2. Data

For this project, we use the Online News Popularity Dataset from the University of California Irvine Machine Learning Repository. The dataset contains information on 39,643 articles published on mashable.com from 2013 to 2014. This information includes the number of times the article was shared, as well as details such as its topic, day of publication, number of images, length, and sentiment analysis. These features do not include information about the actual words found in the article, which limits how specific our prediction can be. However, the dataset also contains URL links to the articles themselves, which allows us to gather the content of the articles.

One concern about the dataset is that the age of articles varies significantly. The number of times an article is shared will tend to be higher the longer the article has been online. In this case, shares would not really reflect the popularity of the article, but would simply reflect its age. To avoid this issue, we make the assumption that the sharing of an article levels off at around 30 days after its publication. In this dataset, there are about 1,000 articles that had been published fewer than 30 days before data collection. We remove these during preprocessing, leaving only share values that capture popularity, rather than age.

3. Regression

We run a linear regression using all features in the dataset to predict the number of shares an article will have. After the preprocessing described above, we separate 2/3 of the articles to train the model and 1/3 to test it. As a baseline, we also compare the test set values to an array of randomly predicted shares drawn from a normal distribution with the same mean and variance as the training set values.

The random predictions result in a test root mean square error (RMSE) of 16728 shares, while the full linear regression predictions result in a test RMSE of 12704 shares. It is difficult to interpret what a good result would be, since the average number of shares is very low (3409) but many articles have shares in the hundreds of thousands. However, it is evident that the linear regression model is a significant improvement over the baseline random predictions.

We take a closer look at the effect of each feature on the model by predicting shares using linear regression with each feature, individually. We find that, while all of the features perform better than random prediction, most of them are relatively insignificant. A handful of the features hold the majority of the predictive power.

3.1. Principal Components

In an attempt to simplify this dataset, we conduct Principal Component Analysis, searching for directions along which the data has high variance. The principal components that we find, however, do not improve on the original features. When we use the top 10 principal components in a linear regression, the RMSE is 12725, which is approximately the same as using the features directly.

3.2. Backward Selection

To extract only the most relevant features for our analysis, we use backward step selection. Of our initial model of 56 regressors, backward selection narrows our model down to 23. However, when using only these 23 features in a linear regression, the RMSE on the test set is 12705, which is not an improvement on the full feature model.

3.3. Lasso and Ridge Regression

As an alternative approach to feature selection, we run ridge and lasso regression. These are two methods that penalize predictors that are near zero and help prevent overfitting. We use 5-fold cross validation on the training set to estimate the regularization parameters. The cost function for lasso/ridge regression is:

$$min_\beta \in \mathbb{R}^{p+1} - [\frac{1}{2N}\sum_{i=1}^{N} y_i - x_i^T\beta)^2] + \lambda[(1-\alpha)||\beta||_2^2/2 + \alpha)||\beta||_2^2/2 + \alpha||\beta||_1]$$

Lasso corresponds to $\alpha = 1$, while ridge corresponds to $\alpha = 0$.

Table 1 compares the performance of each model.

Table 1: Regression Performance

	Training RMSE	Test RMSE
Linear Regression	11031	12704
Principal Components Regression	11093	12725
Forward Stepwise Selection	11033	12705
Ridge Regression	11034	12703
Lasso	11035	12696

Lasso does slightly better than linear regression but the difference is not substantial.

3.4. Feature importance for regression

While none of the regression models attain outstanding performance on the test set, there are some variables that are particularly significant. Significance of a feature is determined by the p-value of its coefficient after running linear regression. Table 2 shows the most significant predictors:

Table 2: Feature Importance

	coefficients
Avg. keyword (avg. shares)	1.44
Avg. keyword (max. shares)	-0.1736
Entertainment Data Channel	-1435
Min. shares of referenced articles	0.0379
Number of links to other articles	-85.49

The two highest predictors are keywords, which describe the number of shares of other articles about similar topics. This suggests that the actual content of an article is more predictive than its more structural features. Another significant feature is the minimum share rate of other referenced Mashable articles. Again, this suggests that content is very important, as articles that are related to earlier popular topics will be popular as well.

4. Classification

An alternative approach to regression is to binarize the data into two categories of either "unpopular" or "popular" news articles. An article is labeled popular if it falls above a certain threshold number of shares, while it is labeled unpopular if it falls below that threshold. To start we set our threshold at the median, which is 1,400 shares. Since the dataset is high-dimensional and complex, we try several different classification algorithms: LDA, QDA, random forests, logistic regression, penalized logistic regression, SVM with linear kernel, and SVM with RBF kernel.

4.1. Logistic Regression and Penalized Logistic Regression

We run logistic regression, as well as penalized logistic regression. The latter model is similar to standard logistic regression, but includes L1 and L2 norm penalties for regularization, which help prevent overfitting. In this case, the coefficients β are estimated by solving:

$$min_{\beta} \in \mathbb{R}^{p+1} - [\frac{1}{N}\sum_{i=1}^{N} y_i(x_i^T\beta) - log(1 + exp(x_i^T\beta))] + \lambda[(1-\alpha)||\beta||_2^2/2 + \alpha)||\beta||_2^2/2 + \alpha||\beta||_1]$$

We use the implementation by Friedman, Hastie, and Tibshirani (2010) in the R package glmnet, which solves the problem with a quadratic approximation to the log-likelihood and coordinate descent. We tune the regularization parameters λ and α using 5-fold cross validation on the training set.

4.2. SVM

We evaluate SVMs with the linear and radial basis function kernels, using a grid search to select the SVM parameters. We further tune the parameters on each value of the grid search using 5-fold cross validation.

4.3. Gaussian Discriminant Analysis

We use two variants of Gaussian Discriminant Analysis: linear discriminant analysis and quadratic discriminant analysis. Both algorithms create a model of the distribution of the labels, but LDA attempts to draw a linear boundary between the classes, while QDA attempts to draw a quadratic boundary.

4.4. Random Forests

The final classification algorithm we implement is random forests. Random forests use multiple decision tree classifiers as weak learners, and then weighs each weak learner to create a strong classification model.

4.5. Comparison

Table 3 compares each model on the test set using various performance measures.

Table 3: Classification Performance

	Accuracy	PPV	NPV	Sensitivity	Specificity
Logistic Reg.	0.65	0.66	0.64	0.63	0.68
Penalized Log. Reg	0.65	0.66	0.64	0.63	0.68
Linear SVM	0.64	0.64	0.63	0.62	0.65
SVM RBF Kernel	0.53	0.6	0.52	0.2	0.87
LDA	0.65	0.66	0.64	0.62	0.68
QDA	0.61	0.68	0.58	0.39	0.82
Random Forest	0.66	0.67	0.66	0.65	0.67

The models all attain accuracy between 53-66%, with SVM with RBF kernel as the least accurate and random forests as the most accurate. Penalized logistic regression does not differ much from standard logistic regression, whil SVM with RBF kernel is much less accurate than SVM with linear kernel. While 66% is significantly better than random, and is similar to the result achieved by Fernandes, et al., it does leave a lot of room for improvement. The other performance metrics are similarly successful.

4.6. Higher Popularity Thresholds

While we have some success at predicting whether an article is popular or unpopular with the threshold at the 50th percentile, a news agency may be more interested in predicting only the most highly shared articles. Given the heavy-tailed distribution of shares, these articles gain far more exposure and attention. To see whether we can forecast these highly popular articles we increase the popularity threshold to both the 90th and 99th percentiles and use random forests, the strongest performing learning algorithm above, to predict popularity.

This task, however, runs into a class imbalance problem, where the algorithm can simply label all articles as unpopular and attain a high degree of accuracy. This model is not learning what makes an article popular, it is just learning that most observations belong in the unpopular class. To get a better estimate, we instead undersample from unpopular articles so that there is a roughly equal number of training and test examples from each class. To do this, we divide the training and test sets as before, and then randomly remove unpopular articles from our data until the number of popular articles equals the number of unpopular articles.

Table 4 compares the results of these three thresholds across various performance measures.

Table 4: Performance on Higher Thresholds

	Accuracy	PPV	NPV	Sensitivity	Specificity
90% Thresh: Standard	0.9	0	0.9	0	1
90% Thresh: Downsample	0.64	0.17	0.95	0.67	0.64
99% Thresh: Standard	0.9912	NA	0.9912	0	1
99% Thresh: Downsample	0.71	0.02	1	0.69	0.71

Random forests with these higher thresholds achieves similar accuracy as above, though does somewhat worse with other metrics.

5. Predicting with Article Content

While the predictions of the above methods are better than random, there is certainly room for improvement. Mining for information directly from the article content could be another useful source of information that helps news agencies curate their content. We use article content to construct a bag of words representation of the data. Since words that are either too common or too uncommon will contain little information, we remove common stop words, words that appear in over 50% of the articles, and words that appear in fewer than 0.1% of the articles. Extracting this data results in a very large set of words ($\approx$ 17,453). To narrow this down, we obtain a feature score for each word, and only keep the top 300 scoring features. We use mutual information to obtain the feature score, where the mutual information between a feature and the number of shares is:

$$MI(X,Y) = \sum_{x,y} p(x,y) log(\frac{p(x,y)}{p(x)p(y)})$$

Table 5 shows the results of this model and Table 6 lists the words with the highest feature score:

Table 5: Performance with Article Words

	Accuracy	PPV	NPV	Sensitivity	Specificity
Without words	0.66	0.67	0.66	0.65	0.67
Top 300 words	0.67	0.65	0.68	0.67	0.66

Table 6: Top Predictive Words

courtesy	facebook	2014	social	apple	mashable	nba
getty	mobile	iphone	s	game	2013	glass
istockphoto	phone	olympics	job	says	series	twitter

From this initial exploration of prediction from article content, we are not able to significantly improve on accuracy. However, the feature scores of the words does shed light on which topics are most or least popular. In particular, many technology related terms have high feature scores. One possible reason why the presence of these terms in an article may not significantly improve accuracy is because there is already a feature that captures whether an article is about technology.

6. Summary and Future Work

In this project, we predict popularity of online news articles using both regression and classification. We find that classification reaches 66% accuracy using random forests and that regression reaches 12696 RMSE using lasso. These results show predictive power in the features and are significantly better than random, however they are not overwhelmingly successful.

There are many directions we could take this project in the future. For example, we could further explore prediction from article content. In this project, we only use single words as features, but N-grams may be able to find more information about the tone and structure of articles. Other measures, like identifying and assessing the effect of "clickbait" headlines could also inform news organizations. These approaches could also be extended to other areas relevant to online news. In particular, given recent concerns over the spread of online fake news stories, similar classification algorithms could be tested on identifying and flagging such articles.

7. Acknowledgements

We would like to thank Professors Andrew Ng and John Duchi for their excellent instruction this quarter, as well as Hao Sheng and the rest of the teaching staff for their helpful comments and suggestions on this

project.

8. References

1. Fernandes, K., Vinagre, P., & Cortez, P. (2015). A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. In Portuguese Conference on Artificial Intelligence 535-546.
2. Keneshloo, Y., Wang, S., Han, E. H. S., & Ramakrishnan, N. Predicting the Popularity of News Articles.
3. UC Irvine Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets.html
4. Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Modelsvia Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.