News Article Popularity - A Predictive Task

Daniel Reznikov
drezniko@ucsd.edu

Jaskaran Virdi
jsvirdi@eng.ucsd.edu

Sudeep Nagabhirava
snagabhi@eng.ucsd.edu

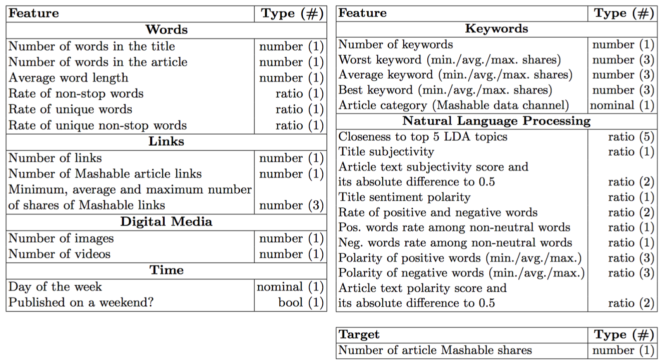
Shreyas Udupa Balekudru
sudupaba@eng.ucsd.edu

***Abstract*— In this paper, we try to predict the popularity of online news articles on `Mashable`. We first did statistical analysis on the data and showed that it is very hard to predict the popularity (number of shares) of the news articles from its text features. For our models, articles which have been shared over 1400 times are considered popular. We performed an evaluation over seven models and on comparison found that Random Forests and Adaboost, which are both ensemble models, have more predictive power than other models.**

I. Dataset Details

The selected dataset is a collection of 40,000 news articles scraped from `Mashable` (an online media content provider) over a span of 2 years from January 2013 through January 2015. In their generous donation to the UC-Irvine dataset repository, the original authors[2] include a host of features including URLs, tagged keywords, temporal data statistics, and sentiment analysis. Crucially, they include `number_of_shares` which we have targeted as the subject of our predictive learning task in this assignment.

The features of this dataset contain many interesting and useful insights. We have a total of 39,644 instances of news articles, each comprised of 61 attributes (58 predictive attributes, 2 non-predictive, 1 target field).



Feature	Type (#)
Words	
Number of words in the title	number (1)
Number of words in the article	number (1)
Average word length	number (1)
Rate of non-stop words	ratio (1)
Rate of unique words	ratio (1)
Rate of unique non-stop words	ratio (1)
Links	
Number of links	number (1)
Number of Mashable article links	number (1)
Minimum, average and maximum number of shares of Mashable links	number (3)
Digital Media	
Number of images	number (1)
Number of videos	number (1)
Time	
Day of the week	nominal (1)
Published on a weekend?	bool (1)

Feature	Type (#)
Keywords	
Number of keywords	number (1)
Worst keyword (min./avg./max. shares)	number (3)
Average keyword (min./avg./max. shares)	number (3)
Best keyword (min./avg./max. shares)	number (3)
Article category (Mashable data channel)	nominal (1)
Natural Language Processing	
Closeness to top 5 LDA topics	ratio (5)
Title subjectivity	ratio (1)
Article text subjectivity score and its absolute difference to 0.5	ratio (2)
Title sentiment polarity	ratio (1)
Rate of positive and negative words	ratio (2)
Pos. words rate among non-neutral words	ratio (1)
Neg. words rate among non-neutral words	ratio (1)
Polarity of positive words (min./avg./max.)	ratio (3)
Polarity of negative words (min./avg./max.)	ratio (3)
Article text polarity score and its absolute difference to 0.5	ratio (2)

Target	Type (#)
Number of article Mashable shares	number (1)

Fig. 1. Attribute Information

We analyzed each of the 58 predictive features and identified a few that were particularly interesting and held promise in powerfully influencing our model. From our analysis on feature statistics shown in Table I, we suspect some of the promising features include `self_reference_avg_shares` which counts the average number of shares from other articles explicitly referenced from the candidate article. Additionally, we find from computing the variance of data across the weekday feature that there is a steep drop-off of posted articles on the weekends and we hypothesize that a weekend article will have less scope to be shared broadly, affecting its popularity and sharing potential. This hypothesis is validated by plotting the number of shares as a function of the day on which it was shared. We observe that the number of shares are highest during the middle of the week and rapidly fall off during the weekends.

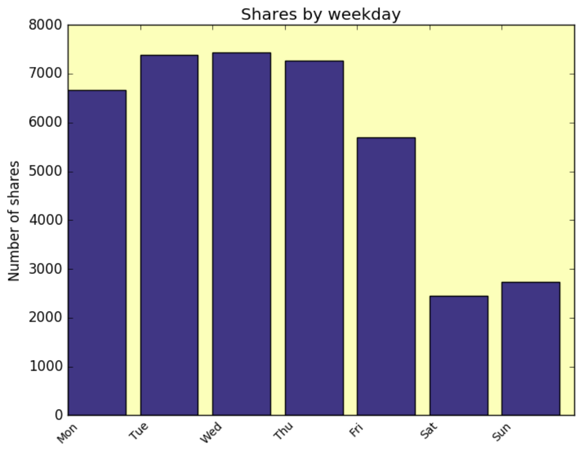


Fig. 2. Number of shares by weekday

We are interested to understand the predictive power of article sentiment analysis. For example, in our features we not only have data on the overall article polarity, but also on the min/max/avg polarity scores of the positive, negative, and neutral words.

TABLE I
Features Statistics

Feature	Min	Max	Mean	SD
n_tokens_title	2	23	10.3987	2.114
num_hrefs	0	304	10.8837	11.3319
num_imgs	0	128	4.5441	8.3093
num_videos	0	91	1.2499	4.1078
num_keywords	1	10	7.2238	1.9091
self_reference_max_shares	0	843300	10329.2127	41027.0592
self_reference_avg_sharess	0	843300	6401.6976	24211.0269
weekday_is_saturday	0	1	0.0619	0.2409
weekday_is_sunday	0	1	0.069	0.2535

We performed Principal Components Analysis (PCA) on the 58 dimensional feature space to analyze the covariance of features in our dataset. In projecting our data into PCA space we were surprised to observe that the PCA dimensions do not lend themselves well to discriminating popular vs unpopular news articles. This is an important insight for this predictive task.

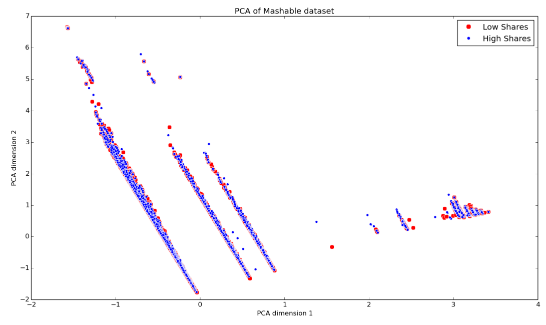


Fig. 3. PCA on 2 dimensions

Two dimensions of the feature space explain 92.6% of the variance in the data, and three dimensions of the feature space explain 96.2% of the variance in the data. In these plots it is clearly evident that our classes are not separable in PCA-space.

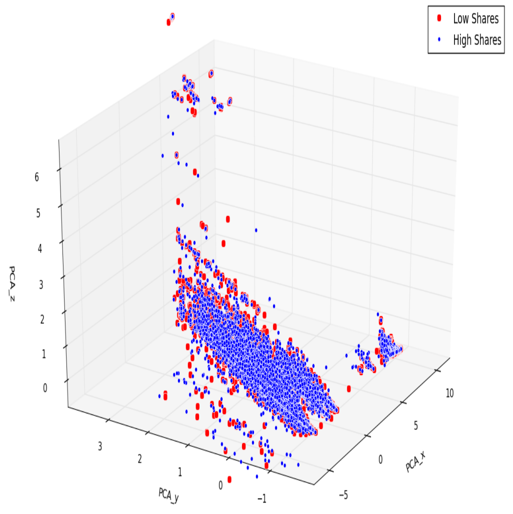


Fig. 4. PCA on 3 dimensions

As seen in figure 4, the features do not clearly demarcate the two classes. This served as the first indication that our classification results may not be exceptionally strong because classification problems discriminate powerfully where there exists a manifold which clearly separates the data classes. Indeed, as we report in this paper, despite clever feature selection, our best models only perform reasonably, never exceptionally.

II. The Predictive Task

We perform predictive analysis of popularity of a news article with binary classification. A news article with over 1400 shares is considered to be popular. We chose this `threshold` in order to equally separate popular and non-popular news articles because the median value of number of shares in the dataset is 1400.

A. Model Evaluation

We are using three metrics to compare models - Accuracy, AUC (the area under the ROC curve), and the F1-Score. We use accuracy as a measure as it directly indicates what proportion of our model predictions were right. The AUC measures the trade-off between true positive rate and false positive rate of each model. The F1-Score measures the harmonic mean of the precision and recall and is independent of the class distribution. The F1-Score assumes that the `threshold` is specified for the classification task while the ROC curve is a function of the `threshold`. The best model performs with a high AUC and a high F1-Score.

1) Accuracy: This measure is naive but still provides some intuition about the ability of a model to identify whether an article will be popular or not. This measure computes proportion of correct predictions

$$ACC = \sum \frac{isEqual(Prediction, Actual)}{\mid TestArticles \mid}$$

2) Area Under ROC Curve: For each of the 10 folds during cross-validation, we plot a line on a Receiver Operating Characteristics graph which relates the True Positive Rate (TPR) to the False Positive Rate (FPR). We compute the average area under this curve (across k-folds) and report it as AUC. This metric is a more robust measure of model performance because in addition to capturing accuracy, this metric also captures precision and recall. AUC has the added benefit of being independent of class distribution.

3) F1-Score: The F1-Score is robust because it shows an unweighted, fair measure for accuracy by taking the harmonic mean of precision and recall. We compute this metric for each of the 10 folds during cross-validation, and finally report the average of the F1-Scores.

One baseline is to compare each model to the performance of a random binary classifier which would have an accuracy of 50% and an F1 score of 0.5. For a stronger baseline, we pick a Naive Bayes classifier which operates under the assumption that the features are all conditionally independent. By training this classifier on all predictive features, we obtain a mean AUC of 0.62, an accuracy of 53.07% and an F1-Score of 0.24. Successful models would outperform these baseline scores.

The models we select are appropriate for the type of binary (popular vs unpopular) classification task that we are undertaking because each of the models proposed are designed to learn a decision boundary between classes. The models proposed are sufficiently different in their approaches to suggest that there will be variance in how well they perform. Based on our inferences from Principal Components Analysis, we predict that a non-linear model will perform better on this task. Ensemble methods with Decision Tree classifiers may prove to be more powerful for this particular classification task.

B. Feature Selection

The dataset contains 58 predictive features, 2 non-predictive features and 1 feature to be predicted. We carefully pick features for each of the proposed models. For linear regression, we use all predictive features available and then classify examples in accordance with the `threshold`. For Logistic Regression, we employ Recursive Feature Elimination (RFE) to arrive at the best twenty features to use for classification. This technique first trains the model on all features, drops 10% of the features with the least significant coefficients and iterates through the same procedure until we get the number of features we desire. We use similar methods for feature selection for the other models as well. The Random Forest classifier uses all predictive features available for the classification task. Using the Gini impurity metric, it then retrieves importance scores for the features. The top scoring features, based on the Gini impurity metric, follow in the table: II.

TABLE II
MOST IMPORTANT/PREDICTIVE FEATURES

Avg. keyword (avg. shares)	0.0441
Avg. keyword (max. shares)	0.0399
Closeness to LDA topic 2	0.0319
Is data channel 'World'?	0.0307
Best keyword (avg. shares)	0.0297
Closeness to LDA topic 1	0.0295
Closeness to LDA topic 4	0.0291
Worst keyword (avg. shares)	0.0289
Closeness to LDA topic 4	0.0288
Avg. shares of referenced articles in Mashable	0.0286

III. MODELS USED

Seven models were used to perform binary classification. For each model, we used 10-fold cross-validation to give us an accurate estimate of model prediction performance, except SVM for which 5-fold cross validation was used as the computation time was very high.

A. Linear Regression

Linear Regression was used to predict the number of shares for every news article. This was converted to a classification result based on the `threshold`. All 58 predictive features were used for this model. The weakness of this approach is that it assumes that our features are mutually independent and that our classes are linearly separable. Furthermore, we are using this regression approach on a classification problem which does not lend itself well to a high performing model.

B. Logistic Regression

RFE was used to retrieve the top 20 features for the Logistic Regression model. The regularization parameter λ used is 1.0 with L2 regularization. An advantage of this model is that we are using a non-linear activation function (sigmoid) to provide contrast to the linear separability assumption of Linear Regression.

C. Support Vector Machine

RFE was used to retrieve the top 20 features for the model and then a SVM classifier was trained with a radial basis function kernel and the regularization parameter, $C = 10$. We tried linear, polynomial, and radial kernels. We choose RBF because it yielded the best accuracy. An advantage of SVM is that it is guaranteed to converge to a global minimum. However, the computation time required is comparatively higher and it does not return probabilistic confidence.

D. Random Forest

In Random Forest, we used 100 trees and the maximum number of features considered at each decision node was $log_2(total\_features)$ = 6. Since, Random Forest is an ensemble classifier, it gives a very good generalization performance and does not overfit the data. Also, since it uses decision trees it can learn non-linear hypothesis and is in top two classifiers in terms of performance.

E. Adaboost Classifier

We used decision stump as a base estimator for Adaboost and used grid-search on 10, 50 and 100 decision stumps and got the best performance with 100 decision stumps. We used this classifier because it has a fairly good generalization performance and in addition to bagging, it uses boosting which helps in learning complex non-linear hypothesis which is necessary because the data is not linearly separable. This gave us the best performance out of all the other classifiers we used.

F. K-Nearest Neighbors (KNN)

This method determines the class of each testing sample by taking the majority vote from its k nearest training samples. To arrive at the right number of nearest neighbors to consider, we performed a search over a list of multiple k values and then picked a k value that gave the best AUC score. The strength of this model is that it is robust to fluctuations in feature values, taking an aggregate similarity measurement across features (L2-Norm). A weakness of this model is that we are making an assumption about which distance metric to use. For example, while it makes sense for real-valued features to use the Euclidean Distance, perhaps Manhattan Distance is more appropriate for one-hot encoded features like `day_of_the_week`.

G. Naive Bayes

We chose the Naive Bayes Classifier as the baseline because of its operating principle that the features are conditionally independent. The results confirm that this assumption is invalid and as such all models that mitigate this assumption will perform better allowing it to be used as a baseline. This model outperforms a random model but this assumption leads to weak discriminative power. Therefore, the other selected models can be effectively compared to the Naive Bayes baseline.

Failed Attempts

In this section, we report some of the implemented models which did not give us very good results. These failed attempts gave us better insight into the problem and helped us build more predictive models. When Linear Regression was used to predict the number of shares for each article, a very high MSE (Mean Squared Error) led us to model the problem as a classification task. When run on the top 5 PCA components, Linear Regression was not very discriminative.

In several cases, using all features for the classifier performed poorly. This led us to perform RFE (Recursive Feature Elimination) to obtain the top 20 features. Logistic Regression and k-Nearest Neighbors algorithms scored better when these features were used. The SVM classifier was also trained on only the top 20 features as it was computationally infeasible to train the classifier in a very large feature space.

IV. Literature & Domain Review

We are using an existing dataset of news articles on `www.mashable.com` The dataset was scraped by Fernandez et. all for their paper titled, "A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News"[2]. The results obtained by us are comparable to the results obtained in this paper.

The subject of news popularity has been extensively studied in related contexts, especially home-page rankings for news articles on sites like `www.yahoo.com` and `www.cnn.com` . Many aggregation of time-sensitive articles is interested in promoting content that others will find as interesting. To this end, the paper by Fernandez et. al[2] extends their models to provide the authors with recommendations on what enhancements may be made prior to publication to influence article popularity - like using different keywords, changing the length of title of the article, etc.

Additionally, we came across related work by S. Jamalli et. al. that studied, amongst other interesting things, popularity prediction for content on `Digg`[3]. There was also a past Kaggle competition[4] that used the same UC-Irvine dataset as our project, optimizing for a similar goal of popularity prediction.

Upon investigation, the bleeding edge research in this space uses Gradient Boosting and Random Forest as the most powerful predictive techniques to perform article popularity prediction. In general, success along this topic is moderate and subject heavily to strong feature selection. Our results were comparable to results of the authors of the original paper that put together the dataset that we used. But even comparable datasets and results in related papers performed within an analogous range (65%-75% accuracy).

V. Results and Conclusion

TABLE III
Model Prediction Results

Model	Accuracy	AUC	F1-Score
Adaboost	**64.50%**	**0.70**	**0.68**
Random Forest	**64.09%**	**0.70**	**0.65**
Linear Regression	54.83%	0.54	0.65
SVM	60.50%	0.66	0.63
Logistic Regression	63.04%	0.67	0.62
KNN	62.20%	0.67	0.62
Niave Bayes (Baseline)	53.07%	0.62	0.24

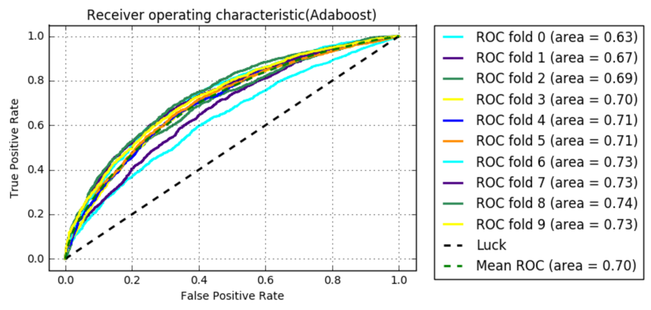


Fig. 5. ROC graph for AdaBoost Classifier Mean ROC $\bar{0}$.70

The results obtained from the models are shown in Table III. AdaBoost and Random Forest yielded better results over the other models. Using AdaBoost, the binary classification accuracy was 64.5%, the average AUC is 0.70 over all 10 folds of cross validation and the F1 score is 0.68. These results outperform our baseline scores and provide a meaningful way to model news popularity. From our analysis, we know that the features have very low discriminative power for predicting popularity.

We used two models for feature representation. For all of the models except SVM, KNN and Logistic Regression, all predictive features were included. However, for SVM, KNN and Logistic Regression, RFE was performed to pick the top 20 features. By doing this, many features which do not help in prediction are omitted. For Random Forests and AdaBoost, all features were included while training them because they are ensemble methods and they rank features based on importance in obtaining predictive results.

Random Forests performed well because the ensemble model is inherently robust to over-fitting and lends itself well to our high-dimensional data. With only a little bit of hyper-parameter tuning, it very quickly outperformed the other models.

References

[1] scikit-learn, Scikit-learn: Machine Learning in Python, Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., Journal of Machine Learning Research, volume=12, pages=2825–2830,2011

[2] Fernandes, Kelwin, Pedro Vinagre, and Paulo Cortez. ”A proactive intelligent decision support system for predicting the popularity of online news.” Portuguese Conference on Artificial Intelligence. Springer International Publishing, 2015.

[3] S. Jamali and H. Rangwala, ”Digging Digg: Comment Mining, Popularity Prediction, and Social Network Analysis,” 2009 International Conference on Web Information Systems and Mining, Shanghai, 2009, pp. 32-38.

[4] ”Predict the Popularity of an Online News Article.” Kaggle: Your Home for Data Science. N.p., 16 Mar. 2016. Web. 11 Mar. 2017. ¡https://inclass.kaggle.com/c/predicting-online-news-popularity¿.