Predicting and Evaluating the Popularity of Online News

He Ren
Department of Electrical Engineering
heren@stanford.edu

Quan Yang
Department of Electrical Engineering
quanyang@stanford.edu

Abstract—**With the expansion of the Internet, more and more people enjoys reading and sharing online news articles. The number of shares under a news article indicates how popular the news is. In this project, we intend to find the best model and set of feature to predict the popularity of online news, using machine learning techniques. Our data comes from Mashable, a well-known online news website. We implemented 10 different learning algorithms on the dataset, ranging from various regressions to SVM and Random Forest. Their performances are recorded and compared. Feature selection methods are used to improve performance and reduce features. Random Forest turns out to be the best model for prediction, and it can achieve an accuracy of 70% with optimal parameters. Our work can help online news companies to predict news popularity before publication.**

Keywords - **Machine learning; Classification; Popularity prediction; Feature selection; Model selection**

I. Introduction

In this information era, reading and sharing news have become the center of people's entertainment lives. Therefore, it would be greatly helpful if we could accurately predict the popularity of news prior to its publication, for social media workers (authors, advertisers, etc). For the purpose of this paper, we intend to make use of a largely and recently collected dataset with over 39000 articles from Mashable website, to first select informative features and then analyze and compare the performance of several machine learning algorithms.

Some prediction approaches are based on analyzing early users' comments [1], or features about post contents and domains [2]. Another proposed method [3] predicted the article's popularity not only based on its own appeal, but also other articles that it is competing with. Prediction models with SVMs, Ranking SVMs [3], Naive Bayes [2] are investigated, and more advanced algorithms such as Random Forest, Adaptive Boosting [4] could increase the precision. This paper however, incorporates a broader and more abstracter set of features, and starts with basic regression and classification models to advanced ones, with elaborations about effective feature selection.

This paper has the following structure. Section II introduces our dataset and feature selection. Section III gives our implementation of various learning algorithms. We analyze the result and compare the performances in Section IV. In Section V, we discuss possible future work.

II. Dataset & Feature Selection

A. Data collection

Our dataset is provided by UCI machine learning repository [4], originally acquired and preprocessed by K.Fernandes et al. It extracts 59 attributes (as numerical values) describing different aspects of each article, from a total of 39644 articles published in the last two years from Mashable website. The full feature set is mainly categorized as in Table 1.

B. Features

A full feature set may include much noise. We first attempted PCA for dimension reduction but it did not provide any improvements for our models, then we used filter methods (mutual information, and Fisher criterion) for feature selection and improved our prediction accuracy.

PCA is a commonly used dimensionality reduction algorithm, which could give us a lower-dimensional approximation for original dataset while preserving as much variability as possible. However, the PCA results could only made our

TABLE I. ALL AVAILABLE FEATURES

Aspects	Features
Words	Number of words of the title/content; Average word length; Rate of unique/non-stop words of contents
Links	Number of links; Number of links to other articles in Mashable
Digital Media	Number of images/videos
Publication Time	Day of the week/weekend
Keywords	Number of keywords; Worst/best/average keywords (#shares); Article category
NLP	Closeness to five LDA topics; Title/Text polarity/subjectivity; Rate and polarity of positive/negative words; Absolute subjectivity/polarity level
Target	Number of shares at Mashable

models perform worse. This is because the original feature set is well-designed and correlated information between features is limited.

Filter Methods

1) Mutual information: We calculate the mutual information $MI(x_i, y)$ between features and class labels to be the score to rank features, which can be expressed as the Kullback-Leibler(KL) divergence:

$$MI(x_i, y) = KL(p(x_i, y)||p(x_i)p(y))$$

2) Fisher criterion: Fisher criterion is another effective way in feature ranking. The Fish score (for data with two classes) for jth feature is given by:

$$F(j) = \frac{(\bar{x}_j^1 - \bar{x}_j^2)^2}{(s_j^1)^2 + (s_j^2)^2}$$

where

$$(s_j^k)^2 = \sum_{x \in X^k} (x_j - \bar{x}_j^k)^2$$

The numerator indicates the discrimination between popular and unpopular news, and the denominator indicates the scatter within each class. The larger the F-score is, the more likely this feature is more discriminative. Then we used crossed validation (with logistic regression) finding that a feature size of $k = 20$ (see Fig. 1) using F-score gives us better performance than other values of k and mutual-information-based criterion. This observation also applies to other of our models. Therefore, we used this feature set as default for the rest of project.

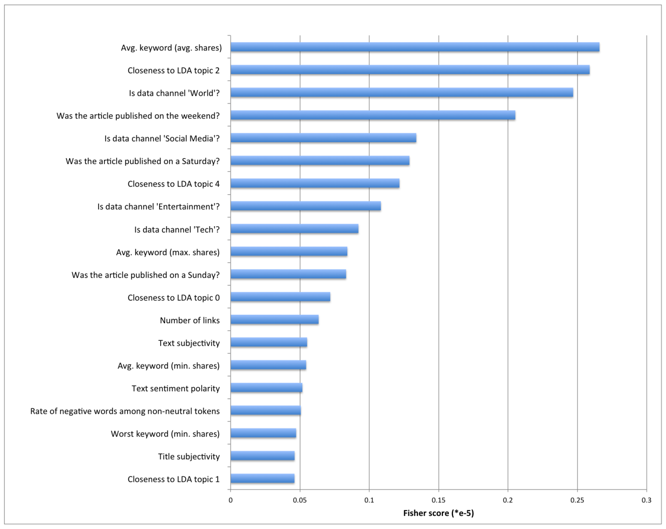

Fig. 1. Top 20 features with highest Fisher scores in feature selection

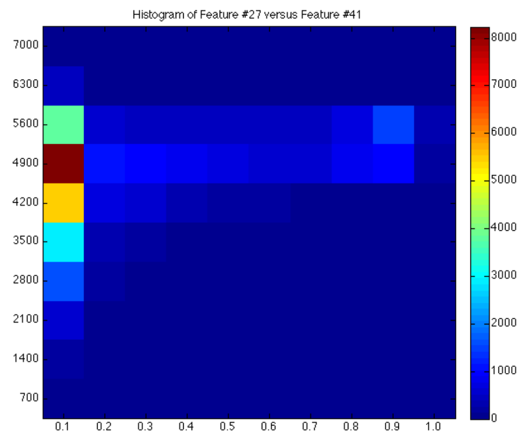

Fig. 2. 2D histogram of all data with respect to top 2 features

III. MACHINE LEARNING APPROACHES

A. Linear Regression

First we used linear regression to get a quick start. Linear regression represents a least-square fit of the response to the data. It chooses the hypothesis

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i$$

by minimizing the cost function

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Due to the high variance of the target variable (number of shares), direct application of linear regression was not acceptable, specifically, on testing samples, only 20% prediction values (number of shares) are within 1000 of the actual results. We discretized the target value to binary categories (as in Table 2), and consider a prediction correct if its value and the actual result have the same sign (both + or -). It gave us 66% accuracy. Although we applied the regression model on a classification problem, the result is quite desirable.

TABLE II. Categories and classifications for linear rg. and logistic rg.

Number of shares	$0 - 1400$	> 1400
Linear Rg. Categories	-1	1
Logistic Rg. Classification	0	1

B. Logistic Regression

We then use classification model trying to improve our accuracy further. For logistic regression, the hypothesis is

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

and parameters are chosen as to maximize their likelihood

$$\prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

We classified the data as in Table 2 and used stochastic gradient ascent rule to implement it, and we got similar result as linear regression model.

For multinomial classifications (say k classes), the model uses logarithmic function to estimate the relative probability of each category with reference to the kth category. For example, for $k = 3$ (i. e. we classify the data into 3 categories, “unpopular", “popular", “very popular"), we comparing the follows:

$$\log\left(\frac{P(y^{(i)} = 1)}{P(y^{(i)} = 3)}\right) = \sum_{k=1}^{n}\beta_{k,1}x_{k-1}^{(i)}$$

$$\log\left(\frac{P(y^{(i)} = 2)}{P(y^{(i)} = 3)}\right) = \sum_{k=1}^{n}\beta_{k,2}x_{k-1}^{(i)}$$

and the prediction result is the class with the largest probability. Both generalization and training error increases with the increasing k, (e.g., for k = 3, logistic regression gives 51.6% accuracy). Since we mainly focused on predicting whether a news would be popular or not, we did not go further in multiclass problems.

C. Support Vector Machine

We started SVM with linear kernel, which use the following formula to make predictions:

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}\right)^T x + b \\ &= \sum_{i=1}^{m}\alpha_i y^{(i)}\left\langle x^{(i)}, x\right\rangle + b \end{aligned}$$

In the equation, the kernel can be replaced with more complex ones, as long as:

$$K(x,y) = \phi(x)^T\phi(y)$$

TABLE III. SVM Kernels we used

Kernel	Parameter	Expression
Linear	None	$K(x,y) = x^T y$
Polynomial	Degree d	$K(x,y) = (x^T y + 1)^d$
Gaussian	σ	$K(x,y) = \exp(\frac{-\lvert x-y\rvert^2}{2\sigma^2})$

The kernels we used are in Table 3. The reason we use different kernels is because linear kernel has high bias problem. Polynomial and gaussian kernels can operate in a high-dimensional, implicit feature space without computing the coordinates of the data in that space. In this way, they can offer more flexible decision boundaries.

D. Random Forest

In bagging (Bootstrap Aggregation), numerous replicates of the original dataset are created to reduce the variance in prediction. Random Forest use multiple decision trees which are built on separate sets of examples drawn from the dataset. In each tree, we can use a subset of all the features we have. By using more decision trees and averaging the result, the variance of the model can be greatly lowered. Given a training set

$x^{(1)}, x^{(2)}, \cdots, x^{(n)}$ with responses $y^{(1)}, y^{(2)}, \cdots, y^{(n)}$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits treees to these examples [6]:

- For $b = 1, \cdots, B$:
  1. Sample, with replacement, n training examples called X_b, Y_b
  2. Train a decision or regression tree f_b on X_b, Y_b
- After training, predictions for unseen examples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x')$$

or by taking the majority vote in the case of decision trees.

For Random Forest, there are two main parameters to be considered: number of trees and number of features they select at each decision point. Theoretically, accuracy will increase with more trees making decision. We use cross validation to see how the performance changes with these parameters. We ensured that every value within a certain range that our computer can support is tested, and the result is plotted. In this case we are able to see exactly the relationship between performance and parameters.

IV. Results

In this project, we implemented 10 different machine learning models. In this section, we apply 5-fold cross validation to models and compare their performances. Their accuracy and recall (sensitivity) are listed in Table 4. We went deeper into how parameter affects performance for SVM and Random Forest, since they have more parameters to consider.

As a classification model, logistic regression achieves a decent accuracy, better than most of the models. Its Receiver Operating Characteristic Curve is shown in Fig. 3. The AUC value is 0.74, which means logistic regression gives fairly good result.

By observing training and test error, we saw that SVM with linear kernel has high bias problem. Therefore we used more complex kernels and trained SVM on full feature set to solve the problem [5]. The result is shown in Table 5.

TABLE IV. Performance of different algorithms

Algorithms	Accuracy	Recall
Linear Regression	0.66	0.67
Logistic Regression	0.66	0.70
SVM ($d = 9$ Poly Kernel)	0.55	0.45
Random Forest (500 Trees)	**0.69**	**0.71**
k-Nearest Neighbors ($k = 5$)	0.56	0.47
SVR (Linear Kernel)	0.52	0.59
REPTree	0.67	0.62
Kernel Partial Least Square	0.58	0.60
Kernel Perceptron (Max loop 100)	0.45	0.99
C4.5 Algorithm	0.58	0.59

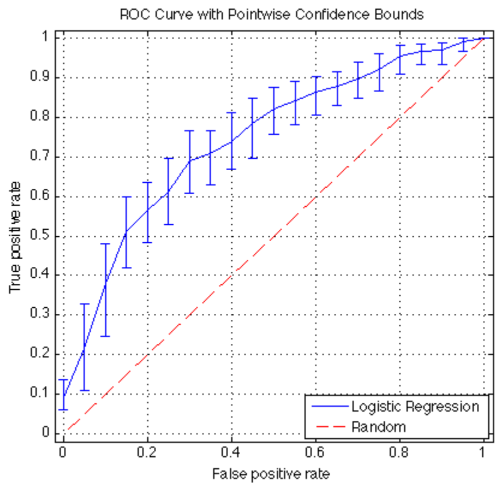

Fig. 3. ROC curve of logistic regression with confidence bounds

Even if we use high degree polynomial kernels, high bias problem is slightly mitigated while accuracy seems to reach a bottleneck. As we plot the training set on various combinations of features, examples from two classes always mix together, and it shows no potential boundary. We infer that the data is not separable enough for SVM to handle, even for extremely high degree polynomial kernels.

For SVM with 9-degree polynomial kernel, which is empirically an optimal setting here for SVM, Fig. 4 shows how test error and training error change with increasing number of training

TABLE V. SVM Results with various kernels

Kernel	Linear	Poly ($d = 7$)	Poly ($d = 20$)	Gaussian
Test Error	0.48	0.47	0.45	0.46
Training Error	0.45	0.45	0.45	0.47

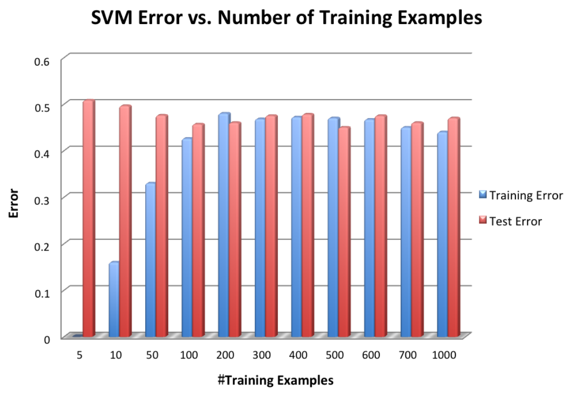

Fig. 4. SVM error with increasing number of training examples

examples. This is the best result SVM can give, and its accuracy doesn't improve with more training examples.

Random Forest has the best result for this classification problem. It can have different number of decision trees and different number of features used for each decision point. The number of training examples can also change. Therefore, implementation should be done in a systematic way. We change only one variable at a time. We first use a default setting for Random Forest and increase the number of training examples. The error decreases to a certain level as shown in Fig. 5 (left). Then we set the number of trees to be constant, and change the number of features used for decision. It turns out that $\log(N_{max})$ is the best value. Finally, we change the number of trees continuously from 5 to 500, and plot the result in Fig. 5 (right). The accuracy reaches a limit of 69%, which is the best among all algorithms.

V. Future Work

As is seen from the result, no algorithm can reach 70% accuracy given the data set we have, even though they are state-of-the-art. To improve accuracy, there is little room in model selection but much room in feature selection. In the preprocessing round, 59 features were extracted from news articles, and our later work is based on these features. However, the content of news articles hasn't been fully explored. Some features are related to the content, such as LDA topics (feature #39 - #43), which are convenient to use for learning, but reflect only a small portion of information about the content.

In the future, we could directly treat all the words in an article as additional features, and then apply machine learning algorithms like Naive Bayes and SVM. In this way, what the article really talks about is taken into account, and this approach should improve the accuracy of prediction if combined with our current work.

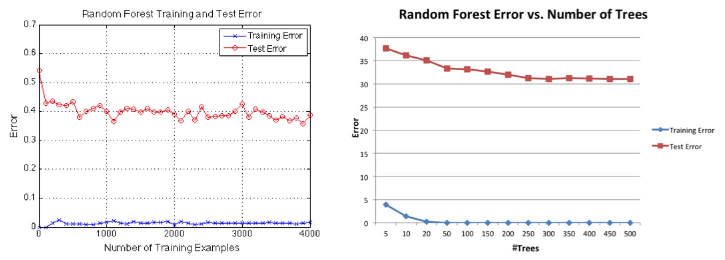

Fig. 5. Random Forest training and test error with respect to number of training examples (left) and number of decision trees (right)

References

[1] Tatar, Alexandru, et al. "Predicting the popularity of online articles based on user comments." Proceedings of the International Conference on Web Intelligence, Mining and Semantics. ACM, 2011.

[2] "Predicting the Popularity of Social News Posts." 2013 cs229 projects. Joe Maguire Scott Michelson.

[3] Hensinger, Elena, Ilias Flaounas, and Nello Cristianini. "Modelling and predicting news popularity." Pattern Analysis and Applications 16.4 (2013): 623-635.

[4] K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. *Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence*, September, Coimbra, Portugal.

[5] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: A library for support vector machines." *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011): 27.

[6] James, Gareth, et al. *An introduction to statistical learning*. New York: springer, 2013.