News Popularity Classification for Mashable

News Yorker, Center for Data Science, New York University

Dian Zhang (dz584), Lyuang Fu (lf1664), Junge Zhang (jz3502), Yichao Shen (ys3197)

1. Business Understanding

Online news has become an increasingly prevalent method to convey information in the Digital Age. People are getting used to browsing news during their fragmented time like breakfast or commute, instead of spending more time on traditional media. Therefore, how to push the potentially popular news to readers efficiently is very crucial for the news media, since news with high popularity can directly increase the click rate of the website and boost the website's profit generated from the advertisement displayed. It will also increase readers' stickiness to the website as the readers would recognize the website as an effective source for popular news, and less likely to switch to other online news media.

The project targets on building a classifier to identify the news that can easily raise readers' attention and become popular, using data about the online news popularity of a specific website, namely Mashable. Since Mashable is a comprehensive but medium-size online news website, we believe the model can be applied to similar-scale news websites generally. These news websites can adopt this model to assist them in predicting whether the news would become popular, and then push the potentially popular news content to readers more effectively and timely, while creating a better news reading experience at the same time. This would further increase the overall website traffic, resulting in increasing reader subscriptions, advertisement exposures and click-through, and ultimately, more profits. We selected and trained to mine the data include KNN, Decision Tree, Logistic Regression, Support Vector Machine, Random Forest and Gradient Boosting. The models will use a variety of features such as number of words in the news title, to predict whether the news is popular or not. We used AUC and TPR as standards to evaluate the models.

2. Data Understanding

Our dataset came from UCI machine learning repository, which summarized a heterogeneous set of features about articles published by Mashable in a period of two years. The original source of dataset was used in Professor Fernandes, Vinagre and Cortez's paper to predict the number of shares in social networks[1][2]. All the articles were published by Mashable, so the dataset did not share the original content but only statistics associated with it. Based on the academic background and reputation of UCI machine learning repository and Mashable, we believed our data are reliable, public and real, and they are fittable for our business problem. The table of original features' name, description as well as the data type was listed in appendix. There are 61 columns, including 60 features and one target variable, which named "number of shares". This dataset has 39797 rows, which is large enough for generalizability. Intuitively there are too many features and we believed many of the features need to be dropped. Details of feature engineering will be mentioned later in data preparation. We don't see obvious selection bias from our dataset as Mashable is a global, multi-platform media which covers almost all the fields. Although the data were not up-to-date, it is not a serious concern for us as we believed the generalizability of our model is not significantly influenced by the time.

3. Data Preparation

Our data came from one single csv file containing 61 columns, which generally forced us to focus on target variable definition and feature engineering before any splitting of the data. Since the purpose of our project was to discover news to be posted on Mashable, the target column was "shares". The "shares" column originally includes number of shares for each piece of news. In order to make "shares" more intuitive for the online news website, we intended to transform the

"shares" column to binary classification values, namely as "popular" and "unpopular". To divide the "shares" column in a reasonable way, we plotted the histogram of the "shares" column to inspect its distribution. From the histogram shown on Figure 1, it is a right skewed distribution and we believe it is reasonable to choose the third quartile as the separate threshold of the defined popularity for our project. In another word, news with top 25% number of shares were defined as popular news and others were defined as unpopular news, which also transformed our project as a binary classification problem.

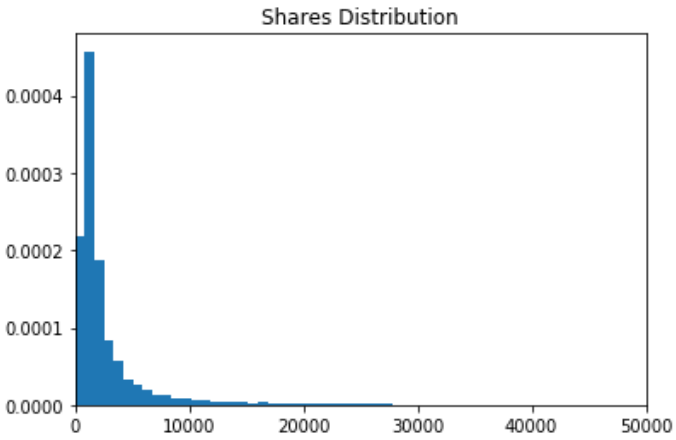

Figure 1: The "shares" Column Distribution

While 60 features seemed too redundant for the models we tended to build, we completed the feature selection in both qualitative and quantitative way. First of all, we dropped the "url" column and the "timedelta" column. While they each represent the web url of the news and the days that the news has been posted, it makes no sense for a business manager to predict the popularity based on them. For those features left, we considered two ways of feature engineering including the Recursive Feature Elimination (RFE) and Singular Value Decomposition (SVD). RFE recursively chooses smaller subsets of features based on either coefficients or feature importance from a chosen model[3]. Instead of the initially introduced SVM recursive feature elimination[4], we by

comparison in predictive model performance, selected Decision Tree as our RFE model and extracted 12 variables as final features. In addition, by using SVD, it yielded 40 features to be used as the final data frame. However, in the later comparison, while RFE and SVD performed similarly, we took the higher memory cost of SVD into account. This resulted in using RFE as our final feature selection method and transformed the data into 12 features and one target variable.

The next step of data preparation was to split data into four parts: training data, validation data, training plus validation data, and test data. We first split whole data to training plus validation data and test data with the proportion of 9:1. Meanwhile, we controlled the test data to have the same popularity proportion as the original data to model the real situation. For the train plus validation data, we further split it to training data and validation data with the proportion of 8:2.

At last, although our target variable is not extremely skewed to one class, we still decided to down sample the training data and training plus validation data to 1:1 proportion between popular and unpopular news, based on our main evaluation metric, TPR (discussed in 4.1), Table 1 shows final TPRs of thee models (discussed in 4.2, 4.3, 4.4) before and after downsampling. It is apparent that downsampling has improved TPRs of models. We believed the reason was down sampling helped models capture more predictive information on popular news (positive instances) in our case. Therefore, we down sampled data to ensure better TPRs in our predictive models.

Table 1: Model TPRs before and after Down Sampling

Model	TPR before Down Sampling	TPR after Down Sampling
Decision Tree	0.089	0.687
SVM	0	0.694
Random Forest	0.114	0.686

4. Modeling & Evaluation

4.1 Evaluation Metrics

Two evaluation metrics were employed through the model process both for the hyperparameter choices and model selection. One evaluation metric is AUC, which is a conservative evaluation metric for the model accuracy. The other evaluation metric that more relates to the business is TPR. In our case, TPR measures the ability of the classifier to identify true positive instances (popular news). The higher TPR means the model performs better on finding out real popular news which can lead to more clicks, and then higher profit for Mashable.

4.2 Baseline Model

Decision Tree is one of the basic data mining algorithms, which establishes a tree system to split data based on information gain value. It is easy to interpret and realize Decision Tree, however, the performance is not stable and the model is easy to overfit data. Careful tuning of parameters and leaf pruning are necessary.

For our classification problem. We decided to use Decision Tree as the baseline model. One reason was the convenience of usage as mentioned, and another was that there were many different scaled data in the features. Some are numerical values, such as the number of tokens in the title and some are binary values, 0 stands for false and 1 stands for true. In such situation, Decision Tree model would be a reasonable model to fit the data. The parameter for Decision Tree includes max_depth and min_samples_leaf. To get the best performance, two lists for these two parameters are used respectively.

By running different sets of max_depth and min_samples_leaf through validation set, model with leaf = 300, depth =800 gained highest value in both AUC and TPR. Figure 2 shows the graphic

interpretation. Final test on the holdout set showed that the baseline decision tree model gave an AUC of 0.686 and TPR of 66.2%.

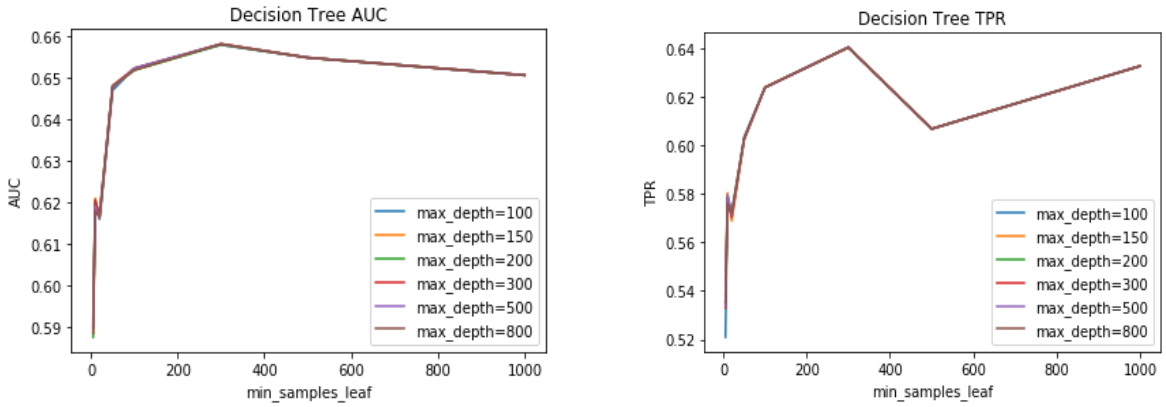

Figure 2: Decision Tree AUC and TPR Based Hyperparameters

4.3 Support Vector Machine (SVM)

SVM is another powerful data mining algorithm for classification. It builds a hyperplane to distinguish the data and has a strong geometric interpretation.[5] The drawbacks are that hyperparameters should be tuned and need sufficient data to ensure the quality of performance. Since the training dataset contains over 14,000 samples, we can expect a relatively good performance of SVM.

Sufficient evidence proves the fact that SVM has a robust and decent performance when the amount of data is big, therefore we choose SVM as second classifier desired to use.

The kernel of SVM classifier is the first and prime parameter to tune, with three options of 'linear', 'rbf' and 'poly'. Generally, 'rbf' and 'poly' are two nonlinear kernel functions commonly used in SVM classifier. Compared with linear kernel, these two kernel options can enhance the robustness of SVM for the complex real world data. In addition, the regularization factor C plays an important role in reducing model's variance and improves the classifier accuracy. We set C value to be from 10^{-5} to 10^{0} and did an iterative test on the different values of C. Notably, we also included C value

= 10^1 in the original iteration. However, it led to significantly longer training time. The possible reason was that our data set had comparatively a large number of features. A larger C value means less regularization, and it leads to more weighted features, overfitting and inefficient training time. Thus, we removed it from the C value iteration.

Figure 3 shows the AUCs and TPRs of SVM classifiers with corresponding kernels for the same range of regularization factor C. It turned out that SVM with nonlinear kernel outperformed linear kernel and 'rbf' beated 'poly' in the final stage of comparison with the highest AUC of about 0.679. The corresponding regularization factor C is 10^{-1}. On the best TPR standard, the corresponding kernel is also 'rbf' with the C=10^{-2} with TPR of 67.9%

The final test on hold-out set showed a decent performance of our SVM model. The model with best AUC score in validation set (kernel = 'rbf', C = 10^{-1}) got the AUC of 0.69 and TPR of 68.1% respectively. On the other hand, the model with best TPR in validation set (kernel = 'rbf', C = 10^{-2}) got the AUC of 0.677 and TPR of 66.86%. Since SVM with kernel = 'rbf', C = 10^{-1} got a better performance in both AUC and TPR, we considered it as the best SVM model.

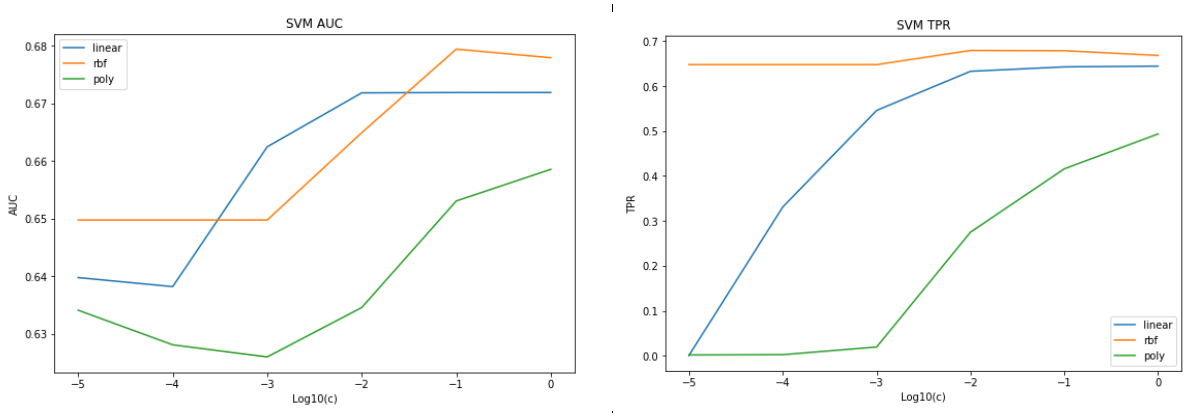

Figure 3: SVM AUC and TPR Based Hyperparameters

4.4 Random Forest

As an upgrading transformation of Decision Tree, Random Forest fit bagged Decision Trees, which means each Decision Tree grows based on bootstrapped training samples. It learns the data with random samples of predictors so that the trees are decorrelated to decrease the total variance, but with additional computational cost.[6]

Since the dataset is relatively large, n_estimators = 200 was a reasonable number of trees needed to learn the data. A candidate list of [10,100,200,1000,2000,5000] was shared for both max_depth and min_samples_leaf. Figure 4 shows that the size of min_samples_leaf had the negative relationship with AUC score and the size of max_depth did not impact too much on the AUC. On the other hand, TPR hit the maximum when min_samples_leaf is large with smaller max_depth.

When testing through the validation set, the model with max depth=1000 and min_samples_leaf=10 achieved the best AUC and the model with max depth=10 and min_samples_leaf=5000 achieved the best TPR. We restored these two sets of hyperparameters and applied them on the final test set. The best Random Forest model turned out to be the first one (max depth=200 and min_samples_leaf=10), which had the AUC of 0.710 and the TPR of 68.6%.

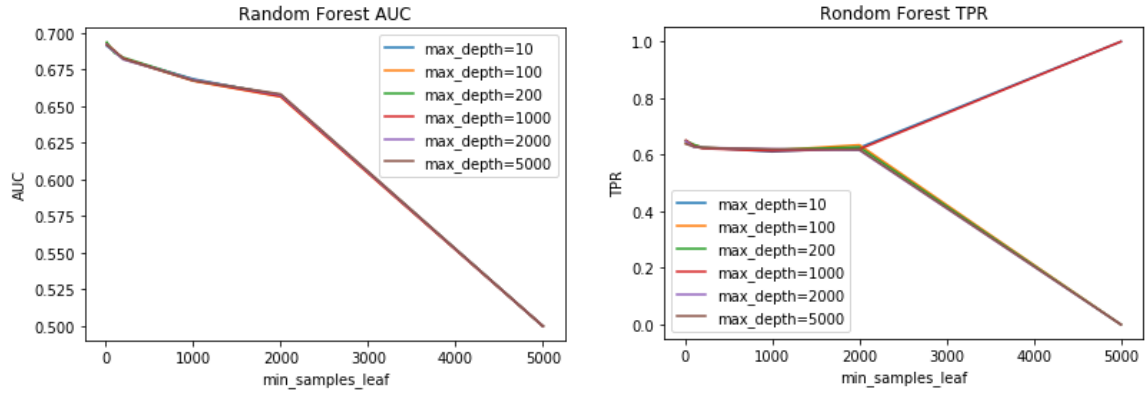

Figure 4: Random Forest AUC and TPR Based Hyperparameters

4.5 Model Conclusion

Comparing the above models, while Random Forest had the best AUC of 0.71 and TPR of 68.6%, SVM had only a slightly lower AUC of 0.694 and TPR of 68,1%. However, Considering the similarity of performance of two models but the lower computational cost of SVM than Random Forest[7], we selected SVM as our optimal model candidate while Random Forest as the backup model.

Other than the top-performing and representative models discussed above, we also have built models including KNN, Logistic Regression and Gradient Boosting. Nevertheless, due to the limited space, we chose not to show them in the main body. The full table of final hyperparameters, AUCs and TPRs of all models can be seen at the Appendix.

5. Deployment

5.1 Model Deployment and Actual Evaluation

One of the key evaluation metrics of the model is the TPR which means we desire Mashable to deploy the model to predict popular news (positive instances) at a high accuracy. In another word, without approximate 70% TPR in our ultimately selected SVM model, it means we can be relatively confident to post news that are predicted as popular on the website where can be more easily accessible to users. For example, the predicted popular news can be put on the first page of the website to acquire more potential profit for the business from more clicks and advertisement impressions.

In the actual production system, the evaluation should still be based on TPR. In other words, the key operational question for the model deployment is whether those predicted popular models are indeed popular. An intuitive way is that the website can set a click or profit standard assessment

to inspect whether the predicted popular news can retrieve expected clicks or profits in a certain period. If the result is not good enough, we may either consider using the alternative Random Forest model if the computational cost does not hurt too much or we may collect more data to fit models again,

5.2 Ethical Issues and Risk Control

As the data is merely concerned with news without any benefit conflicts against individual user, ethical issues are not a big consideration in our model. However, one thing worth to mention about our model is that our ultimate model has about 70% TPR. Although this definitely will not incur direct profit loss. 7 out of 10 real popular news may not optimize the profit for a growing business. To mitigate this unoptimized issue, the model deployment should combine with the business sense and experience from the business manager to screen predicted news before they are actually posted, which may increase the possibility of real popular news.

Reference

[1] UCI Machine Learning Repository. "Online News Popularity Data Set". [online] Available at: http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity. [Accessed 6 Dec, 2018].

[2] 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.

[3] Scikitlearn. "*sklearn.feature_selection.RFE"*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#Sklearn.feature_selection.RFE [Accessed 6 Dec, 2018].

[4] Guyon, I., Weston, J., Barnhill, S. and Vapnik., V. (2002). Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning*, 46(1-3), pp. 389-422.

[5] Scikitlearn. "*sklearn.svm.SVC"*. [online] Available at: https://scikit-learn.org/stable/modules/svm.html [Accessed 7 Dec, 2018].

[6] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). An *Introduction to Statistical Learning with Applications in R.* New York: Springer.

[7] Wainer, J. (2016). Comparison of 14 Different Families of Classification Algorithms on 115 Binary Datasets. Available at: https://arxiv.org/pdf/1606.00930.pdf [Accessed 6 Dec, 2018].

Appendix

Tables

Table 2: Original Feature Table

feature name	feature description	data type
url	URL of the article (non-predictive)	object
timedelta	Days between the article publication and the dataset acquisition (non-predictive)	float64
n_tokens_title	Number of words in the title	float64
n_tokens_content	Number of words in the content	float64
n_unique_tokens	Rate of unique words in the content	float64
n_non_stop_words	Rate of non-stop words in the content	float64
n_non_stop_unique_tokens	Rate of unique non-stop words in the content	float64
num_hrefs	Number of links	float64
num_self_hrefs	Number of links to other articles published by Mashable	float64
num_imgs	Number of images	float64
num_videos	Number of videos	float64
average_token_length	Average length of the words in the content	float64
num_keywords	Number of keywords in the metadata	float64
data_channel_is_lifestyle	Is data channel 'Lifestyle'?	float64

data_channel_is_entertainment	Is data channel 'Entertainment'?	float64
data_channel_is_bus	Is data channel 'Business'?	float64
data_channel_is_socmed	Is data channel 'Social Media'?	float64
data_channel_is_tech	Is data channel 'Tech'?	float64
data_channel_is_world	Is data channel 'World'?	float64
kw_min_min	Worst keyword (min. shares)	float64
kw_max_min	Worst keyword (max. shares)	float64
kw_avg_min	Worst keyword (avg. shares)	float64
kw_min_max	Best keyword (min. shares)	float64
kw_max_max	Best keyword (max. shares)	float64
kw_avg_max	Best keyword (avg. shares)	float64
kw_min_avg	Avg. keyword (min. shares)	float64
kw_max_avg	Avg. keyword (max. shares)	float64
kw_avg_avg	Avg. keyword (avg. shares)	float64
self_reference_min_shares	Min. shares of referenced articles in Mashable	float64
self_reference_max_shares	Max. shares of referenced articles in Mashable	float64
self_reference_avg_sharess	Avg. shares of referenced articles in Mashable	float64
weekday_is_monday	Was the article published on a Monday?	float64

weekday_is_tuesday	Was the article published on a Tuesday?	float64
weekday_is_wednesday	Was the article published on a Wednesday?	float64
weekday_is_thursday	Was the article published on a Thursday?	float64
weekday_is_friday	Was the article published on a Friday?	float64
weekday_is_saturday	Was the article published on a Saturday?	float64
weekday_is_sunday	Was the article published on a Sunday?	float64
is_weekend	Was the article published on the weekend?	float64
LDA_00	Closeness to LDA topic 0	float64
LDA_01	Closeness to LDA topic 1	float64
LDA_02	Closeness to LDA topic 2	float64
LDA_03	Closeness to LDA topic 3	float64
LDA_04	Closeness to LDA topic 4	float64
global_subjectivity	Text subjectivity	float64
global_sentiment_polarity	Text sentiment polarity	float64
global_rate_positive_words	Rate of positive words in the content	float64
global_rate_negative_words	Rate of negative words in the content	float64
rate_positive_words	Rate of positive words among non-neutral tokens	float64
rate_negative_words	Rate of negative words among non-neutral tokens	float64
avg_positive_polarity	Avg. polarity of positive words	float64

min_positive_polarity	Min. polarity of positive words	float64
max_positive_polarity	Max. polarity of positive words	float64
avg_negative_polarity	Avg. polarity of negative words	float64
min_negative_polarity	Min. polarity of negative words	float64
max_negative_polarity	Max. polarity of negative words	float64
title_subjectivity	Title subjectivity	float64
title_sentiment_polarity	Title polarity	float64
abs_title_subjectivity	Absolute subjectivity level	float64
abs_title_sentiment_polarity	Absolute polarity level	float64
shares	Number of shares (target)	int64

Table 3: Hyperparameters, AUCs and TPRs of All Models

Model	Hyperparameter	AUC	TPR
KNN	k=9	0.583	0.534
Decision Tree	max_depth=800, min_samples_leaf=300	0.687	0.662
Logistic Regression	penalty=L1, C=1.0	0.676	0.582
SVM	kernel=rbf, C=0.1	0.694	0.681
Random Forest	max_depth=1000, min_samples_leaf=10	0.710	0.686
Gradient Boosting	max_depth=10, min_samples_leaf=1000	0.707	0.681

Contribution

As a group, we delivery detailed work to everyone. Here is the description.

Junge Zhang:

- Data cleaning and feature engineering
- Fit data to Gradient Boosting and Random Forest model and tune hyperparameter

- Write “Data Preparing” and “Deployment” of the report

Yichao Shen:

- Fit the data to Knn and Decision Tree model and tune hyperparameter
- Write “Model and Evaluation” of the report

Lyuang Fu:

- Fit the data to Logistic Regression model and tune hyperparameter
- Write “Data Understanding” of the report

Dian Zhang:

- Fit the data to SVM model and tune hyperparameter
- Write “Business Understanding” of the report

Coding

All of coding is available on our GitHub pages, one link can be referred to is https://github.com/d1s0rder/Online-News-Popularity-Prediction.