Capstone Project: Predicting Popularity of Online News Articles

Yimeng Dong

1 Definition

1.1 Project Overview

With the help of Internet, the online news can be instantly spread around the world. Most of peoples now have the habit of reading and sharing news online, for instance, using social media like Twitter and Facebook. Typically, the news popularity can be indicated by the number of reads, likes or shares. For the online news stake holders such as content providers or advertisers, it's very valuable if the popularity of the news articles can be accurately predicted prior to the publication. Thus, it is interesting and meaningful to use the machine learning techniques to predict the popularity of online news articles.

Various works have been done in prediction of online news popularity. In [1], the popularity of online articles is analyzed based on the users' comments. [2] defines the popularity in terms of a competition where the popular articles are those which were the most appealing on that particular day. Ranking Support Vector Machine (SVM) is used to classify the appealing/non appealing of online news article. In [3], the number of retweets is predicted using both the features of the retweet content (length, words, number of hashtag, etc.) and the features of author (number of followers, friends, etc.). [4] collects a dataset with almost 40,000 articles from the Mashable website, compares five different methods on classifying popular/unpopular news articles and concludes that the Random Forest (RF) can achieve the best performance.

In this project, based on the dataset including 39,643 news articles from website *Mashable*, we will try to find the best classification learning algorithm to accurately predict if a news article will become popular or not prior to publication.

1.2 Problem Statement

In this project, we will use machine learning techniques to solve a binary classification problem, which is to predict if an online news article will become popular or not prior to publication. The popularity is characterized by the number of shares. If the number of shares is higher than a pre-defined threshold, the article is labeled as popular, otherwise it is labeled as unpopular. Thus, the problem is to utilize a list of articles's features and find the best machine learning model to accurately classify the target label (popular/unpopular) of the articles to be published. As the problem can be formulated as a binary classification problem, we will implement and compare three classification learning algorithms including Logistic Regression, RF, and Adaboost. The best model will be selected based on the metrics introduced in next part.

1.3 Metrics

As a classification task, we will adopt the following three evaluation metrics: accuracy, F1-score and AUC. For all three metrics, the higher value of the metric means the better performance of model.

(a) Accuracy: Accuracy is direct indication of the proportion of correct classification. It considers both true positives and true negatives with equal weight and it can be computed as

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{dataset size}} \tag{1}$$

Although the measure of accuracy might be naive when the data class distribution is highly skewed, but it is still an intuitive indication of model's performance.

(b) F1-score: F1-score is an unweighted measure for accuracy by taking harmonic mean of precision and recall, which can be computed as

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2}$$

It is a robust measurement since it is independent of data class distribution.

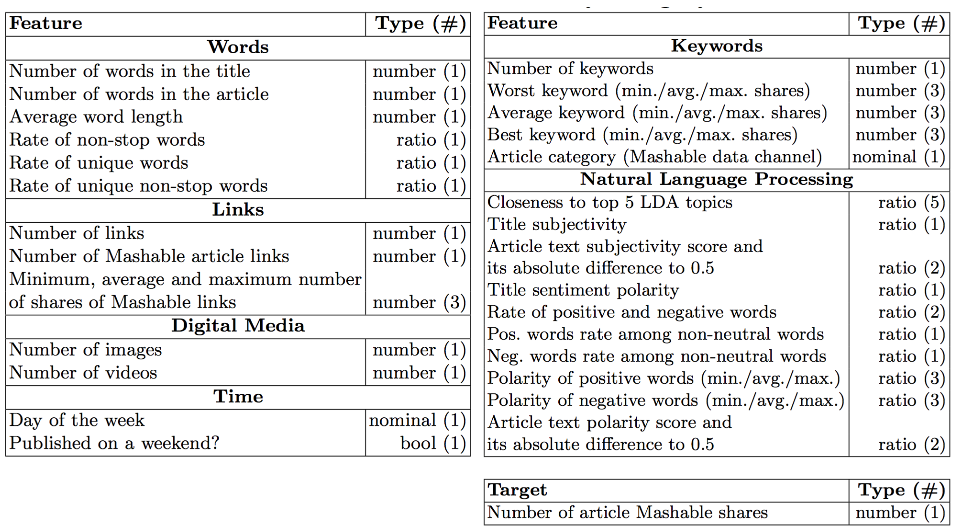
(c) AUC: The AUC is the area under the ROC (Receiver Operating Characteristics) curve, which is a plot of the True Positive Rate versus the False Positive Rate. AUC value is a good measure of classifier's discrimination power and it is a more robust measure for model performance.

2 Analysis

2.1 Data Exploration and Visualization

The dataset is consisted of 39,643 news articles from an online news website called *Mashable* collected over 2 years from Jan. 2013 to Jan. 2015. It is downloaded from UCI Machine Learning Repository as `https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity#` and this dataset is generously denoted by the author of [4].

For each instance of the dataset, it has 61 attributes which includes 1 target attribute (number of shares), 2 non-predictive features (URL of the article and Days between the article publication and the dataset acquisition) and 58 predictive features as shown in Fig. 1. The dataset has already been initially preprocessed. For examples, the categorical features like the published day of the week and article category have been transformed by one-hot encoding scheme, and the skewed feature like number of words in the article has been log-transformed.



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)

Figure 1: List of predictive attributes of dataset. [4]

Firstly, we need to determine the appropriate threshold for number of shares to discriminate the news to be popular or unpopular. So we show the statistics of the target attribute "shares" in Fig. 2, and we find the median of target attribute is 1,400, thus it is reasonable if we take 1,400 as a threshold. Then we can use this threshold to convert the continuous number target attribute into a boolean label.

```
count     39644.000000
mean       3395.380184
std       11626.950749
min           1.000000
25%         946.000000
50%        1400.000000
75%        2800.000000
max      843300.000000
Name:  shares, dtype: float64
```

Figure 2: Statistics of target attribute "shares".

By observing various features, I think there are several relevant features like day of the week and article category. In Fig. 3, the count of popular/unpopular news over different day of the week is plotted. We can clearly find that the articles published over the weekends has larger potential to be popular. It makes sense because it is very likely that people will spend more time online browsing the news over the weekends. In Fig. 4, the count of popular/unpopular news over different article category is plotted. We can observer that in category of technology ("data_channel_is_tech") and social media ("data_channel_is_socmed"), the proportion of popular news is much larger the unpopular ones, and in category of world ("data_channel_is_world") and entertainment ("data_channel_is_entertainment"), the proportion of unpopular news is larger the popular ones. This might reflects that the readers of *Mashable* prefer the channel of technology and social media much over the channel of world and entertainment.

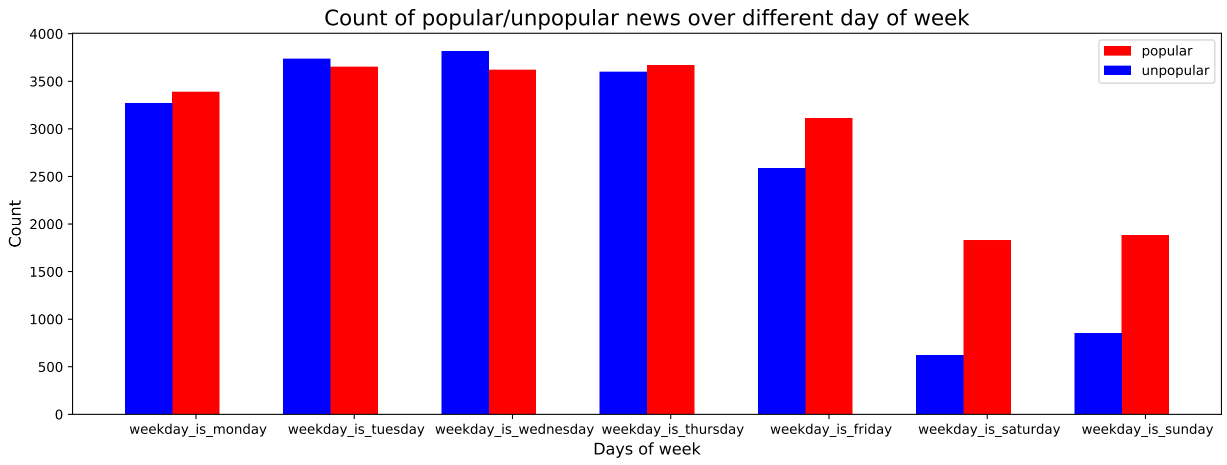


Figure 3: Count of popular/unpopular news over different day of week.

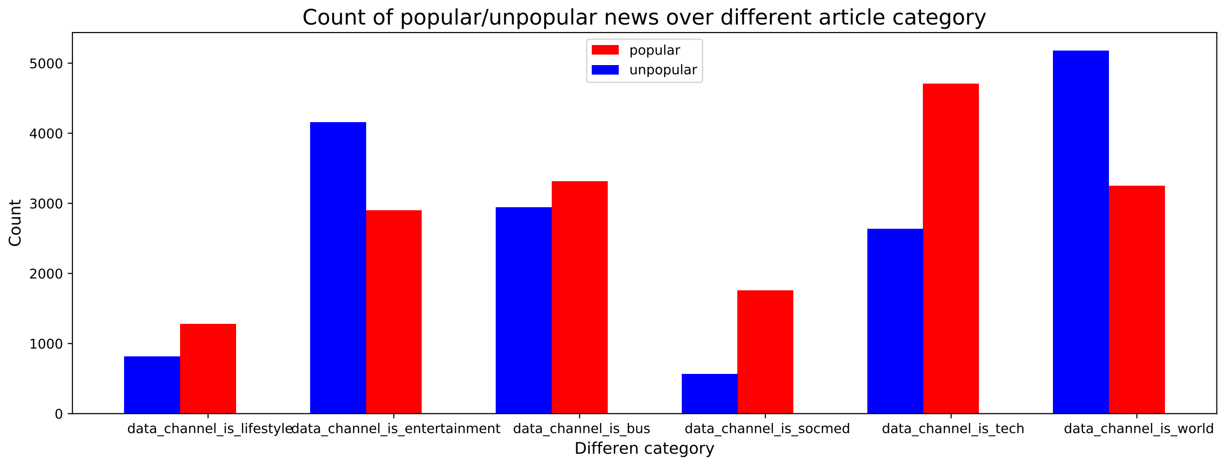
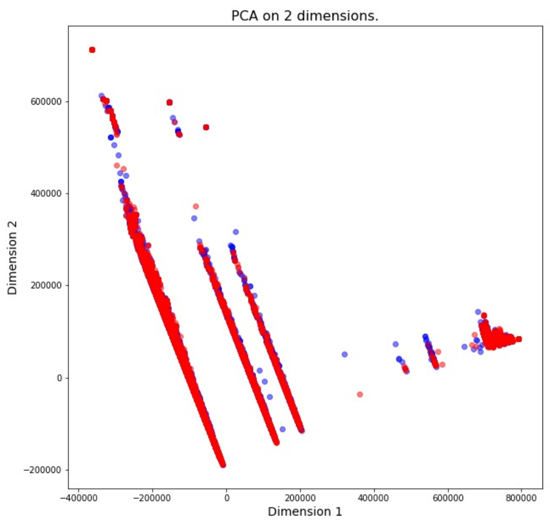
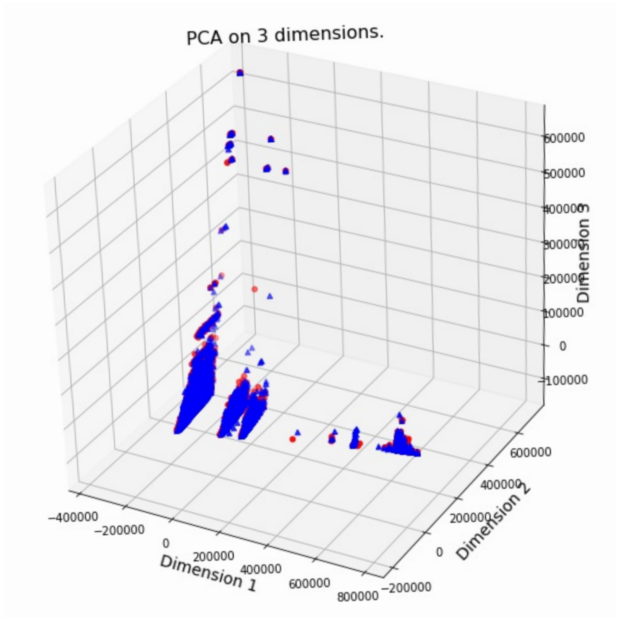


Figure 4: Count of popular/unpopular news over different article category.

Next, we do the principle component analysis (PCA) to visualize the data. As shown in Fig. 5, we project the data point onto first 2 and 3 principle components, respectively. It is clear that the dataset is not linearly separable in PCA space.



(a)



(b)

Figure 5: (a) Data projected on first 2 principle components (b) Data projected on first 3 principle components. Red: popular; Blue: unpopular

2.2 Algorithms and Techniques

Since we formulate this problem as a binary classification problem as stated in Section 1.2. In this project, three classification learning algorithms including Logistic Regression, RF, and Adaboost will be implemented and compared based on the evaluation metric such as accuracy, F1-score and Area Under ROC Curve (AUC). All the learning algorithm will be implemented by sklearn toolbox in this project.

2.2.1 Logistic Regression

Logistic regression is a linear model for classification. In this model, the output of a single trial can be interpreted as class probability, which is generated by a logistic function or sigmoid function. sklearn provides a function called LogisticRegression(). One typical tunable hyperparameter is ”C”, which is the inverse of regularization strength. The advantage of logistic regression is that the training and predicting speed is very fast.

2.2.2 RF

RF is an ensemble classifier, it fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

sklearn provides a function called RandomForestClassifier(). One typical tunable parameter is called ”n_estimators”, which represents the number of trees in the forest.

2.2.3 Adaboost

AdaBoost classifier is an ensemble classifier that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. sklearn provides a function called AdaBoostClassifier(). One tunable parameters is ”n_estimators”, which is the maximum number of estimators at which boosting is terminated. Another is ”learning_rate”, which means the contribution of each classifier shrinks by the value of ”learning_rate”. There is a trade-off between ”n_estimators” and ”learning_rate”. The boosting can help in learning complex non-linear hypothesis which is necessary because the dataset here is not linearly separable.

In this project, we will first implement all three algorithms with the default hyperparameter settings, then we will use grid search method to refine the hyperparameters and thereby improve the model's performance. For logistic regression, we will tune the hyperparameter ”C”, which is the inverse of regularization strength. ”C” is basically used for control over-fitting. A smaller ”C” means higher regularization over the model parameters, thus smaller ”C” will decrease the magnitude of model parameters to prevent over-fitting. For RF, we will tune hyperparameter ”n_estimators”, which is the maximum number of trees you can build before taking the maximum voting or averages of predictions. Higher number of trees can give better performance but makes the code running slower. For Adaboost, the default base estimator is decision tree classifier. Similar to RF, hyperparameter ”n_estimators” represents the maximum number of trees you can build before termination. Higher number of trees may lead to better performance but will decrease the model running speed. The trade-off between ”n_estimators” and ”learning_rate” is as follows: a larger ”learning_rate” typically means smaller ”n_estimators” in Adaboost but might lead to over-fitting, while a smaller ”learning_rate” typically means larger ”n_estimators” and it will control over-fitting but slow down the running speed.

2.3 Benchmark

For this project, we can take the dataset's donator's work [4] as a benchmark model. By carefully tuning the hyperparameters, Ref. [4] use RF model to achieve 0.67 of accuracy score, 0.69 of F1 score and 0.73 of AUC score. Based on the dataset, I will try to construct the best model, which can achieve comparable performance as the benchmark model.

3 Methodology

3.1 Data Preprocessing

As mentioned in Section 2.1, some data preprocessing works have been done by the data's donator. The categorical features like the published day of the week and article category have been transformed by one-hot encoding scheme, and the skewed feature like number of

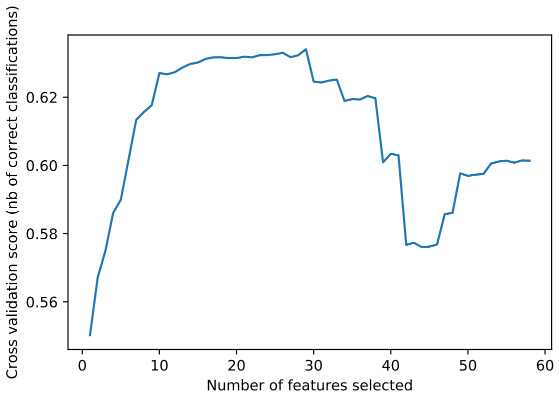


Figure 6: The cross validation score versus the number of feature selected using RFECV for logistic regression estimator.

words in the article has been log-transformed. Based on this, I further preprocess the dataset by normalizing the numerical feature to the interval $[0, 1]$ such that each feature is treated equally when applying supervised learning. I also select the median of target attribute as the threshold to convert the continuous target attribute to boolean label.

Since there are 58 features in the dataset, it is reasonable to conduct a feature selection to reduce the data noise and increase the algorithm running speed. One effective way is using recursive feature elimination with cross validation (RFECV) to automatically select the most significant features for certain classifier. sklearn provides a function called REFCV() that can help us.

Firstly, we run RFECV with a logistic regression estimator. The cross validation score versus the number of feature selected is shown in Fig. 6. From the figure, we can find there is drop of score when number of features is 29. Thus RFECV algorithm select 29 most relevant features from 58 original features. The selected 29 features are listed in Fig. 7. Interestingly, the day of week and article category features are included in these 29 features as we discussed and visualized in Section 2.1.

n_tokens_title	n_non_stop_words	rate_negative_words	average_token_length
data_channel_is_entertainment	title_sentiment_polarity	weekday_is_thursday	min_negative_polarity
data_channel_is_socmed	weekday_is_wednesday	LDA_00	weekday_is_monday
data_channel_is_tech	is_weekend	LDA_01	weekday_is_tuesday
data_channel_is_world	LDA_02	LDA_04	weekday_is_saturday
avg_negative_polarity	avg_positive_polarity	abs_title_subjectivity	weekday_is_Sunday
n_unique_tokens	num_keywords	kw_min_min	kw_min_avg
kw_avg_avg			

Figure 7: 29 feature selected using RFECV with logistic regression estimator.

Next, we run RFECV with a RF estimator. The cross validation score versus the number of feature selected is shown in Fig. 8. We find that RFECV selects 56 features for RF, which is almost the full features in dataset. The only two features RFECV excludes for RF are [’ n_non_stop_words’, ’ data_channel_is_lifestyle’].

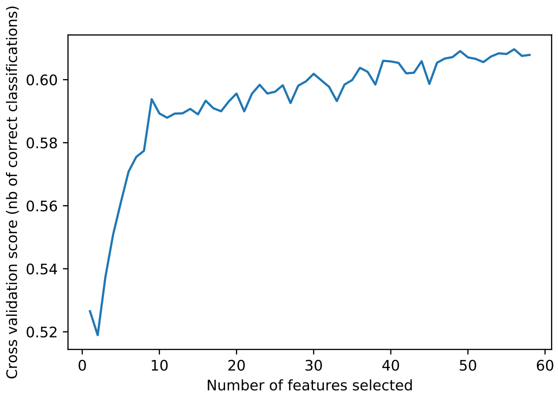


Figure 8: The cross validation score versus the number of feature selected using RFECV with RF estimator.

Lastly, we run RFECV with Adaboost estimator. The cross validation score versus the number of feature selected is shown in Fig. 9. From the figure, we can find there is a peak when number of features is 18. Thus RFECV algorithm select 18 most relevant features from 58 original features. The selected 18 features are listed in Fig. 10. In the next section, I will implement the classification algorithms using their own selected features.

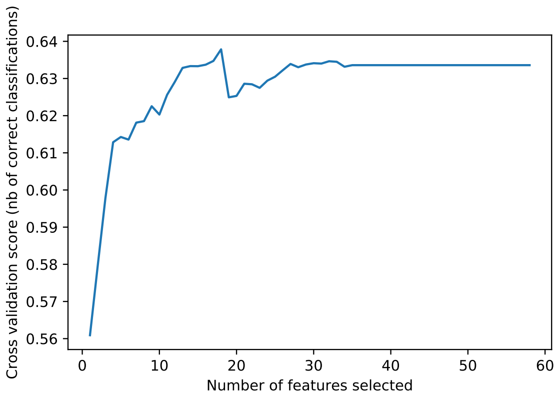


Figure 9: The cross validation score versus the number of feature selected using RFECV with Adaboost estimator.

3.2 Implementation

Before algorithm implementation, for each algorithm, I also randomly split dataset with its own selected features into training set (90%) and testing set (10%). The logistic regression, RF and Adaboost are implemented by the sklearn function LogisticRegression(), RandomForestClassifier() and AdaBoostClassifier(), respectively. At this stage, I will first use the default setting for model hyperparameters. The default hyperparameters are logistic regression - {"C": 1.0}; RF-{"n_estimators": 10}; Adaboost-{"n_estimators": 50, "learning_rate": 1.0}. I will try to refine the parameters in next part. With the selected feature for each

n_non_stop_unique_tokens	num_self_hrefs	num_imgs	num_videos
data_channel_is_entertainment	num_keywords	kw_avg_min	kw_min_min
data_channel_is_socmed	kw_min_max	kw_max_max	kw_max_avg
global_sentiment_polarity	kw_avg_avg	self_reference_min_shares	self_reference_max_shares
global_subjectivity	kw_min_avg		

Figure 10: 18 feature selected using RFECV with Adaboost estimator.

algorithm, I run the three classification algorithms and their performance is shown in Fig. 11.

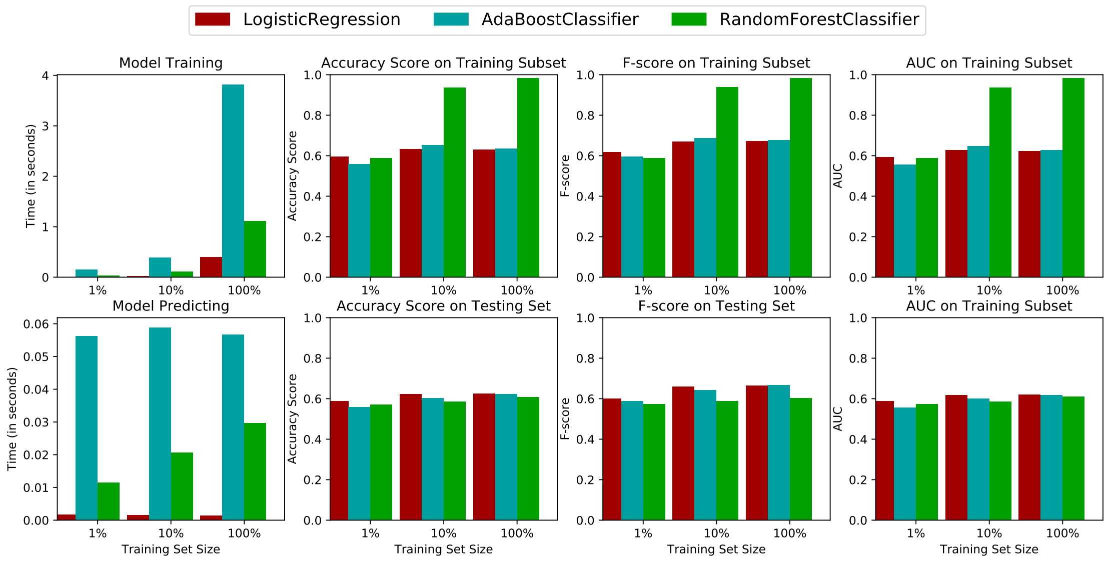


Figure 11: Performance of three classifiers under default parameter setting.

The three metrics (accuracy, F1-score and AUC) are summarized in Fig 12. Under default parameter setting, Adaboost performs best in all three metrics, RF performs better than logistic regression in AUC while logistic regression performs better than RF in accuracy and F1-score. As for the training and testing speed, logistic regression is much faster than the other two, and RF runs faster than Adaboost.

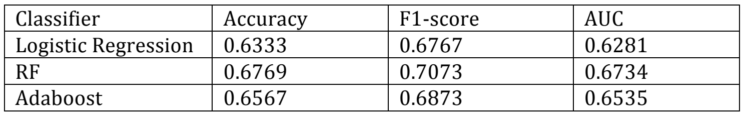
Classifier	Accuracy	F1-score	AUC
Logistic Regression	0.6323	0.6754	0.6271
RF	0.6308	0.6347	0.6318
Adaboost	0.6542	0.6832	0.6513

Figure 12: Metrics score of three classifiers under default hyperparameters.

3.3 Refinement

In this part, I will use the grid search method for each of the three classifiers to refine their hyperparameters. The grid search method exhaustively search through all possible combinations of model parameters, cross validate the model and then determine which set of model parameters gives the best performance. Since the grid search can help select an optimal model parameter, thus the learning algorithm can be optimized. In sklearn, we can use the function GridSearchCV() to implement grid search. The selected gird search ranges for hyperparameters are: logistic regression - {"C": [0.1,0.5,1,2.5,5,10]}; RF-{"n_estimators": [475,500,525]}; Adaboost-{"n_estimators": [200,300,500,600], "learning_rate": [0.1,0.5,1]}.

After running the grid the search, the refined hyperparameters are obtained as logistic regression - {"C": 2.5}; RF-{"n_estimators": 500}; Adaboost-{"n_estimators": 300, "learning_rate": 0.5}. Now we run the three classifiers with refined parameters, the three metrics scores are summarized in Fig. 13. Compared with the metrics in Fig. 12, we can find the metrics of logistic regression and Adaboost are just slightly improved, but the metrics of random forest are significantly improved after tuning by grid search.



Classifier	Accuracy	F1-score	AUC
Logistic Regression	0.6333	0.6767	0.6281
RF	0.6769	0.7073	0.6734
Adaboost	0.6567	0.6873	0.6535

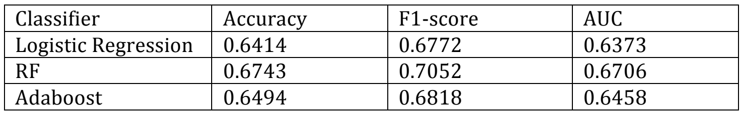
Figure 13: Metrics score of three classifiers under refined hyperparameters.

4 Results

4.1 Model Evaluation and Validation

After initial implementation and further refinement for the three classifiers, we find the best performance is obtained by the RF classifier with 500 trees in the forest. The best obtained metrics of RF are accuracy 0.6769, F1-score 0.7073 and AUC 0.6734. The final scores are not exceptional, which is sort of within the expectation, because the dataset not linear separable as shown in the PCA in Section 2.1. But it still achieve a reasonable performance in news popularity prediction compared with a random guess.

To test the robustness of the model, we change split ratio of training/testing set from 0.1 to 0.15, and then run the three classified with the same refined hyperparameters. Now the metrics are shown in Fig. 14. Compared with Fig. 13, the performance of the model is still similar and the best performance is still given by RF.

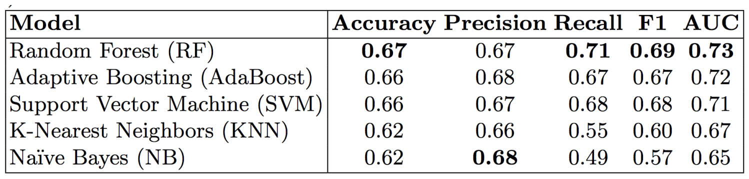


Classifier	Accuracy	F1-score	AUC
Logistic Regression	0.6414	0.6772	0.6373
RF	0.6743	0.7052	0.6706
Adaboost	0.6494	0.6818	0.6458

Figure 14: Test the model with training/testing set ratio 0.15.

4.2 Justification

Fig. 15 displays the final metrics obtained from the benchmark work of [4] as we discussed in Section 2.3. The best performance is also given by RF model, which achieves 0.67 of accuracy score, 0.69 of F1 score and 0.73 of AUC score. The metrics of my RF model are accuracy 0.6769, F1-score 0.7073 and AUC 0.6734. By comparison, although the AUC score is not better than the benchmark model, the accuracy and F1-score are all better than the benchmark model. So we can say the obtained model achieves a comparable performance as benchmark model and it is significant enough to solve the popular news classification problem.



Model	**Accuracy**	**Precision**	**Recall**	**F1**	**AUC**
Random Forest (RF)	**0.67**	0.67	**0.71**	**0.69**	**0.73**
Adaptive Boosting (AdaBoost)	0.66	0.68	0.67	0.67	0.72
Support Vector Machine (SVM)	0.66	0.67	0.68	0.68	0.71
K-Nearest Neighbors (KNN)	0.62	0.66	0.55	0.60	0.67
Naïve Bayes (NB)	0.62	**0.68**	0.49	0.57	0.65

Figure 15: The metrics of benchmark model in [4].

5 Conclusion

5.1 Free-Form Visualization

In this part, the performance of three classifiers with refined hyperparameters is finally displayed in Fig. 16. As shown in the visualization, the training and testing time of RF is greatly increased since 500 trees are used in the forest, but it helps RF achieve the best performance in terms of accuracy, F1-score and AUC.

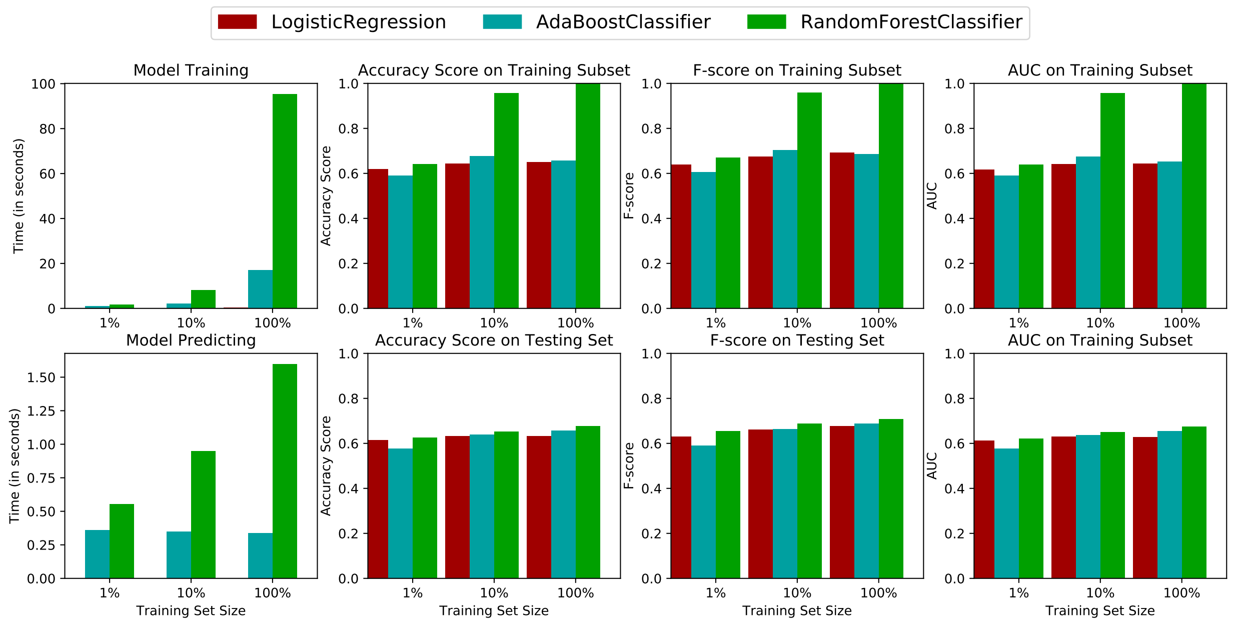


Figure 16: Performance of three classifiers under refined parameter setting.

5.2 Reflection

To summarize, the project is conducted through following steps:

(a) Data collection: The dataset of some 40,000 online news articles are downloaded from UCI Machine Learning Repository, which is originally collected and donated by the author of [4].

(b) Data preprocessing: Based on the initial data preprocessing in original dataset, I further preprocess the dataset by normalizing the numerical feature such that each feature is treated equally when applying supervised learning. I also select the median of the target attribute (number of shares) as an appropriate threshold to label all the data as either popular or unpopular.

(c) Data exploration and visualization: I explore the relevance of certain feature by visualization. And I also visualize the data distribution by PCA.

(d) Feature selection: To select the most relevant features among all 58 features, I use RFECV to select the most relevant features for each of the classifier.

(e) Classifier implementation and refinement: The classification algorithms including Logistic Regression, RF and Adaboost are implemented using sklearn. Then the model's hyperparameters are tuned by grid search method.

(f) Model evaluation and validation: The refined models are evaluated and compared using three metrics (accuracy, F1-score, AUC). The performance is also compared with the benchmark model.

The interesting aspect of this project is from the feature selection part. As in Section 2.1, I assume the day of the week and article category could be the relevant features by analysis and visualization. And later in the part of feature selection by RFECV, my guess is verified, which also shows the effectiveness of RFECV. The difficult part of the project is (1) how to define a problem, collect the corresponding data and engineer/select the relevant features; (2) how to choose appropriate learning algorithm and refine the hyperparameters. Since the obtained model achieve a fairly good performance, I think RF model with a refined hyperparameter should be used in a general setting to solve these types of problems.

5.3 Improvement

To further improve the performance, I think there are three possible ways: (1) increase the size of dataset since RF has a strong learning capability and a rich dataset might improve its prediction performance; (2) try more advanced cross validation methods although it might increase the training time; (3) engineer and add more relevant features to the original dataset. For instance, we could use all the words in an article as additional features, and then try the classifier such as Naive Bayes to see if it can achieve a better performance.

References

[1] A. Tatar, J. Leguay, P. Antoniadis, A. Limbourg, M. D. de Amorim, and S. Fdida, "Predicting the popularity of online articles based on user comments," in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. ACM, 2011, p. 67.

[2] E. Hensinger, I. Flaounas, and N. Cristianini, "Modelling and predicting news popularity," *Pattern Analysis and Applications*, vol. 16, no. 4, pp. 623–635, 2013.

[3] S. Petrovic, M. Osborne, and V. Lavrenko, "Rt to win! predicting message propagation in twitter." *ICWSM*, vol. 11, pp. 586–589, 2011.

[4] K. Fernandes, P. Vinagre, and P. Cortez, "A proactive intelligent decision support system for predicting the popularity of online news," in *Portuguese Conference on Artificial Intelligence*. Springer, 2015, pp. 535–546.