4Twos - Predicting the Popularity of Online News

Hengyu Tang (ht1162) Nhung Le (nhl256) Shirley Xu (xx852) Shradha Taneja (st3277)
Center for Data Science, New York University

Contents

1. Business Understanding

2. Data Understanding
    2.1. Data Collection
    2.2. Data Exploration
    2.3. Selection Bias
    2.4. Outliers

3. Data Preparation and Analysis
    3.1. Target Variable
        3.1.1. Regression
        3.1.2. Binary Classification
        3.1.3. Multi-class Classification
    3.2. Feature Engineering:
        3.2.1. Correlations
        3.2.2. Feature Selection
        3.2.3. Feature Innovations

4. Modeling & Evaluation
    4.1. Regression
        4.1.1. Baseline Model, Performance, and Evaluation
        4.1.2. Model Selection and Parameter Tuning
        4.1.3. Results
    4.2. Binary Classification
        4.2.1. Baseline Model, Performance, and Evaluation
        4.2.2. Model Selection and Parameter Tuning
        4.2.3. Result
    4.3. Multi-class Classification
        4.3.1. Threshold Selection
        4.3.2 Model Selection and Parameter Tuning

5. Deployment

6. Conclusion & Future Work

Appendix

Nhung Le (nhl256) Hengyu Tang (ht1162) Shirley Xu (xx852) Shradha Taneja (st3277)

Abstract **- In this project, we investigated into feature engineering and experimented with various models to predict popularity of online news. The booming of online media highlights the business needs to foresee the effects of articles prior to their publication. We employed three methods for feature selection: Greedy Search, Feature Importances of tree-based models, and Recursive Feature Elimination. For regression, we adopted Multiple Linear Regression, Lasso Regression, and XGBoost to predict the exact number of shares. For classification, we used Logistic Regression, Decision Trees, and various tree-based ensemble methods to classify articles into 2 and 3 groups. All the models are fine tuned with grid search and cross validation and evaluated using accuracy and AUC.**

1.Business Understanding

With explosion in information, how to grab audiences' attention and effectively deliver ideas becomes both a technical concern and a business problem. Wider social influence nowadays usually results in higher future profits and sound brand reputations. It is hence a demanding task for authors and editors of online news websites to identify potentially popular articles. It is also more crucial to identify these articles prior to their publications, as modifications after release bear moderate results. If a model describing key features for larger shares is offered, authors will better cater their articles to the popular trends among the audiences hence to effectively attract interest and deliver ideas. Moreover, such model is also beneficial for publishers as it allows the company to maximize their revenues. It is worth investigating as external elements such as publication times and publication categories also affect the popularity of an article. Strategic decisions can be made to better marketing reaches, such as marketing less popular articles at peak time for better responses. Revenue models can hence be better incorporated into articles without risking losing audience interests, a win-win opportunity for both publishers and readers.

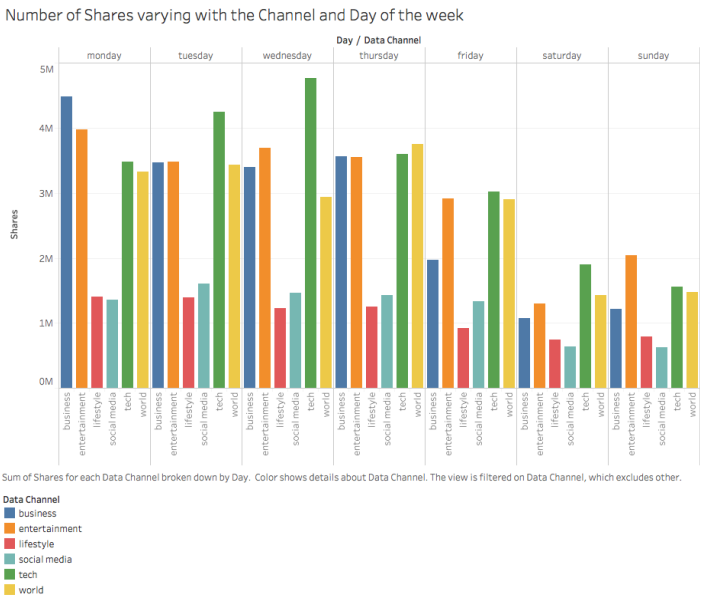
2. Data Understanding

2.1. Data Collection

We obtained the dataset from the UCI Machine Learning Repository. The original dataset contains 39,644 observations and each observation has 61 attributes (58 predictive attributes, 2 non-predictive attributes, 1 target) that describe different aspects of an article from the Mashable website. The dataset was originally acquired and preprocessed by Fernandes et al. and the predictive features were categorized into 6 groups: Publication time, Digital media, Words, Links, Keywords, Natural language processing features.

2.2 Data Exploration

We decided to play with the data a little, and created some visualizations with Tableau in order to confirm some of the hypothesis we had. For example: the articles related to fashion and/or travel published on the weekends might get more shares as people are free on weekends and spend more time on leisure activities.



The graph above confirms our hypothesis, we see business articles have more shares on Monday as compared to any other day and on weekends Entertainment articles receive more shares compared to any other data channel. (More visualizations can be found on github)

2.3. Selection Bias

The dataset is based on articles published between 2013 and 2014, hence might not capture features of

recent articles well. The removal of original articles restricted us from further verifying the data validity. The dataset was also preprocessed. With original data missing, we were unable to assess the representativeness of the articles chosen to in the dataset, hence it is difficult for us to measure the selection bias during the data mining process.

2.4. Outliers

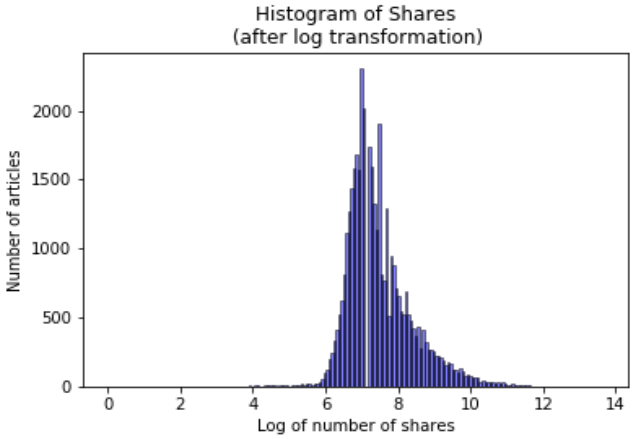
Visualization analysis shows outliers. We will discard outliers if those only account for a small percentage of the entire dataset. For example, we discarded outliers of 'num_hrefs' and 'num_videos' since they make up of less than 1% of the dataset.

3. Data Preparation and Analysis

3.1. Target Variable

3.1.1. Regression

The histogram of the target variable `shares` shows that the distribution of number of shares has an extremely long tail. Indeed, on average, an article receives 3,395 shares, while the number of shares for some highly popular articles ranges from 600,000 to 843,300. Therefore, we decided to transform the target variable (`shares`) for the regression by the logarithm function.



3.1.2. Binary Classification

We split the target variable using its median value (shares = 1400) for binary classification so the two classes are balanced. Therefore, the newly created target variable 'popular' flags 1 if the number of shares is greater than 1,400 and 0 otherwise.

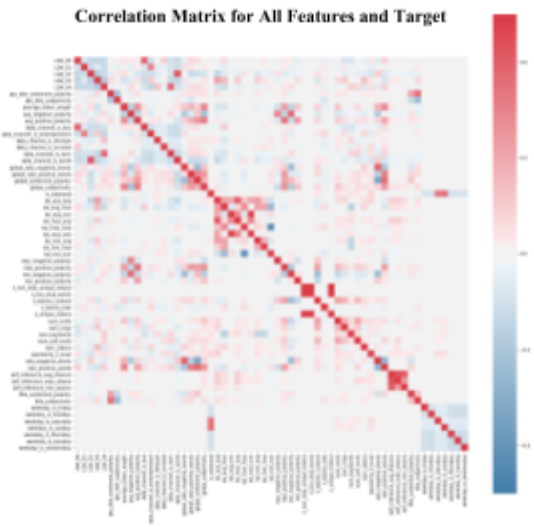
3.1.3. Multi-class Classification

The best method to discretize the popularity is to set the thresholds at one standard deviation away from the mean of the log transformed shares data because the log-transformation of the target variable follows a normal distribution. The thresholds are 695, 4470 with class sizes: 3693, 30092, 5859.

3.2. Feature Engineering

3.2.1. Correlations

Before applying more rigid methods for feature selection, we first explored the correlations among the features and the target and found that the keyword based features and the Natural Language Processing features are most effective in predicting the popularity of an article.



3.2.2. Feature Selection

- **Greedy Search**

We used AdaBoosting classifiers for greedy search, with mean + stderr of logloss as metric. For each iteration, we trained the model with current best feature list and added new feature that produced smallest mean + stderr logloss. The stopping criteria is when increase in metric is less than 0.00001 and we avoided arbitrarily deciding on the optimal size of feature list. However, when carrying out greedy search, it is not ensured that the combination of best features of current time steps will generate the best feature lists over all time steps. Moreover, such

greedy search is very computationally expensive. Therefore, such method may not be suitable to apply in all use scenarios when the number of features grow. The iteration result is shown in Table 1.

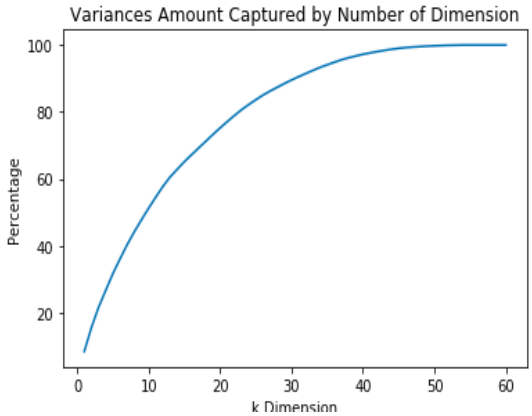
Best feature per iteration	LogLoss (mean+stderr)	Improvement
'kw_avg_max'	0.6872	0
'LDA_00',	0.6836	-0.0037
'LDA_01',	0.6805	-0.0031
'global_subjectivity'	0.6779	-0.0026
...	...	...
'self_reference_avg_sharess'	0.6641	-0.00026
'self_reference_min_shares'	0.6639	-0.00015
'num_hrefs'	0.6639	-0.00

Table 1. Greedy Search Result

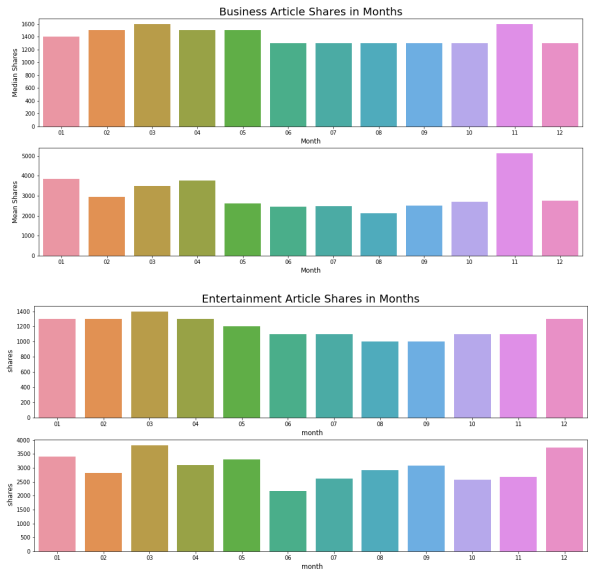
- **Feature importances of tree-based models**
  Given that we used various tree-based models for the classification task, we adopted the feature importances of Decision Tree, Random Forest, and AdaBoosting, then selected 15 features that have high feature importances values on all the lists. Although this summoning process reduces number of features, it brings in extra selection bias.
- **Recursive Feature Elimination**
  We further experimented with RFE using Logistic Regression and Random Forest. During the elimination process, the least important feature is discarded after each iteration until reaching the desired size of the feature set. However, this method is relatively computationally expensive to run. We failed to run RFE using SVM as the computation took too long.

3.2.3. Feature Innovations

- **PCA**
  Principal Component Analysis (PCA) is a dimensionality reduction method, resulting in fewer relationships between predictors to consider, thus making the model less likely to overfit. As shown in the graph below, if we reduce our number of features from 58 to 30, we are able to capture ~90% of all variances in our data. If we use 45 features, we are able to capture up to 97% of all variances in our data. With the goal to avoid overfitting while capturing most of the information of our data, we chose k = 30 (i.e., reduce our feature space from 58 dimensions to 30 dimensions)



- **New Features**
  Further investigation shows that the entertainment articles reached a peak at May and December, which corresponds well to the time period when people planned for upcoming vacations. The business focused articles showed a steep decrease during December, revealing the gap of holiday seasons. Therefore, we encoded the month information into the dataset deriving from feature 'url' using one-hot-encoding. However, the feature selections process shows little importance of the month features in articles' popularity. The graph belows shows the median and mean shares of the Business and Entertainment articles.



4. Modeling & Evaluation

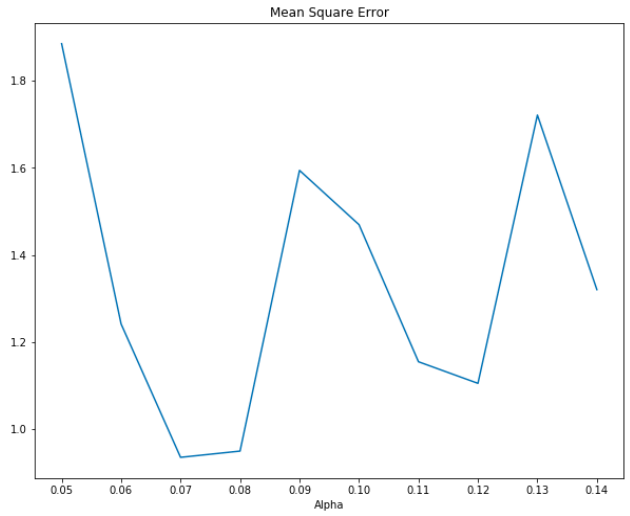
4.1. Regression

4.1.1. Baseline Model, Performance, and Evaluation

In baseline model (Multilinear Regression), we used root mean square error (RMSE) and cross validation method to evaluate models. This metric represents the sample standard deviation of the difference between predicted values and observed values (called residuals). Note that given the distribution of the target variables, we log-transformed the target variables thus the RMSE here represents the difference between logs of the predicted and observed values.

4.1.2. Model Selection and Parameter Tuning

- **The LASSO Regression:** LASSO stands for Least Absolute Shrinkage and Selection Operator. This is a regression analysis method that adds a regularize component to penalize the use of too many features, thus performing both variable selection by setting highly correlated features as 0 and reducing overfitting. By cross validation, the penalty parameter alpha = 0.07 provides model with the lowest mean square error. Note that we used Mean Square Error instead of Root Mean Square Error because for each alpha, we ran 5 folds of cross validation and got the average value. As a result, we want the alpha with the minimum value of mean square errors.



- **XGBoost:** XGBoost means extreme gradient boosting that ensembles gradient boosting, stochastic gradient boosting, and regularized gradient boosting to improve computational speed and predicting power. Given the limited computation power, we chose number of trees to be 100 with the learning rate of 0.08.

4.1.3. Results

As shown in Table 2. below, XGBoost model after applying PCA method yields the best result. This confirms the predicting power of XGBoost and the importance of dimensionality reduction in feature selection.

Model	Mean (RMSE)	Std Error (RMSE)
Baseline	1.0816	0.3512
Lasso (Alpha = 0.07)	0.8991	0.0064
Lasso (PCA, Alpha = 0.07)	0.9244	0.0083
XGBoost	0.8924	0.0789
XGBoost (PCA)	**0.8742**	**0.0039**

Table 2. Regression Result

4.2. Binary Classification

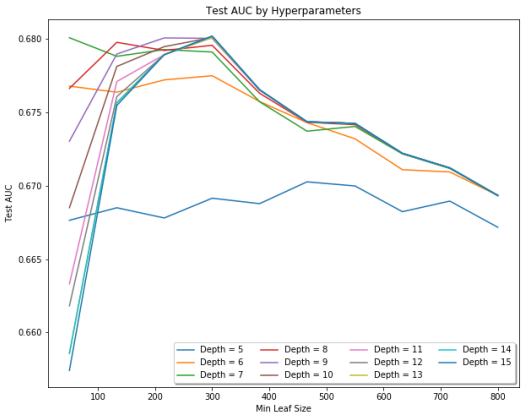
4.2.1. Baseline Model, Performance, and Evaluation

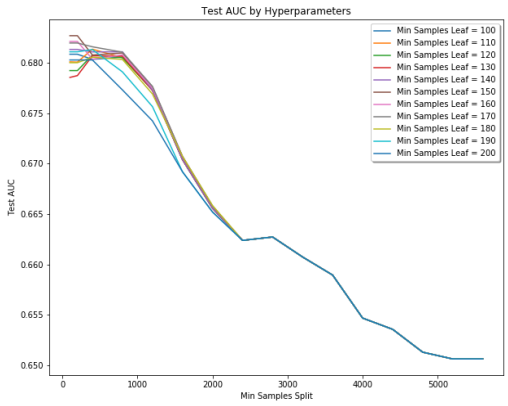
For our crude baseline binary classification model, we trained a Logistic Regression model with all the

default parameter values on the entire dataset.
We did not apply any feature engineering. The AUC is 0.6073.
Since our data is balanced, Accuracy could unbiasedly demonstrate the predicting power of models. In addition, AUC is a popular informative evaluation metric for classification models. Thus, we used both Accuracy score and AUC score to compare and improve our models.

4.2.2. Model Selection and Parameter Tuning

- **Logistic Regression:** Logistic Regression is a relatively well-behaved model for binary classification and it runs much faster than other models. It also outputs probability estimates that are useful for ranking. However, Logistic Regression might not perform well if the feature set is large. For our project, we used GridSearchCV (5 folds) for tuning two hyperparameters: C (inverse of regularization strength) and penalty (the norm used in the penalization) and found that the model achieves the best performance when C = 10,000 and penalty = l1.
- **Decision Tree:** Decision tree has very few assumptions for the data and it is not sensitive to outliers. Additionally, decision tree is easy to explain and interpret because it is closely related to human decision-making process. However, decision tree is relatively unstable and small changes in data might result in a completely different tree. A single decision tree tends to overfit the data as well. Therefore, in our project, we focused on tuning three hyperparameters: max_depth, min_samples_leaf, and min_samples_split. In order to significantly reduce the computation time for GridSearchCV to find the optimal set of hyperparameters, we first attempted to narrow down the range for the hyperparameters by visualizing the test set AUC for different combinations of hyperparameters. We can see from the plots that the best depth is 7, 8, or 9; the best minimum number of samples at a leaf node ranges from 100 to 200; the best minimum number of samples required to split an internal node ranges from 100 to 800. We then provided these targeted ranges to GridSearchCV (5 folds) and obtained the optimal set of hyperparameters: max_depth = 7, min_samples_leaf = 130, and min_samples_split = 600. Compared with the default Decision Tree, we found tuning hyperparameters significantly improves the performance of Decision Tree.

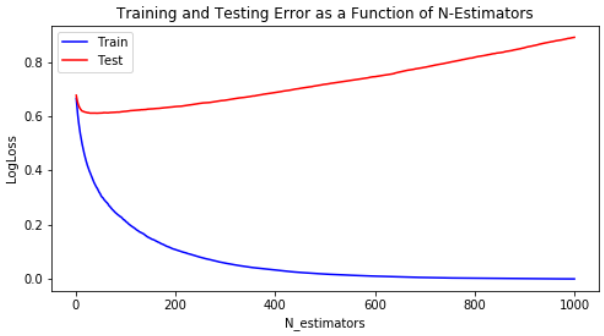




- **Random Forest:** Random Forest is an ensemble of Decision Trees. It controls overfitting and improves the prediction accuracy by fitting a number of Decision Trees on various sub-samples of the dataset and averaging the results from the Decision Tree predictions. The Random Forest we implemented inherits the optimal hyperparameter values of the tuned Decision Tree. However, there are two more hyperparameters specific to Random Forest: n_estimators and max_features. We then used GridSearchCV (5 folds) to find that the optimal number of trees in the Random Forest

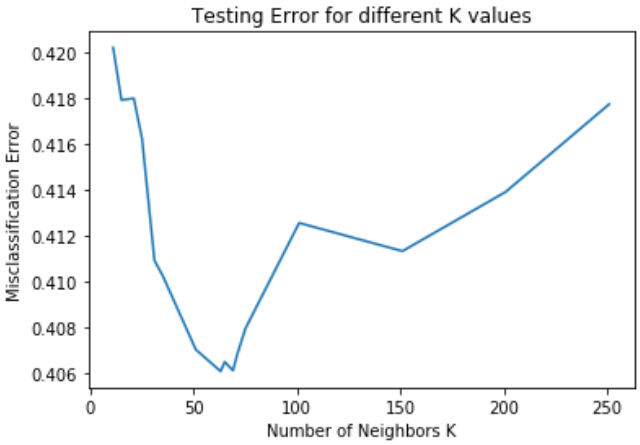
is 200 and the optimal number of features that will be randomly selected for each tree is 15.

- **AdaBoosting:** Adaboosting trains additional weighted classifier where misclassified instances are weighted more. We believe AdaBoosting is useful since the final decision boundary is the weighted sum of the decision boundaries of the weaker learners. It is crucial in learning more complex relations. Given that the learning rate is the hyperparameter deciding the weight assigned to correctly classified and incorrectly classified models, we believe that tuning learning_rate will help improve the performance of the classifier.
- **Gradient Boosting:** Gradient Boosting trains the new tree over the residual errors of the previous trees following a gradient descent optimization process. Gradient Boosting is more robust compared with Random Forest when few features dominates the predictions. We again used the optimal Decision Tree and proceeded to tune the rest hyperparameters, especially n_estimators. As shown by the plot, it is apparent that the model faces a greater possibility of overfitting as the number of estimators increases, which also creates a waste of computation powers in reality. The optimal n_estmators is 250, learning_rate = 0.1



- **XGBoosting:** We decided to experiment with XGBoosting as it follows the same principle of Gradient Boosting, but a more regularized model formalization to control overfitting, hence it usually generates better results than Gradient Boosting. However, we noticed the computation is more costly compared with Gradient Boosting. We again tuned the hyperparameters of n_estimators and learning_rate. Due to the limitation of computational power, we limited our trees to 250. However, we further tested with a larger number of estimators, which only gave a slight increase in test accuracy: 0.01. Hence we believe it is reasonable to adopt 250 as the optimal n_estimators, and learning rate = 0.3.

- **Nearest Neighbor Classifier:** The numeric target variable 'shares' was converted into 2 classes (similar to the baseline model.) Different values of k were tried - as can be seen from the graph on the right. Best performance was achieved with k = 63 (accuracy of 59%).



- **Gaussian Naive Bayes:** This model is amongst the basic models that can be implemented. The accuracy achieved was 50.5%.

4.2.3. Results

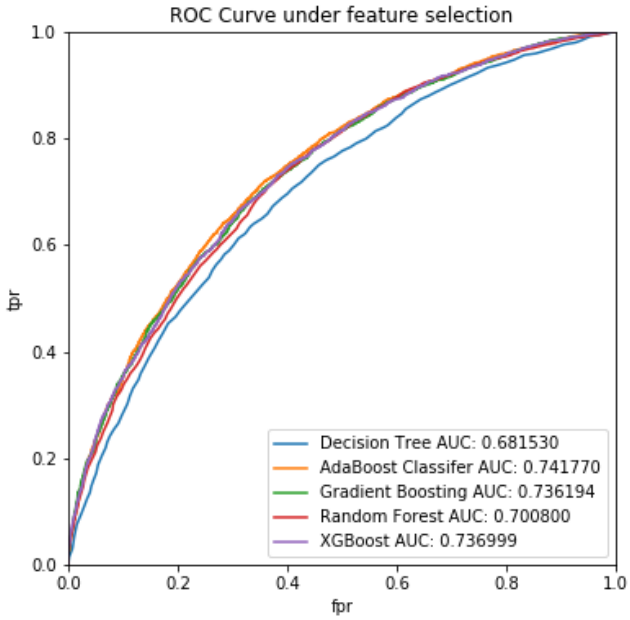
As shown by the result table(Table 3.), it is apparent that both metrics, test accuracy and AUC, are improved after adopting more complex models, hyper parameter tuning and cross validations. The ensemble methods are better at capturing complex relation between features and targets, hence it is reasonable for ensemble methods to perform better. The increase in n_estimators of the ensemble methods bring in better performances than default settings, as the more trees the model grows, the better the model will be at

capturing relations and train 'hard-to-fit' data samples.

Model	Acc	AUC	Acc (tuned)	AUC (tuned)
Baseline		0.6073		
LogReg	0.6203	0.6643	0.6335	0.6828
DeciTree	0.5717	0.5717	0.6326	0.6815
RanForest	0.6141	0.6633	0.6459	0.7008
GDBoost	0.6737	0.7359	0.6750	0.7362
AdaBoost	0.5908	0.6264	**0.6802**	**0.7418**
XGBoost	0.6771	0.7423	0.6759	0.7370
KNN	0.5798	0.6108	0.5939	0.6265
Naive Bayes	0.5056	0.5335	-	-

Table 3. Binary Classification Result

The resulting improvement in performances compared with the decision tree by AUC score is plotted below, where the best performed model is AdaBoosting, with n_estimators = 250, learning_rate = 0.03, max_depth = 7, min_samples_split = 600, min_samples_leaf = 130. Theoretically, AdaBoosting has a good learning guarantee. It is especially good at predicting balanced dataset, which in this case the shares are evenly separated into 2 classes by a median threshold. In practical use scenario, we believe for binary classification, it is reasonable to separate articles into two balanced categories to get a general prediction of its popularity.



Moreover, we compared prediction accuracy and AUC with feature selection and with not. The result indicates by
conducting feature selection, it does jeopardize model performance, but rather limited. It hence prompts us that the key features list is able to predict target on the same level with the entire feature list, with less computational power needed. It is also valuable in business as it is impractical for authors/publishers to pay attention to over 50 features, while a a abridged feature list will be in higher demand.

Algorithms	TestAcc (tuned)	AUC (tuned)	TestAcc (feat)	AUC (feat)
AdaBoost	0.7473	0.7456	0.6802	0.7418
GraBoost	0.7419	0.7413	0.6750	0.7362
XGBoost	0.6844	0.7433	0.6759	0.7370

Table 4. Feature Selection Effects On Classifiers

4.3. Multi-class Classification

We believe it is worthwhile to carry out multi-class classifications to offer more space for authors/editors to make decisions, for example, to cater different publishing times more specifically and take more detailed measures to help increase revenues.

4.3.1. Threshold Selection

We divide the data into three groups using different percentile thresholds:

- **33 and 66 percentile:** We divide the shares data at 33 and 66 percentile. The thresholds of shares number are threshold_1 = 1100, threshold_2 = 2100, and the size of resulting class is roughly the same: 14732, 11957 and 12955.
- **25 and 75 percentile:** Given a long tail of number of shares, this cut off at 25 percentile and 75 percentile can segregate articles by popularity, based on the fact that only a small number of articles are truly popular. The resulting threshold_1 = 946, threshold_2 = 2800, and the size of the resulting class is: 9930, 20084, 9630.
- **Standard Deviation:** This method is inspired by left skewed distribution of the target variable. A natural log transformation presents the distribution of shares data a close approximation of a normal distribution.We take thresholds at one standard deviation away from the mean of the log transformed shares data. The thresholds are threshold_1 = 695, threshodl_2 = 4470. The resulting class size is: 3693, 30092, 5859.

4.3.2. Model Selection and Parameter Tuning

Given the limited computation powers, we inherit the best model configuration from the previous binary classification: AdaBoost with n_estimators 250, and learning_rate 0.03. The evaluation metric is the AUC score. It is apparent that the split methods of splitting the target variable over the standard deviation away from the mean works the best given it has best AUC score, both before feature selection and after. It is apparent from the above table that the shrinking of feature sizes show no significant influence over the test accuracy of the model. Although the shrinking of feature size does decrease the accuracy, as shown in even split and quarter split, we conclude the decrease insubstantial. However, there still exists limitations in our approaches. We cannot rule out the possibility that adaBoosting is not the best fit for this multi-class scenario. It is also possible that the models need specific tuning of the parameters for different split methods.

Split Method	AUC (orig features)	AUC (selected features)
Even split	0.7610	0.6987
Quarter split	0.7364	0.7290
Normal split	0.8599	0.8568

Table 5. Multi-Class Classification Result

5. Deployment

Regarding the business deployment scenarios, we can develop a Decision Support System based on our binary classification and multi-class classification models. The Decision Support System first extracts the features regarding various aspects of an article as described above and predicts its popularity prior to the publication. In this way, the Decision Support System could help publishers and editors distribute resources accordingly to maximize profits from advertisements. More importantly, since the Decision Support System could identify the features that greatly contribute to the popularity of an article, publishers and editors could also revise the article adequately to maximize the number of shares for the article, given specific goals and targeted audience. One of the challenges is that we should keep updating the Decision Support System by providing most up-to-date articles so that the system could be fully aware of the current trends and characteristics of articles.

Regarding ethical issues, editors and publishers may use this tool to develop popular articles for propaganda and serve particular political purposes, while manipulating the public's freedom access to diverse news and opinions.

6. Conclusion & Future Work

By visualizing the correlation plots and exploring several feature selection methods, we found that the keyword based features and the Natural Language Processing features are most effective in predicting the popularity of an article. Regarding modeling, AdaBoosting achieves the highest accuracy and AUC and the other tree-based ensemble methods perform similarly well.

Given the time constraints, we used the best model for binary classification to solve the multi-class classification problem. For future work, we would like to rigorously select models, features, and tune hyperparameters to classify articles into 3 groups. Besides modeling for multi-class classification, we would also like to incorporate more features in our models to improve prediction accuracy. The current feature set describes various aspects of an article, but it does not include any content information of the article. In the future, we could consider the individual words and phrases in an article as additional features so that the models could take the content of an article into consideration as well. In addition, we would also like to collect more up-to-date articles as well as articles from a variety of sources to implement a more accurate, state-of-the-art, and comprehensive model while reducing selection bias.

Appendix

1. Project code can be found at Github.
2. "UCI Machine Learning Repository: Online News Popularity Data Set". 2015. *Archive.Ics.Uci.Edu*. https://archive.ics.uci.edu/ml/datasets/online+news+popularity.
3. "AARSHAY JAIN. 2016. "Complete Guide To Parameter Tuning In Xgboost (With Codes In Python)", *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/.
4. "Brems, Matt. 2017. "A One-Stop Shop For Principal Component Analysis – Towards Data Science". *Towards Data Science*. https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c.
5. "Zakka, Kevin. 2016. "A Complete Guide To K-Nearest-Neighbors With Applications In Python And R". *Kevinzakka.Github.Io*. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/.
6. 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.
7. Feature List by greedy search:

   {' kw_avg_max', ' LDA_00',' LDA_01', ' global_subjectivity',' month_is_jan', ' kw_avg_avg',' data_channel_is_socmed', ' num_videos',' LDA_02',' timedelta',' data_channel_is_lifestyle',' n_unique_tokens',' is_weekend',' LDA_03',' kw_min_avg',' LDA_04',' min_positive_polarity" data_channel_is_entertainment',' month_is_oct',' self_reference_avg_sharess',' self_reference_min_shares',' num_hrefs'}

8. Feature list by decision tree feature ranking alone (30 features):

   {'kw_avg_avg', 'kw_max_avg', 'kw_avg_max', 'average_token_length','global_sentiment_polarity', 'global_subjectivity', 'LDA_01', 'LDA_02','self_reference_min_shares', 'n_unique_tokens','global_rate_positive_words', 'n_non_stop_unique_tokens', 'kw_avg_min','avg_positive_polarity', 'LDA_00', 'n_tokens_content','avg_negative_polarity', 'LDA_04', 'global_rate_negative_words','LDA_03', 'kw_max_min', 'num_hrefs', 'kw_min_avg','self_reference_avg_sharess', 'self_reference_max_shares','n_tokens_title', 'abs_title_subjectivity', 'kw_min_max','title_sentiment_polarity', 'num_imgs'}

9. Feature list by several tree-based models feature rankings (15 features):

   {'kw_max_avg','kw_avg_avg','LDA_02','self_reference_min_shares','is_weekend', 'kw_min_avg',' data_channel_is_world','LDA_04','average_token_length','global_subjectivity','global_rate_positive_words','n_non_stop_unique_token','n_unique_token',' global_sentiment_polarity',' LDA_00' }

10. Feature list by recursive feature eliminations:(Random Forest):

    {'timedelta','n_tokens_content','n_unique_tokens','n_non_stop_unique_tokens','average_token_length','kw_avg_min','kw_avg_max','kw_max_avg',' kw_avg_avg','self_reference_min_shares','self_reference_avg_sharess','LDA_00','LDA_01','LDA_02',' LDA_03',' LDA_04',' global_subjectivity',' global_sentiment_polarity','global_rate_positive_words',' avg_positive_polarity' }

11. Feature list by recursive feature eliminations:(Logistic Regression, 30 features):

    {'LDA_00', 'LDA_01', 'LDA_02', 'LDA_03', 'LDA_04','abs_title_subjectivity',

'average_token_length',
'data_channel_is_bus','data_channel_is_entertainment','data_channel_is_lifestyle','data_channel_is_socmed', 'data_channel_is_world',
'global_rate_negative_words',
'global_sentiment_polarity','global_subjectivity', 'is_weekend',
'min_positive_polarity','n_non_stop_unique_tokens', 'n_non_stop_words',
'n_unique_tokens', 'rate_negative_words',
'rate_positive_words','title_sentiment_polarity', 'title_subjectivity',
'weekday_is_friday','weekday_is_saturday',
'weekday_is_sunday',
'weekday_is_thursday',
'weekday_is_tuesday',
'weekday_is_wednesday'}

Contributions

In order to achieve such an ambitious yet achievable analysis, we separated the work between all of us in the following way:

- Nhung Le (nhl256): Developing baseline models for regression and classification models; Setting up machine learning framework; Investigating and applying Principal Component Analysis method; Developing and tuning regressing models.
- Shirley Xu (xx852): Investigating distribution of variables and correlation; Exploring feature selection methods; Developing and tuning logistic regression, decision tree, and random forest models for binary classification.
- Hengyu Tang (ht1162): Cleaning data; Exploring feature selection methods; Developing xgboost, adaboost, and gradient boosting models for binary classification; Developing and tuning models for multi-class classification.
- Shradha Taneja (st3277): Implementation of kNN and Gaussian Naive Bayes algorithms; Data exploration and visualization using Tableau (available in Github Repo).