Prediction of Online News Popularity

Final Report for DATA1030 Fall 2021 at Brown University

Yash Mehta, Data Science Initiative

Supervised by Dr. Andras Zsom

Project on GitHub

1. Introduction

Over the past couple of decades, with online platforms increasingly winning out over physical newspapers, media organizations are increasingly relying on analytics and machine learning to understand their reader base, moderate interactions and analyze which content is more likely to generate traffic. A study conducted by the Pew Research Center found that the average number of unique monthly visitors to the top 50 US newspaper websites increased from 8.2 million at the end of 2014 to 13.9 million at the end of 2020 [3]. The pandemic has accelerated this trend, with traditional advertising revenues falling and platforms relying on subscriptions and digital advertising to generate revenues.

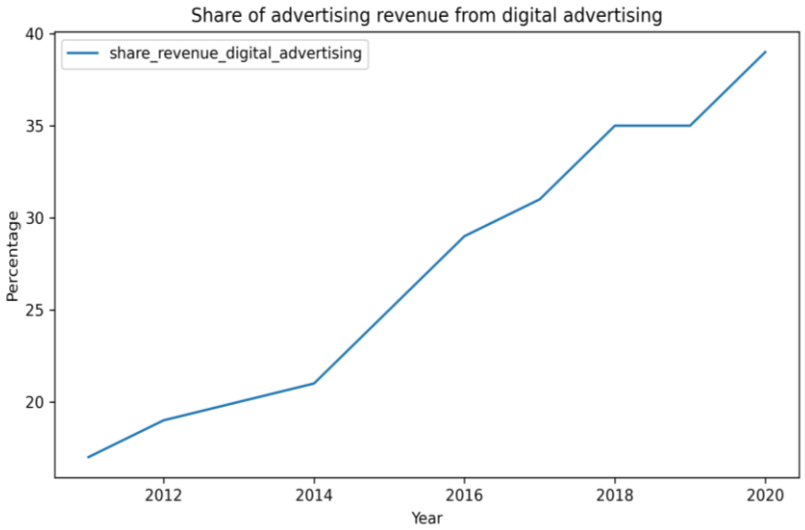

Figure 1: Share of advertising revenue from digital advertising. Data from Pew Research Center [3]

In this context, predicting what content will resonate with readers is important for media organizations, activists and political organizations. Such analysis yields low prediction scores but can be useful as a signal to a content provider if it can indicate where to focus efforts in order to optimize content, or to a platform that publishes such content so that it can focus its moderation efforts.

This project explores the performance of various machine learning models in predicting the likelihood of a news article becoming popular as measured by shares on social media. The paper published by Fernandes et al. [1] covering articles published on the Mashable platform from January 7, 2013, to January 7, 2015 is used as the primary reference. The authors focused on predicting article popularity before publication, and further work by Zhang et. al. [2] expands upon this by using PCA for feature selection, but their thresholds and classification metrics are not comparable to the original paper.

2. Exploratory Data Analysis

The data contain 39,644 records, each corresponding to a unique article. Each record has 61 attributes - 2 of which are non-predictive (url, timedelta), and 1 of which is the target variable (no. of shares). While the number of shares is a continuous variable, in this project a threshold is set, above which an article is labelled 'popular', thus converting the task into a classification problem.

Feature Type	Sample Features
Non-descriptive	url, Time since publication
Content	Topic of article, # words/unique words/word length in title/content Keyword shares
Media	# of images/videos
Connections	# of links, # of links to other articles on Mashable, Popularity of referenced artcles
Timeline	Day of week/Weekend
NLP features	LDA topic closeness, Polarity/subjectivity of content/title
Target variable	**Number of social media shares**

Figure 2: Sample of features from each feature type

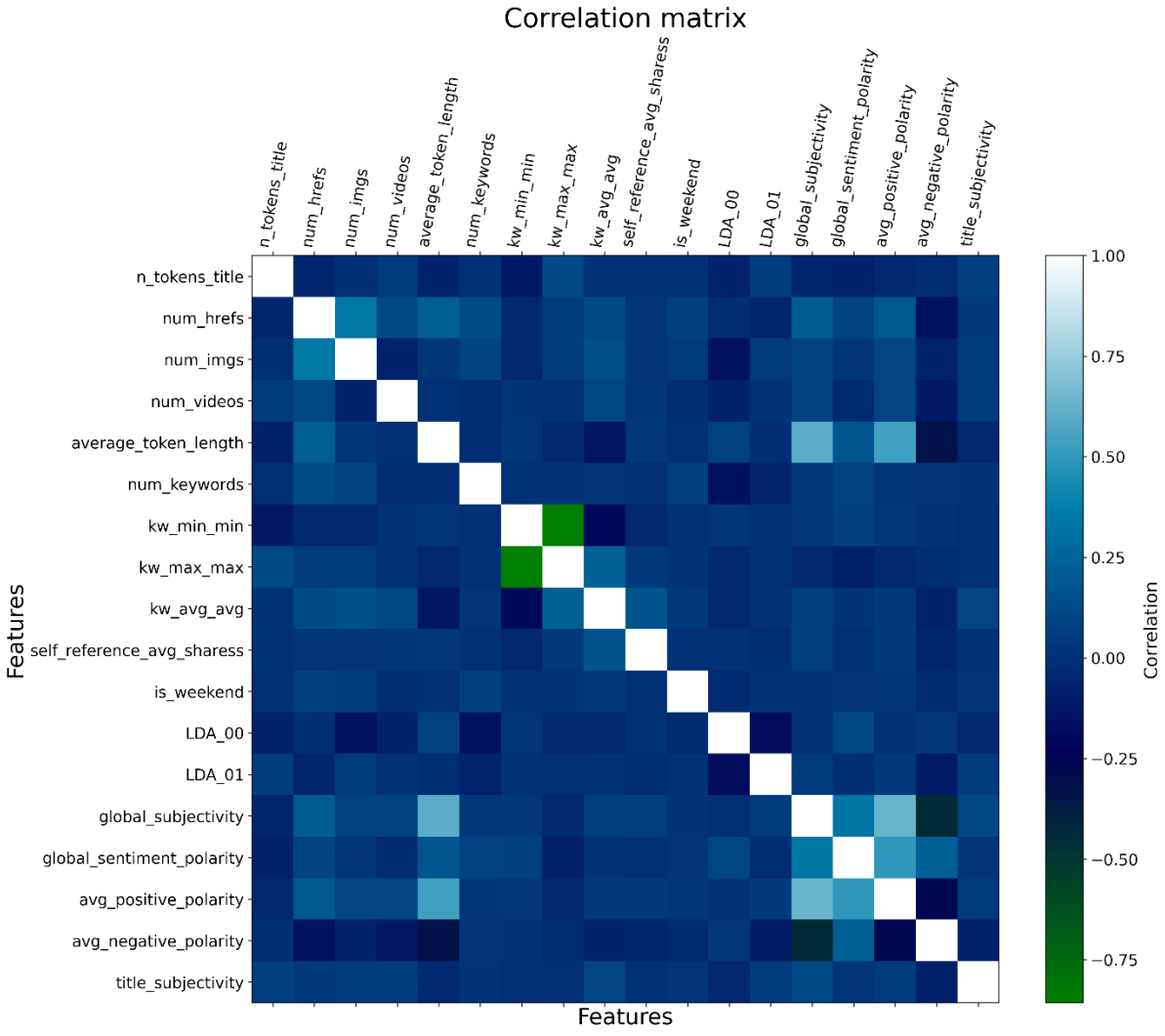

Figure 3: Correlation matrix of selected features colored by magnitude of correlation

The correlation matrix of the features reveals generally low correlations between most features with some exceptions (global_subjectivity and avg_positive_polarity with average_token_length). However, most of the features have extremely long tails and are loosely associated with the target variable.

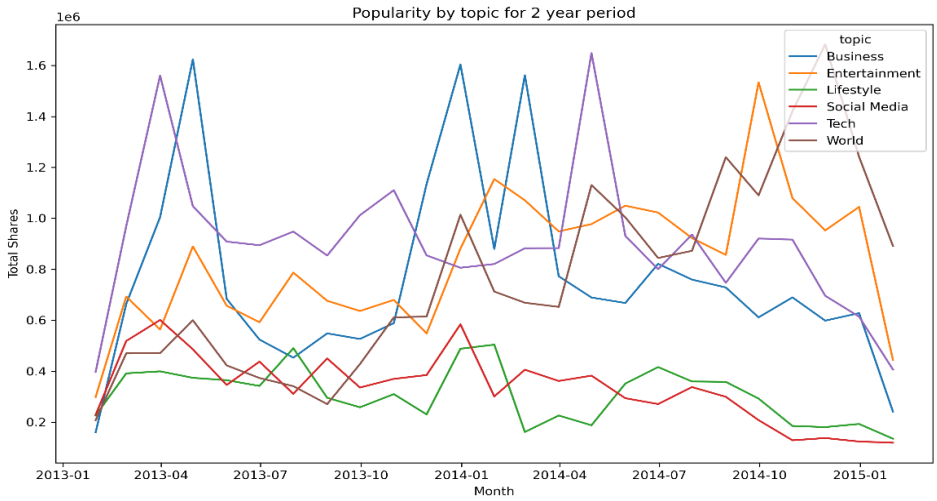

Figure 4: Timeline of topic popularity over each month in the 2-year period

The chart above shows differences in how audiences responded to different topics over the period that the data was collected. Articles related to the 'World' topic were the most shared whereas articles related to 'Lifestyle' and 'Social Media' were among the least shared. It also reflects the general growth trend in the popularity of Mashable articles, although the popularity increase is not proportionally shared across topics.

A major consideration is to convert the target variable from a continuous variable into a target variable by setting a threshold value. The target variable has a long-tailed distribution as highlighted below:

Mean	Standard Deviation	Minimum	25th percentile	Median	75th percentile	Maximum
3,395.38	11,626.95	1.00	946.00	1,400.00	2,800.00	843,300.00

Figure 5: Mean, standard deviation and various percentiles of the target variable

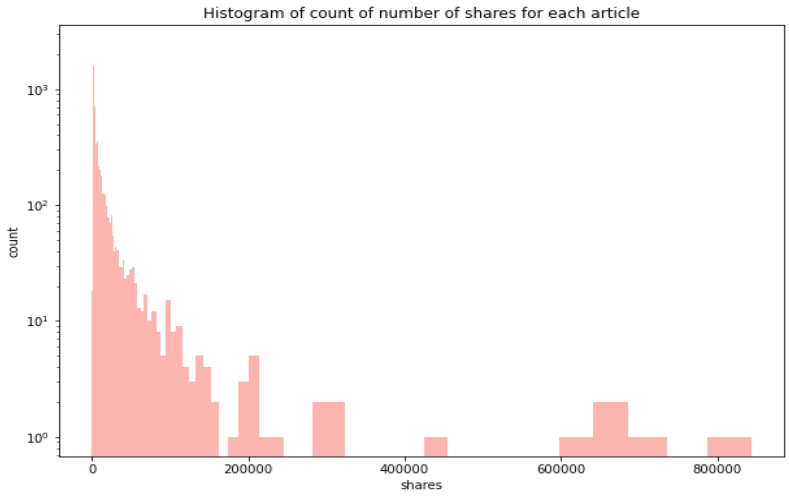

Figure 6: Long-tailed distribution of target: a small proportion of articles get a very large number of shares

Converting the problem to a classification problem helps avoid having to make predictions in the long tail where data points are very thinly scattered. A threshold value of 1,400 has been set for the number of shares above which an article is considered to be popular. The threshold has been selected because it creates a balanced split between popular and unpopular articles, and because it facilitates benchmarking against findings from literature.

3. Methods

3.1 Data Splitting & Preprocessing

The dataset consists of records of online articles published by Mashable. Each row corresponds to a unique article and there are a total of 39,644 articles. The target variable is 'popular' - a binary variable where 0 corresponds to not-popular and 1 corresponds to popular. This variable is derived from the 'shares' variable in the original dataset, which is the number of shares for a given article, and we set a threshold = 1400, above which an article is considered popular.

The dataset was generated in [1] by collecting information about all articles published on Mashable – no group structure was induced by article selection or linkages between articles that is mentioned in the original paper. There is nominally a time-dimension in the dataset, however there is no time series structure in the dataset since that feature is non predictive and the other feature variables are not dependent in any way upon a time-dependent state of information. This is confirmed by a correlation of 0.0087 between the timedelta variable and the target. Therefore, the data can be considered to be independent and identically distributed (IID). With this in mind, a holdout set of 10% to review model performance on unseen data is removed and kept aside. Then a basic split of 80:10:10 has been used on the remaining data, resulting in 28543, 3568 and 3568 articles for the train, validation and test sets respectively. 8-fold cross validation has been used to optimize the model hyperparameters.

On the continuous features, the MinMaxEncoder is used for features relating to polarity, Latent Dirichlet Allocation (LDA), no. of tokens in title, average token length and number of keywords since these features are bounded and do not exhibit outliers. The remaining features are not bounded and therefore are encoded with a StandardScaler. On the categorical features (is_weekend, day_of_week and topic), a OneHotEncoder has been used since the features do not exhibit inherent order. Eg. a OneHot encoder has been used on the topic feature. After preprocessing, there are 60 features in the data and there are no missing values.

3.2 Model Selection & Hyperparameter Tuning

The problem is of binary classification with evenly balanced classes with 19,562 popular articles and 20,082 unpopular articles in the data. An F-beta score with a beta value of 1.5 was selected as the evaluation metric so that higher weight is given to precision than recall. The reasoning is that a small increase in false positives (unpopular articles flagged as popular) is preferable to the reverse case while also not going too far and flagging a large majority as positive if we had a beta value of 2 or higher.

Using the methodology described above, 4 machine learning models have been trained and compared: logistic regression model (without regularization, with L1 regularization, with L2 regularization considered separately), random forest classifier, K-nearest neighbors classifier, and an XGBoost classifier. Hyperparameters were tuned using a grid search method for each model to find the optimal parameter combination for each model. This process was repeated on 10 different random states for 8 different splits. Below are the parameters values tried for each model:

Model	Parameters	Mean	Std. Dev
Lasso Regression	**C**: 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3	C: 1.09	C: 2.97
Ridge Regression	**C**: 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3	C: 302	C: 457
RandomForest	**max_features**: 0.5, 0.75, None; **max_depth**: 5, 25, 100	max_features: 0.54 max_depth: 25	max_features: 0.088 max_depth: 0
KNN	**n_neighbors**: 1, 3, 10, 30, 100; **weights**: uniform, distance	n_neighbors: 3	n_neighbors: 0
XGBoost	**n_estimators**: 50, 100, 500, 1000, 2000; **max_depth**: 1,3,10,30,100,	n_estimators: 800 max_depth: 3	n_estimators: 245 max_depth: 0

Figure 7: Hyperparameters tested for each model. Mean & Std. dev of parameters of best models over 10 random states

After tuning, each grid search's best model parameters were extracted and used for comparison on accuracy score on a holdout set against a baseline model which predicts all points as belonging to the most frequent class. Below are the average F-beta(1.5) scores for the best of each model across the ten random states along with their standard deviations.

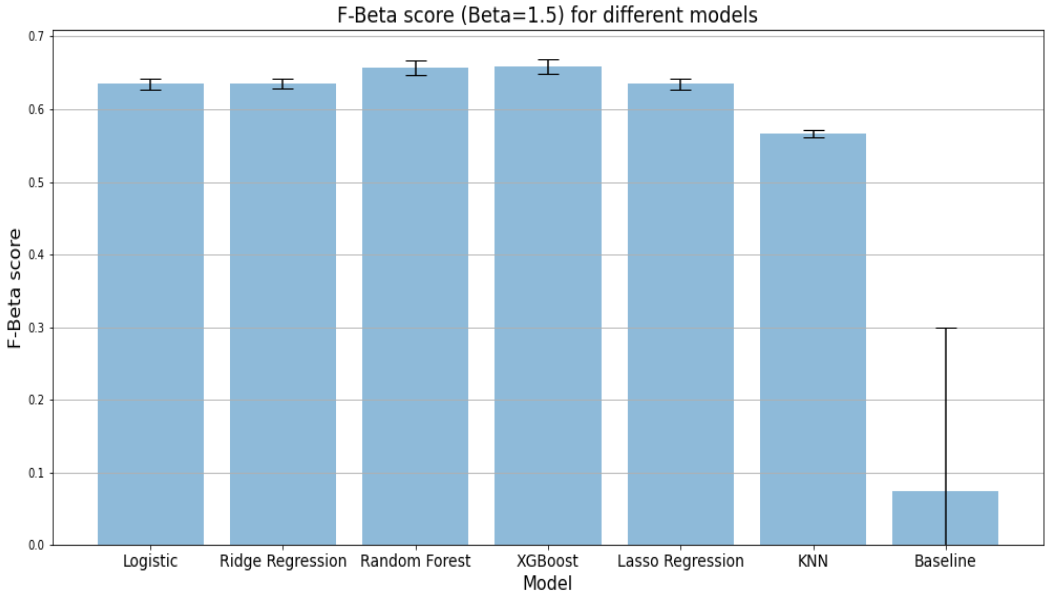

Figure 8: F-beta scores for the best models over ten random states

The XGBoost model had the best performance over the ten random states and was selected as the model of choice for the dataset. In order to select the best hyperparameter combination, each of the proposed best hyperparameter combinations were tested on a separate holdout set over 5 random states. Finally, the best hyperparameter combination was selected for the XGBoost Classifier:

Parameter	Value
n_estimators	1000
max_depth	3
learning_rate	0.05
colsample_bytree	0.9
subsample	0.66

Figure 9: Best hyperparameter combination for the fitted XGBoost classifier

Given the non-deterministic nature of XGBoost, the fitted model was fitted over 10 random states to check the standard deviation in the F-beta score on test, and the model performed consistently well with a mean score of 0.6627 standard deviation of 1.06%

4. Results

4.1 Model evaluation

After choosing the best model and hyperparameter combination, the model was retrained on new splits over 5 new random states. For each split, 80% of the data was allocated to training and 20% was allocated to testing. For each random state, the model was fit to training and then predictions on the test set were recorded to analyze performance. The performance was compared to a baseline model (averaged across 5 random states) which predicted all data points as belonging to the most frequent class over each split for each random state.

The baseline model returned an average F beta score of 0.0747 with a standard deviation of 0.2241 as compared to the fitted XGBoost model which returned an average F beta score of 0.6622 with a standard deviation of 0.0059. The fitted model performed significantly better than the baseline model, exceeding its average performance by 2.6 standard deviations.

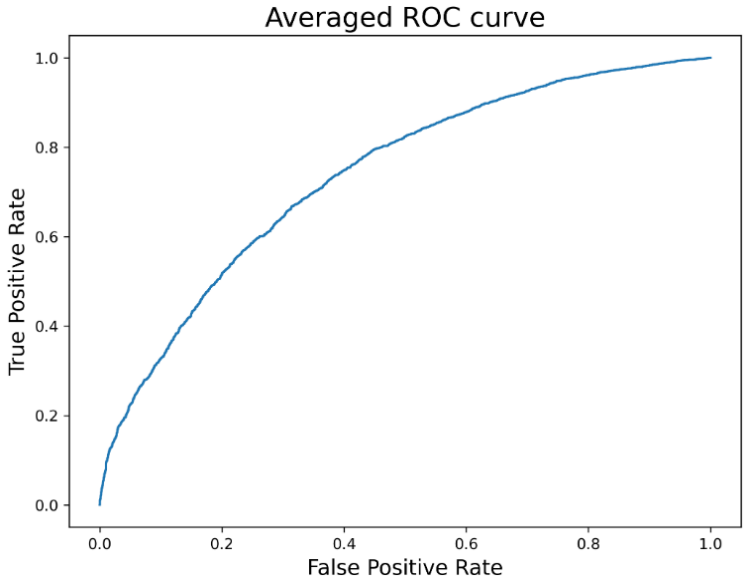

Figure 10: ROC curve of fitted XGBoost model averaged over 5 random states

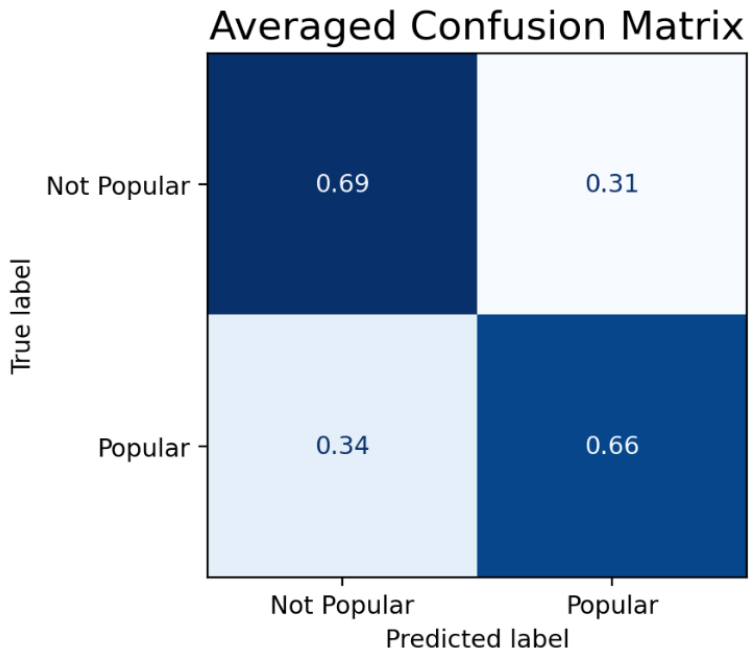

Figure 11: Confusion matrix of fitted XGBoost model averaged over 5 random states

4.2 Feature importance

To facilitate interpretation of the fitted model, three types of feature importances have been plotted – permutation feature importance (how much the model performance deteriorates when a single feature is shuffled randomly), and the weight and total gain scores from the XGBoost model (feature importance as the number of times a feature is used to split the data across all trees and the total gain across all splits the feature is used in respectively). For ease of visualization, only top 10 features for each importance type have been selected.

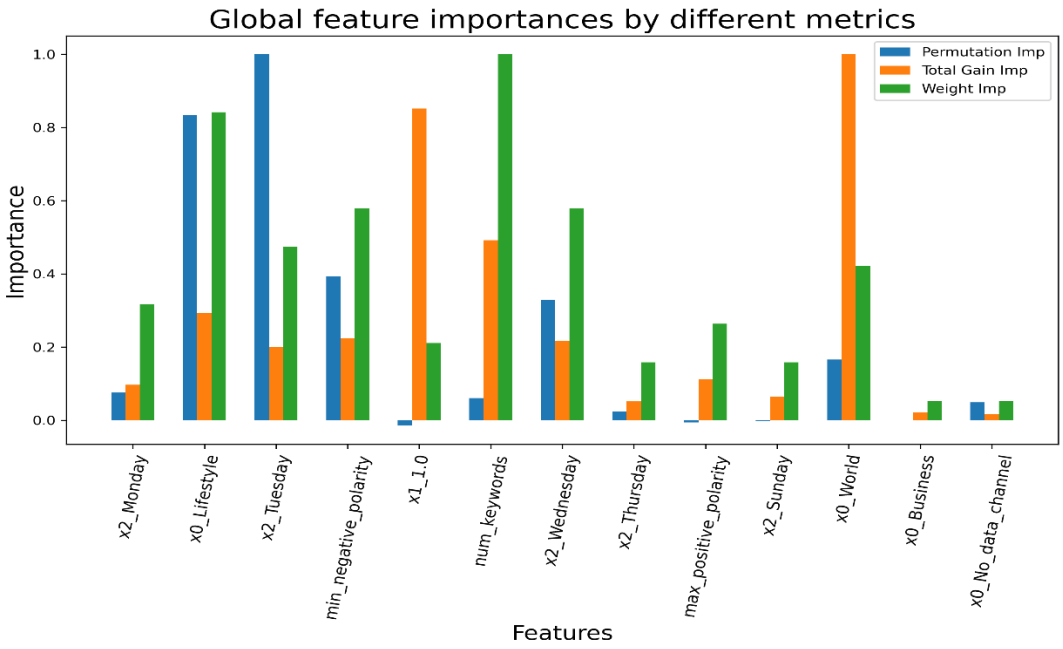

Figure 12: Feature importances by permutation, total gain and weight (top 10 features for each type, normalized)

We find that measured feature importances vary significantly across each metric. SHAP scores offer a more robust way of measuring feature importance.

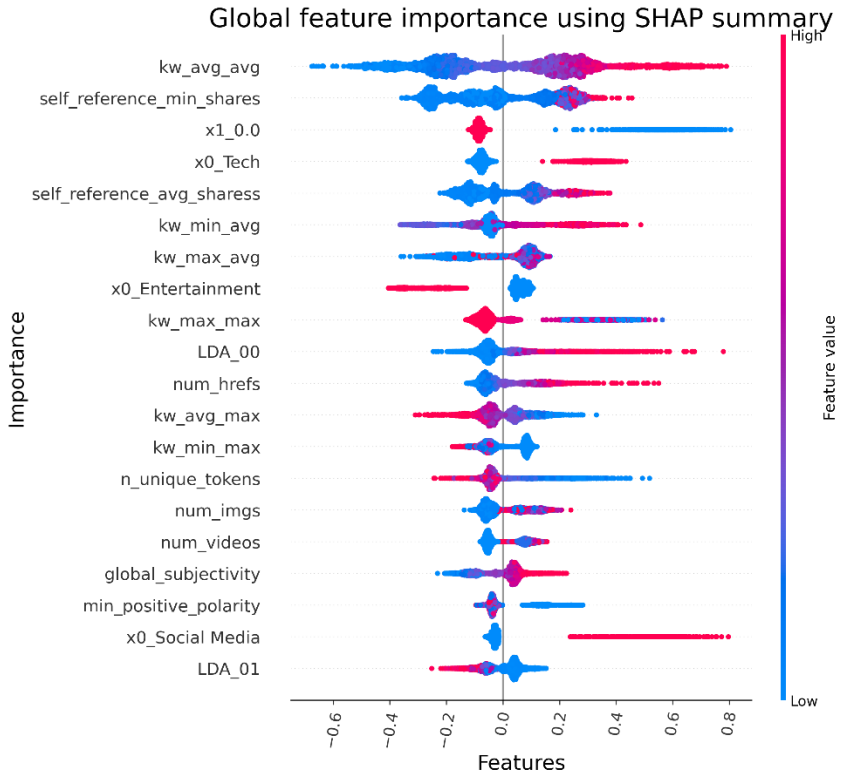

Figure 13: Global feature importances using SHAP summary

Overall, the SHAP scores indicate that the most important features are the ones corresponding to keywords and references to popular articles, which fits in with a general understanding of how content is optimized for digitial discovery. However, some features like x0_Social Media are very important to a subset of articles. To understand how the global importances manifest at locally, SHAP values were observed for 2 specific observations, one for unpopular article and one for a popular one.

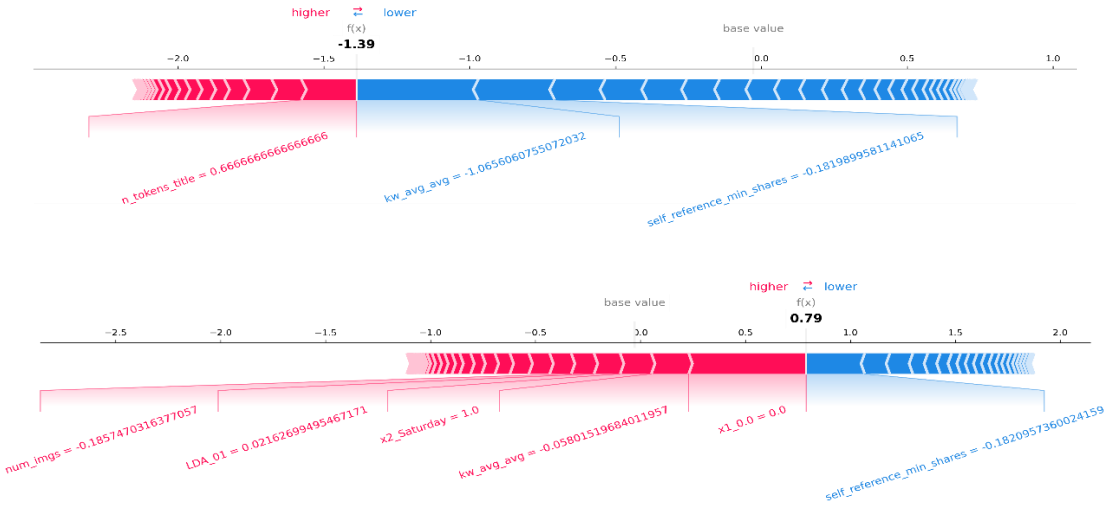

Figure 14: Local feature importances by SHAP scores

The influence of the features identified as most important globally is visible in the classification of these observations, however, for the first observation, the n_tokens_title which is not a significant feature globally plays an outsize role.

5. Outlook

From this analysis of the data, we see that no algorithm has hit an F-beta score of 0.7. Tree based methods have performed better than linear methods and the KNN non-parametric method showed the poorest performance, results which are consistent with the literature. Given their success with text data, deep neural networks may show better performance and should be the next step in the analysis.

One of the issues is that the available features are not the raw articles but derived metrics like keyword counts and LDA scores. While these reflect some information contained in the articles, they may leave out other useful information. A more robust method may be to pull the text from the articles and create new features or use the words in the articles as additional features to fit a more complex algorithm to the dataset so that the actual content of the articles is considered. This approach is that may also lend itself to an easier interpretation of the resulting models.

6. References

[1] Kelwin Fernandes, Pedro Vinagre, and Paulo 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.

[2] He Ren, Quan Yang, Stanford University, *"Predicting and Evaluating the Popularity of Online News"*, Machine Learning Project Work Report, 2015, pp. 1-5.

[3] Michael Barthel, Kirsten Worden Newspapers Fact Sheet, Pew Research Center
https://www.pewresearch.org/journalism/fact-sheet/newspapers/

[4] Kelwin Fernandes, Pedro Vinagre, Paulo Cortez, and Pedro Sernadela (2015). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

GitHub repository: https://github.com/spacegoat1/news_article_popularity