Data1030 Project: Online News Popularity Prediction

Qinmiao Deng
Department of Data Science Initiative
https://github.com/qmdeng/Data1030-Project.git

I. Introduction

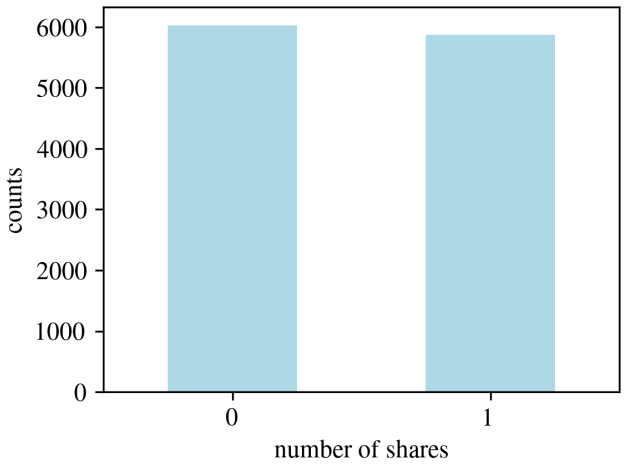
With the development of the internet, online news articles have become increasingly accessible to people worldwide, and people have developed the habit of reading and sharing online news with others through multiple digital media. Thus, it would be greatly valuable if we could predict the popularity of news for online news stakeholders such as advertisers or content providers. Embedded with this intention, we intend to utilize the dataset provided by the UCI machine learning repository, which collected 39644 articles published in a period of two years from the Mashable website and extracts 61 attributes, describing different aspects of each article. This dataset is originally acquired and preprocessed by K. Fernandes et al.

In this project, we aim to predict the number of shares for each online news. Based on research on previous projects on this dataset and due to the high variance of the target variable (number of shares), it is not feasible to directly apply linear regression, specifically, only 18% of prediction values (number of shares) are within 1000 of the actual results. In order to better predict and make our result more referential, we discretized the target value to binary classes, class zero as the group with values below the median and class one as the group with values above the media and consider our problem as a binary classification problem. Our features have covered a wide range of aspects. Some attributes measure the words in articles such as the number of words of content/title, average word length, etc. Others analyze the number of article links and images or videos. Also, the features have mentioned the publication time, like posting on Monday or Tuesday. The rest of the attributes demonstrate the information about keywords, like article category, best or worst keywords, and provide knowledge in an NLP dimension, such as the article's closeness to five different topics, its absolute subjectivity and polarity level, etc.

Most of the online public reports apply machine learning to this dataset to solve a binary classification problem, to determine whether an online news article is popular or not. They deployed multiple ML models such as Logistic Regression, RF, SVM, and KNN. For evaluation metrics, they normally used f-score, AUC and accuracy. One project posted on Kaggle reached 62% accuracy and the other report posted on GitHub achieved 64% accuracy, both using Random Forest.

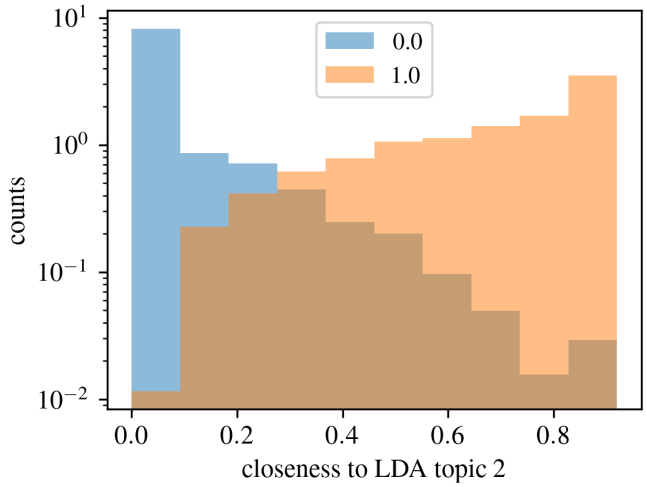
II. Exploratory Data Analysis

The dataset contains 61 attributes, including 1 target attribute, 1 non-predictive feature (URL of the article), and 59 predictive features. For the target variable, we have two classes, where class '0' indicates the group with the number of shares equal to or below the median (<1400) and class '1' demonstrates the group with the rest of the value above the median (>1400). The distribution of each group is quite balanced, shown in the graph below.



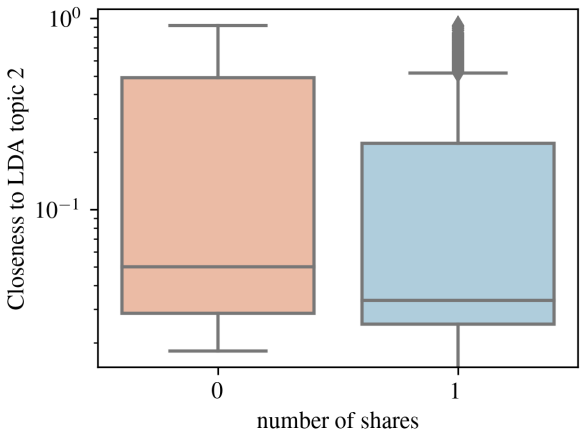
This figure shows a distribution of target variables

After navigating the correlation by plotting the correlation graph, I observed that there are three features, which are features “n_unique_tokens”, “n_non_stop_words”, and “n_non_stop_unique_tokens” that have extremely strong correlations (coefficient larger than 0.98) with each other. Therefore, I managed to drop two of them and just ekkp one (“n_unique_tokens”) as one of our features. Furthermore, some features have a strong correlation with each other, which demonstrates more information about our attributes. For example, feature LDA topic 2 is strongly related to the world channel, where LDA stands for a Topic Modeling technique to classify texts into a particular topic. The positive correlation implies that the topic 2 category mainly indicates the articles written about world news.



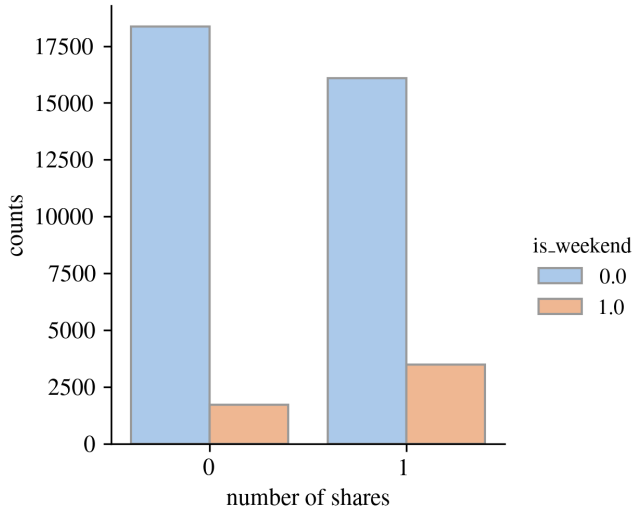
This figure shows a correlation between world channel and closeness to LDA topic 2

Additionally, I noticed that none of the features has a strong correlation with our target and the strongest coefficient with our target variable is about 0.16. Here I plotted two of the top five strongest correlation graphs with target variables, which I found interesting. The first graph shows a negative correlation between the number of shares and Closeness to LDA topic 2. The negative correlation demonstrates that people are less likely to share online news correlated to this sort of topic.



This figure shows a correlation between number of shares and closeness to LDA topic 2

This next graph depicts a positive correlation between weekends/weekdays publish time and our target variable. The positive correlation shows that people intend to share online news which is posted over the weekends.



This figure shows a correlation between number of shares and weekend

III. Methods

From my EDA, I observed my data does not have a group structure, and the dataset is independently and identically distributed, which is an iid data. Additionally, it does not have any time-dependent data, so it is not time series. Also, there is no missing data in the dataset for us to deal with. Before splitting and transforming data, the 'URL' column is dropped, since it is a non-predictive feature. Due to the large amount of our dataset and the extremely long time to run models with all the datapoint used, we applied stratified sampling to extract thirty percent of our data points to commit to further process, which are in total 11894 data points.

For model deployment, I developed a function that takes the unprocessed feature matrix, target variable, a preprocessor (ColumnTransformer), an initialized ML algorithm, and a corresponding parameter grid as inputs. Inside the function, I split the data into Other and Test (80-20) using KFold with 4 splits and preprocess the data, and perform cross-validation through the designed pipeline. For data processing, all

the categorical features have already been transformed by the OneHoteEncoder schema in this dataset. Thus, we only deal with the numerical attributes by applying MinMaxScaler and StandardScaler. Here we could select features related to rates and NLP attributes to be preprocessed by MinMaxScaler since the rate has a clear bound, mostly from 0 to 1 or -1 to 1. Then, for the rest numerical features, we could implement StandardScaler to standardize the range of functionality of the input dataset. Here we preprocessed 42 numerical attributes, but in total there are 57 features have been preprocessed.

After we build the function, we created a pipeline to process all the inputs in a fixed order. The pipeline would first fit_transform three folds into the training set, and the last fold used as validation. It would manage to train the ML algorithm on the training set and evaluate it on the validation set, which would repeat this step itself such that each fold will be an evaluation set once. We then use GridSearchCV to loop through all parameter combinations and collect the results. The function is designed to run 10 times for 10 different random states and the function should return 10 best models and best scores. We used the accuracy score as our evaluation metric since there are only two classes and the thing we care about most is whether the number of shares has been correctly classified into the right class. We take the mean of these 10 test scores as our final test scores.

We in total have deployed five different models. They are Logistical Regression, Random Forest, KKN, SVC, and XGBoost. To measure the uncertainty of non-deterministic models like Random Forest, I run the model for ten different random forests and record each result's standard deviation and the mean of them as well to estimate the uncertainty. For each model, I tuned corresponding hyperparameters and achieved different accuracy scores. Details are shown below:

- **Logistical Regression:**
  {'logisticregression__C': [100, 10, 1.0, 0.1, 0.01]}
  Accuracy score: 62.81%

- **Random Forest:**
  {' randomforestclassifier__max_depth': [1, 3, 10, 30，100],
  'randomforestclassifier__max_features': [0.25,0.5,0.75,1.0]}
  Accuracy score: 64.18%

- **KKN:**
  {'kneighborsclassifier__n_neighbors': [1,10,30,50,80,100]}
  Accuracy score: 61.29%

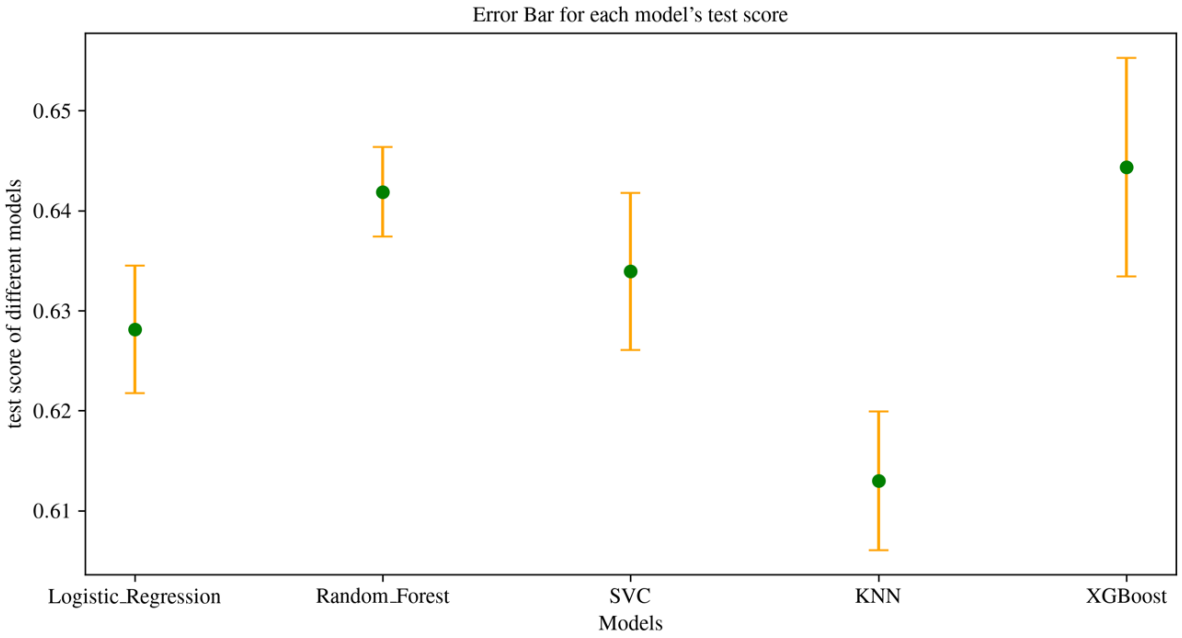
- **SVC:**
  {'SVC__C': [0.1, 1, 10, 100, 1000], 'SVC__gamma': [1, 0.1, 0.01, 0.001, 0.0001]}
  Accuracy score: 63.99%

- **XGBoost:**
  {'learning_rate': [0.0001, 0.001, 0.01, 0.1], 'n_estimators': [1,10,100,1000],
  'max_depth': [1,3,10,30,100]}
  Accuracy score: 64.43%

IV. Results

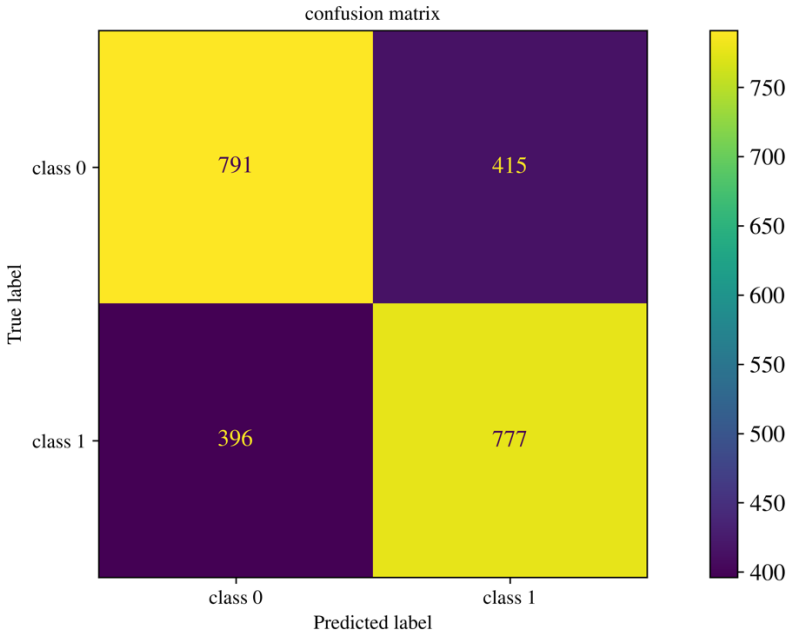
Our baseline score is 50%. All the models we deployed have reached an accuracy score above 60%, which exceeds the baseline score of about 14%. Among all the models, XGBoost becomes the most predictive model, which has got the highest accuracy score of 64.43% with a standard deviation of 0.011.

The test score of XGBoost is 9 standard deviations above the baseline. More detailed information is shown in the graph below.



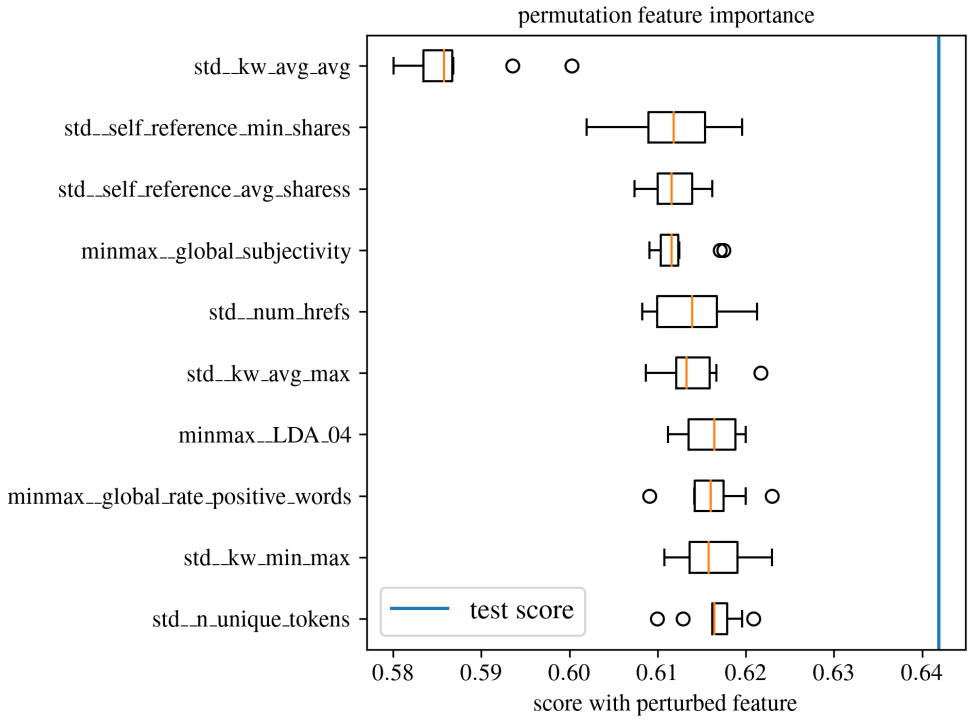
This figure shows both the mean and std of each model's accuracy score

Then, I selected XGBoost as my final model to conduct further interpretation. First, I plot the confusion matrix, where we could observe that the number of False Positive and False Negative are very close and the number of True Negative and True Positive are similar to each other. Further, I calculated the accuracy score, recall score, and precision score -- they are 65.9%, 66.2%, and 65.1% respectively, which do not stand out with a noticeable difference.

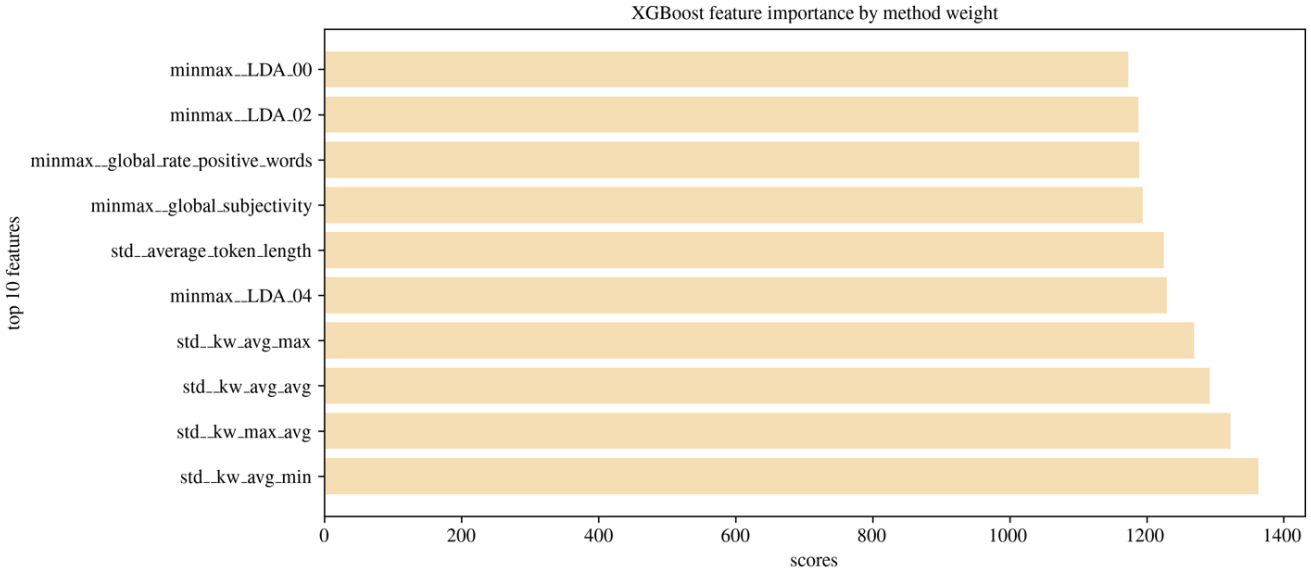


This figure shows the confusion matrix of y_true and y_pred

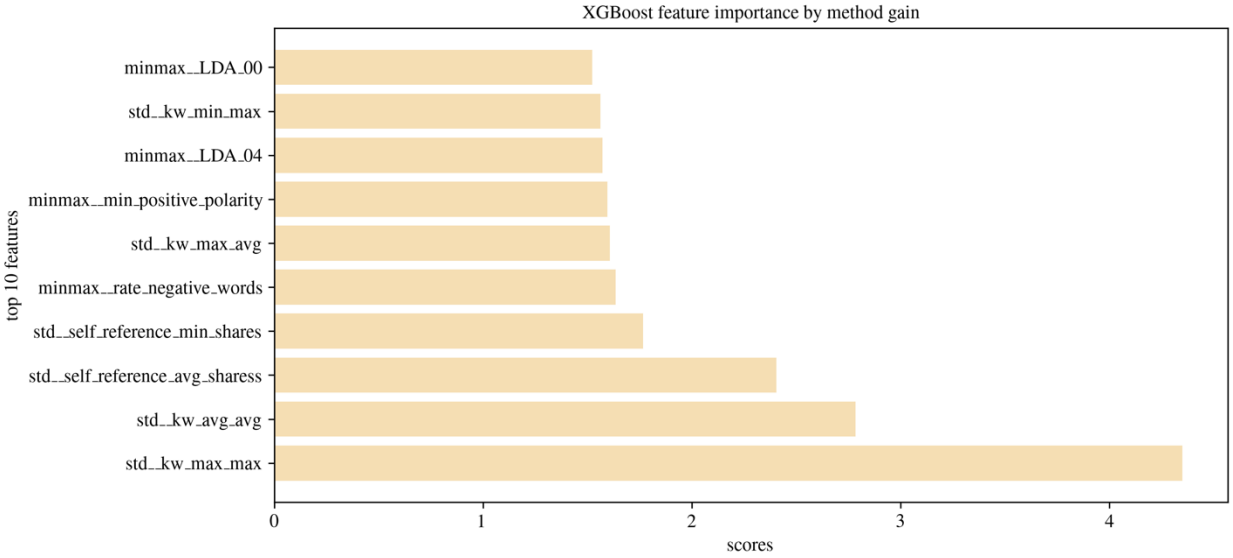
Furthermore, I calculated the global feature importance with five different methods, which are permutation feature importance, XGBoost importance (type = ‘weight’, ‘gain’, ‘cover’), and global SHAP feature importance. Graphs with details are shown below.



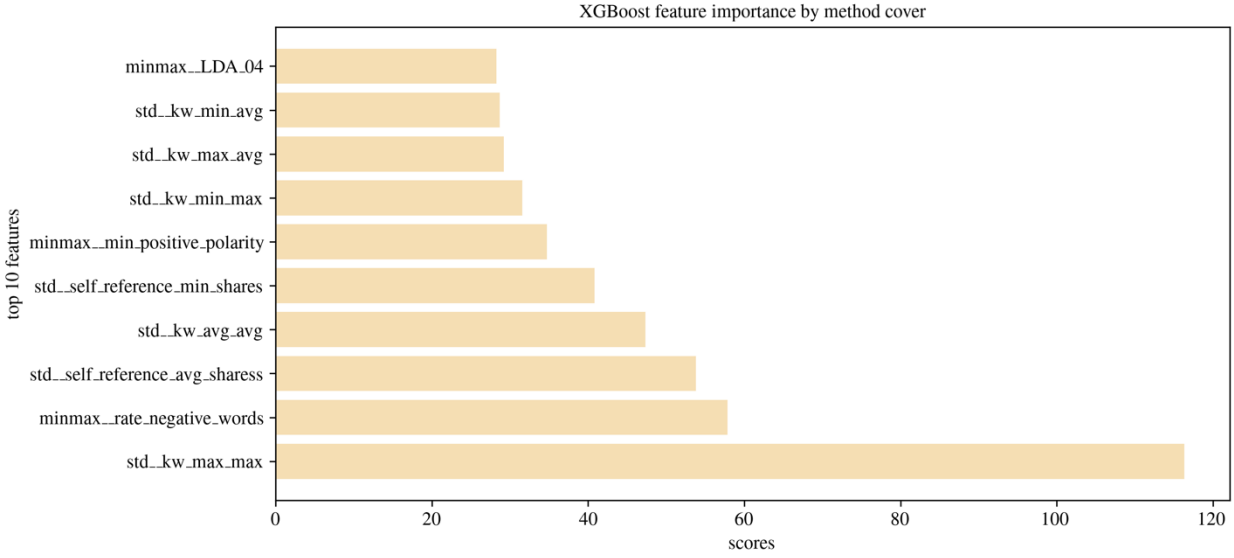
This figure shows top 10 permutation feature importance



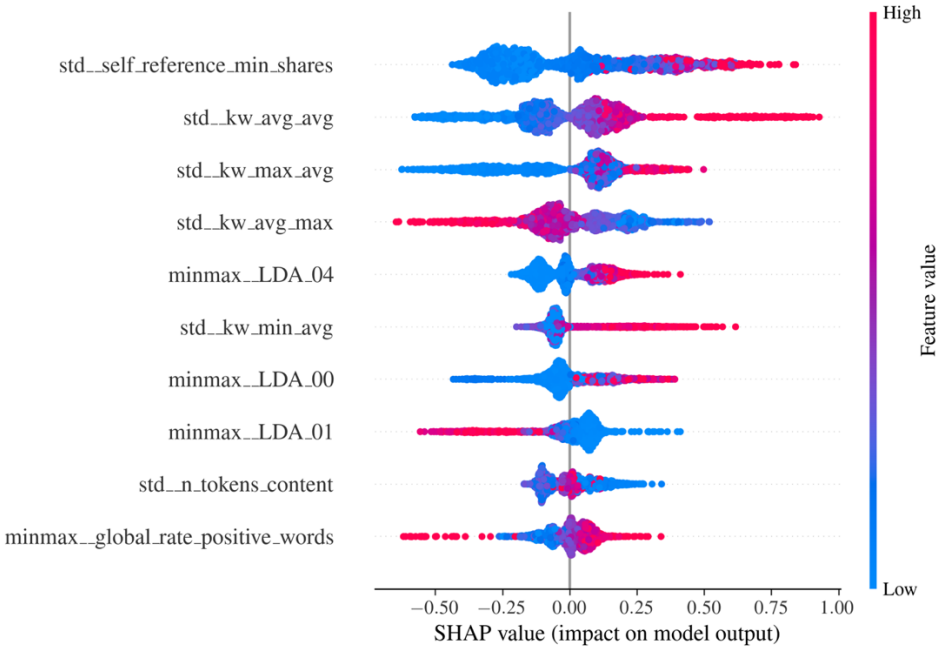
This figure shows top 10 most important features measured by method weight



This figure shows top 10 most important features measured by method gain



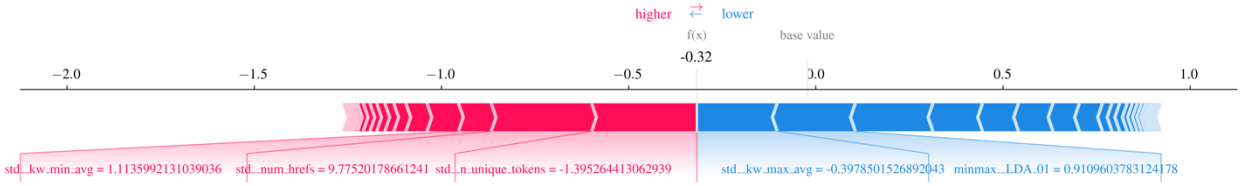
This figure shows top 10 most important features measured by method cover



This figure shows the global SHAP value for the top 10 important features

From the results of global feature importance calculated through five different methods, I found that there is a huge overlap of these top 10 important features. The features 'std_kw_avg_avg' and 'std_kw_max_max' always stick in the top 5 features, which are basically about similar information, the number of best/average keywords in the average or max shares. This founding is within expectation since the feature 'kw_avg_avg' has the strongest correlation with our target variable from our correlation graph. However, the features 'std_self_reference_min_shares' and 'std_self_reference_avg_shares', which are about max/avg shares of reference articles, appear in a high frequency in our top 10 important features. These two features have not shown any strong correlation with our target variable either in the correlation coefficient or linear f_score calculations. However, surprisingly, the feature 'std_num_keywords', also a feature about keywords has become one of our least important features in our feature importance analysis. Also, none of the categorical features has ever appeared in any of the top 10 feature graphs no matter which method we used. This finding is quite surprising since the categorical feature 'is_weekend' and 'data_channel_is_entertainment' are our top five features in correlation graphs.

For the local feature importance, it depicts features each contributing to determine the model output from the base value (the average model output over the training dataset we passed). Generally, the features with a larger arrow would contribute more to the final prediction. Features pushing the value higher are shown in red, and those pushing the value lower are in blue. Take the 4th observation as an example，



This figure shows the local feature importance of 4th observation

From the graph we could observe that the feature in red ‘std_n_unique_tokens’ and the feature in blue ‘std_kw_max_avg’ are the most influential features that determine our final prediction for this observation.

V. Outlook

As is seen from the results above, none of the models’ accuracy scores has exceeded 70%. To further improve its performance, there are several possible ways. First, only 30% dataset instead of all data points have been used in the model deployment, which surely negatively impacted our accuracy score in any of the models. Therefore, if we could have access to more advanced equipment to run our model with the entire dataset being used, our accuracy score would be improved undoubtedly. Second, there has much improvement room for feature selection. In our EDA part, we found that most of the features have a very weak correlation with our target variables. In the future, if we could extract and add more relevant features such as all the words in the news article to the original dataset, it would help the models learn more effectively and achieve a higher accuracy score.

VI. Reference

- https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity
- https://github.com/ymdong/MLND-Online-News-Popularity-Prediction
- https://www.kaggle.com/c/online-news-popularity