Ashley Cai
DATA 1030: Hands-on Data Science
Professor Andras Zsom
Github Repository: https://github.com/ashley-cai/data1030-final

Introduction:

This project aims to look at the factors that influence the popularity of news articles, which is measured through the share count of articles published on Mashable.com. The project examines the UCI Online News Popularity dataset, and is a classification problem to predict whether an article will popular be before publication.

I've worked in newsrooms previously, and the articles that get popular sometimes differ from what news we think is important. The newsroom is a source of truth and critical for information sharing, so it would be helpful to see what factors influences popularity and how to boost important news, as well as know which articles would do naturally well with more promotion and investment.

The 39797 articles were collected from Mashable.com in a 2 year time span, and all data is publicly available. I am using 20% of the total dataset in this project for speed and efficiency, as my laptop cannot reasonably handle the full dataset. There are 60 features aside from the target variable(shares), some of which are directly retrievable. Two of those(URL and Days between the article publication and the dataset acquisition), are non predictive. Other statistics were derived through applying other algorithms to the publicly available text, such as the Latent Dirichlet Allocation (LDA) algorithm.

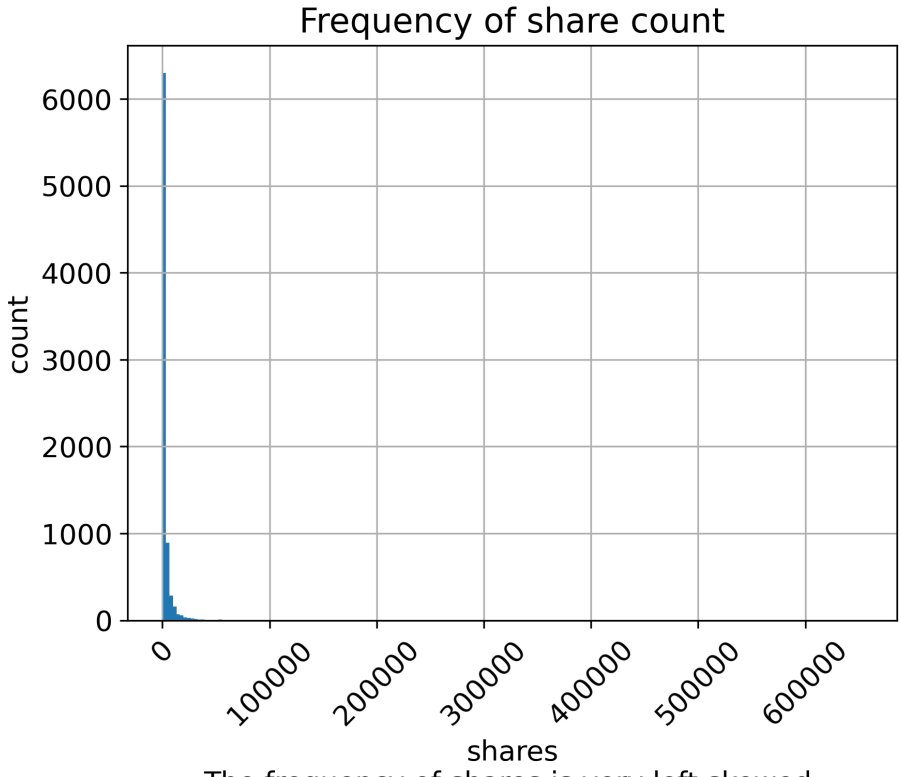
Fernandes, Vinagre, and Cortez from the Universidade do Porto, Portugal used this dataset to predict whether an article will become popular prior to publication. They tested 5 different classification models, and the most successful was Random Forest, with an accuracy of .67 and a recall of .71. According to the paper, the most impactful features are Avg. keyword (avg. shares), Avg. keyword (max. shares), Closeness to top 3 LDA topic, and Article category (Mashable data channel).

Another paper by Tsai and Wu from National Yang Ming Chiao Tung University uses the same dataset to find that the most successful classification model was One-Class SVM, with 88% accuracy. They found that controversial key phrases were indicative of more popularity.

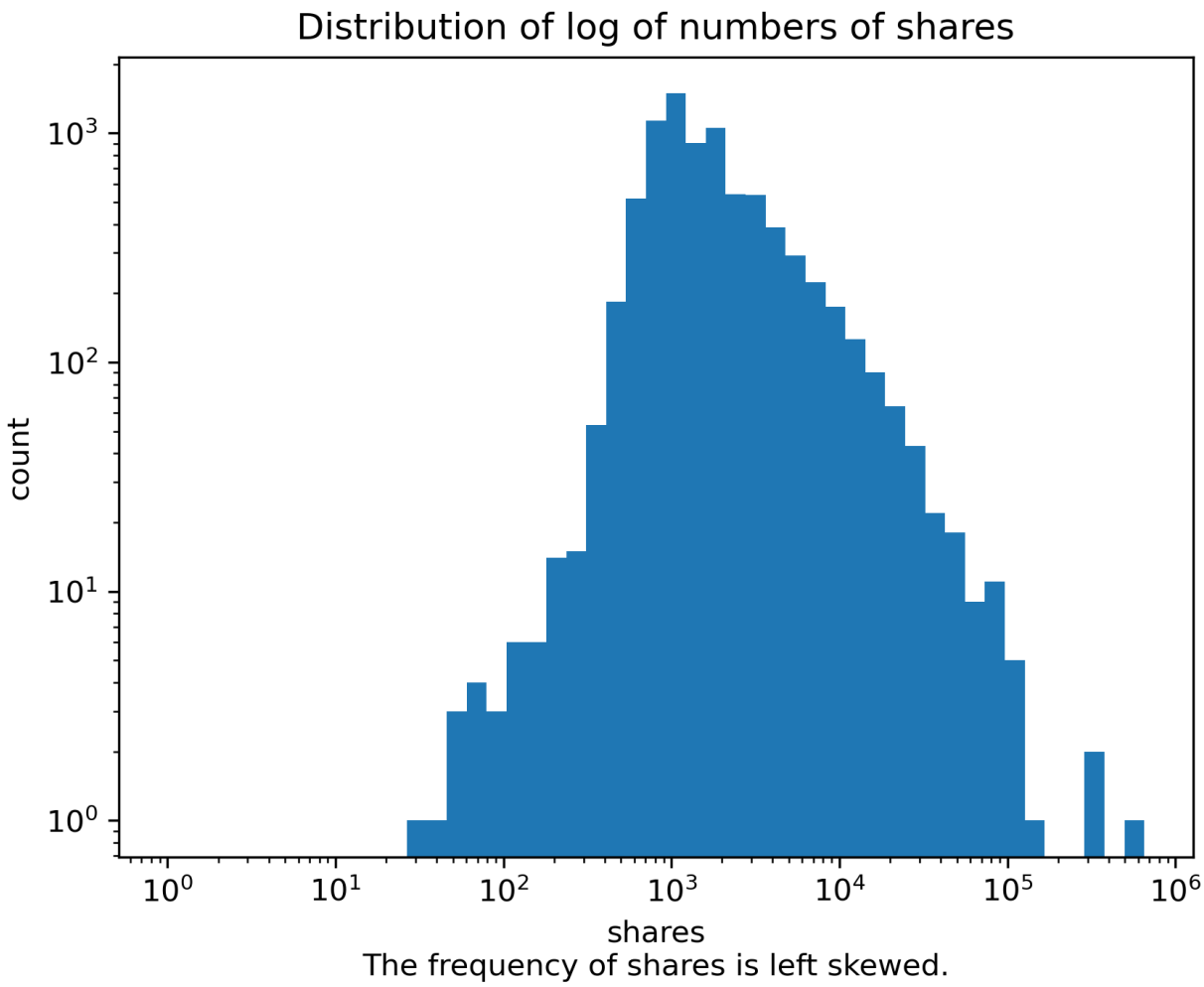
Exploratory Data Analysis

Shares(target variable)

```
count      7929.000000
mean       3327.424896
std       10985.766042
min          28.000000
25%         937.000000
50%        1400.000000
75%        2800.000000
max      652900.000000
```

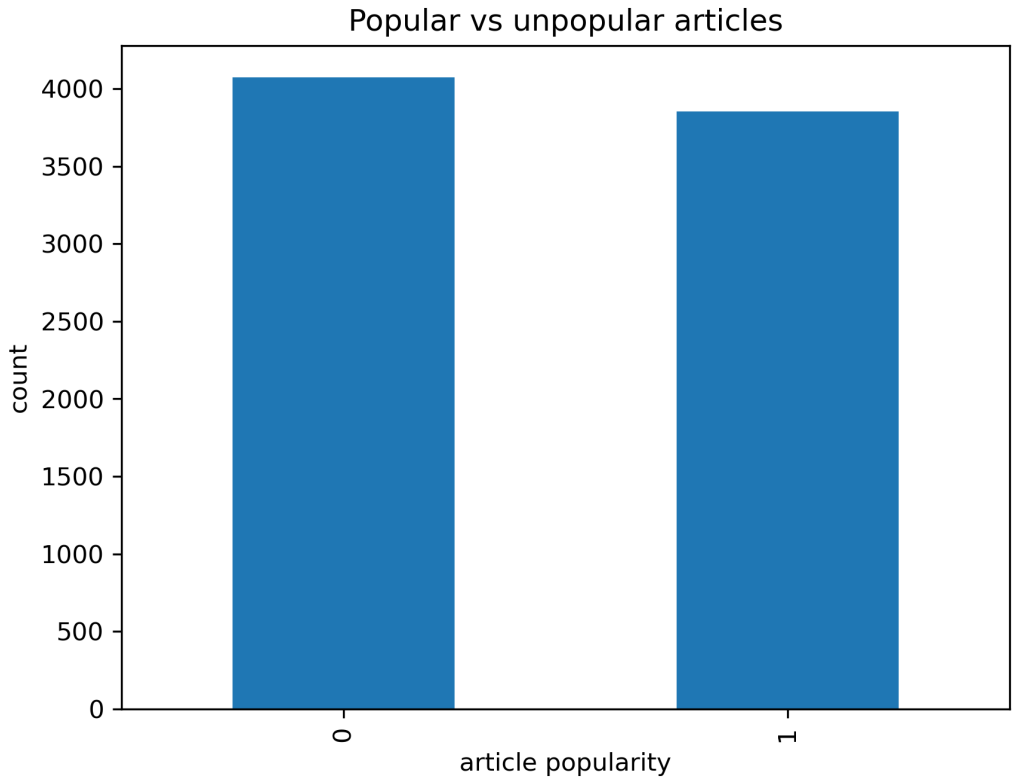


The frequency of shares is very left skewed.

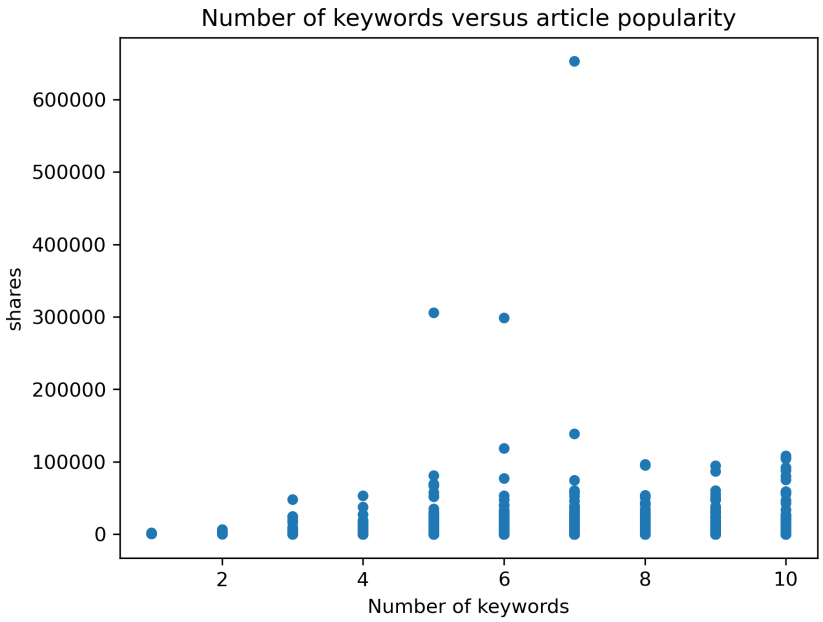


The frequency of shares is left skewed.

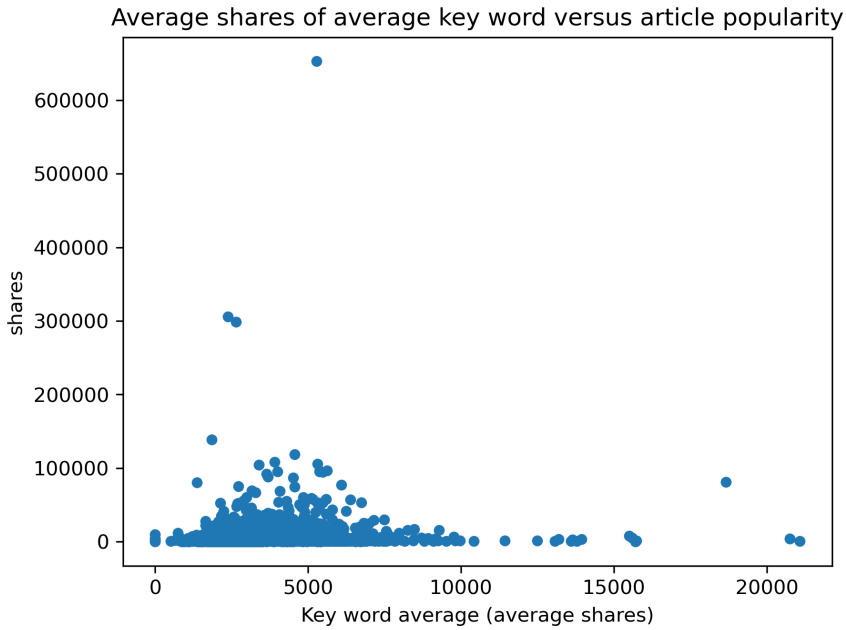
As seen through the statistical summary and the graphs above, the graph is very heavily left skewed, and the values vary over many orders of magnitude. This means that there are a lot of articles that have lower share counts, centering around 1000-3000, and a few outliers that surge into the hundreds of thousands.



The median of the dataset is 1400 shares, and the mean is 3000, which means that the majority of shares are lower than 3000. I chose to do a classification problem because a quick test of the regression problem proved to be wildly inaccurate, and historically other papers have treated this issue as a classification problem. I chose to split the popularity and unpopularity at 1400, as that is the median and I wanted to keep the dataset balanced.

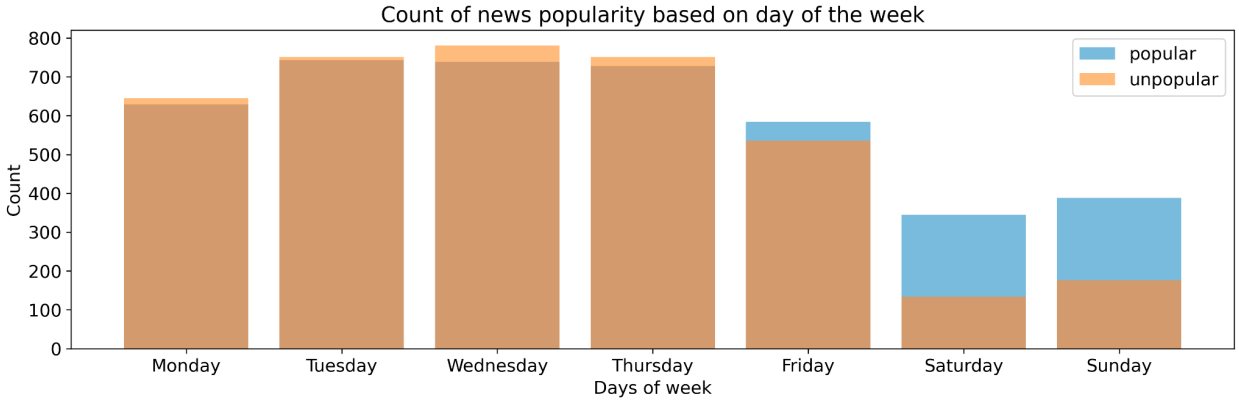


The number of keywords appears to show a slight correlation with the number of shares.

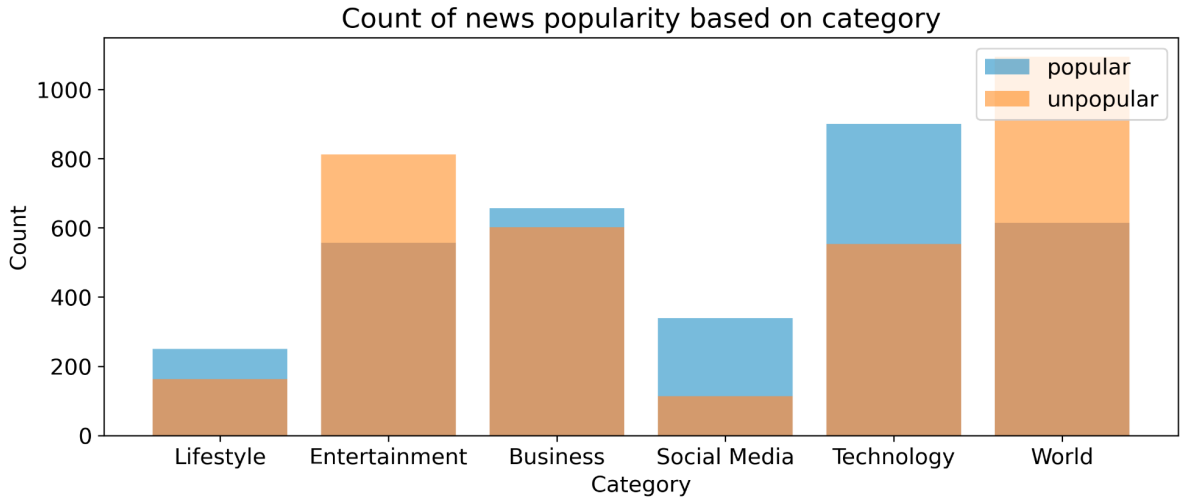


The keyword average appears to show a slight correlation with the number of shares.

These two continuous variables show a slight correlation between the amount of keywords and the number of total shares and the popularity of keywords in the article and the number of total shares. This makes sense, as keywords themselves help with article SEO and social media popularity, and we are looking at a dataset that focuses on digital news. Additionally, the popularity of the keyword both helps with SEO and indicates the popularity of the article topic.



The proportion of popular news on weekends is much higher than on weekdays.



The proportion of technology and social media articles that are popular is higher than the other categories.

The graphs above are also illuminating about the impact of different categorical variables on article popularity. As seen, the proportion of articles published on Saturday and Sunday are more popular, yet there are less total volume published on those days in comparison to the rest of the week. This makes sense, as readers have more free time to read articles on weekends. This expresses some potential in delaying article publication in order to align with the weekend to maximize popularity.

Additionally, technology and social media articles appear to be the most popular category or article, whereas world and entertainment are less popular, despite entertainment and world having high volumes of publication. This shows some potential to cover more articles in other categories.

Methods

Preprocessing

I used StandardScaler on the continuous features because they were not reasonably bounded and MixMaxScaler is only used on continuous features that are bounded. The categorical features appear to be preprocessed already, as each category is split into a binary variable. There are 60 total features in the data – 58 are predictive features, and 2 of them are not.

Additionally, there appear to be no missing values in any of the features, so there are no features that need to be imputed or otherwise addressed. There are also far more data points than features, so there does not need to be feature selection.

Splitting

The dataset is not grouped or time-series, so it is considered IID. The data is heavily skewed with a few outliers of several magnitudes greater than other articles, but the classification problem is through the median, which makes the split of the data relatively evenly distributed(as seen in the prior section). Therefore, a kfold split makes the most sense considering the ML question and future use in order to properly test and validate the data, and to make sure I don't overfit on my original test data. I did an 80-20 split with my other and test, and ran kfold with 4 splits on the other.

Selecting an ML Algorithm

I tested 5 different ML Algorithms and tried different hyperparameters for all of them. My ML pipeline involved preprocessing, splitting, and feeding into a GridSearchCV for hyperparameter tuning. I ran each GridSearchCV three times to find the best average scores and measure uncertainty due to non-deterministic models.

Logistic Regression

Logistic regression is a linear classification model. It is quick to train and to predict. The hyperparameters I tested on it:

```
'logisticregression__penalty':   ['none', 'l1', 'l2', 'elasticnet']
'logisticregression__l1_ratio': np.linspace(.1,1,num = 3)
'logisticregression__C': [100, 10, 1.0, 0.1, 0.01]
```

RidgeClassifier

Ridge regression is a linear classification model. It is also quick to train and to predict. The hyperparameters I tested on it:

```
'ridgeclassifier__alpha': np.linspace(.01,1,num = 4)
```

SupportVectorClassifier(SVC)

Support Vector Classifier is a linear classification model. It works well on datasets with a high margin of separation, but is slow on large data sets. The hyperparameters I tested on it:

```
'svc__C': [0.1, 1, 10, 100]
'svc__gamma': [1, 0.01, 0.0001]
```

K neighbors classifier

K neighbors classifier is a non-linear classification model. It doesn't make assumptions about underlying data, but is slow with large datasets. The hyperparameters I tested on it:

```
'kneighborsclassifier__n_neighbors' : [30,100],
'kneighborsclassifier__weights': ['uniform', 'distance']
```

Random forest classifier

Random forest classifier is a non-linear classification model. It works well with non-linear data, however, it is slow to train on large datasets. The hyperparameters I tested on it:

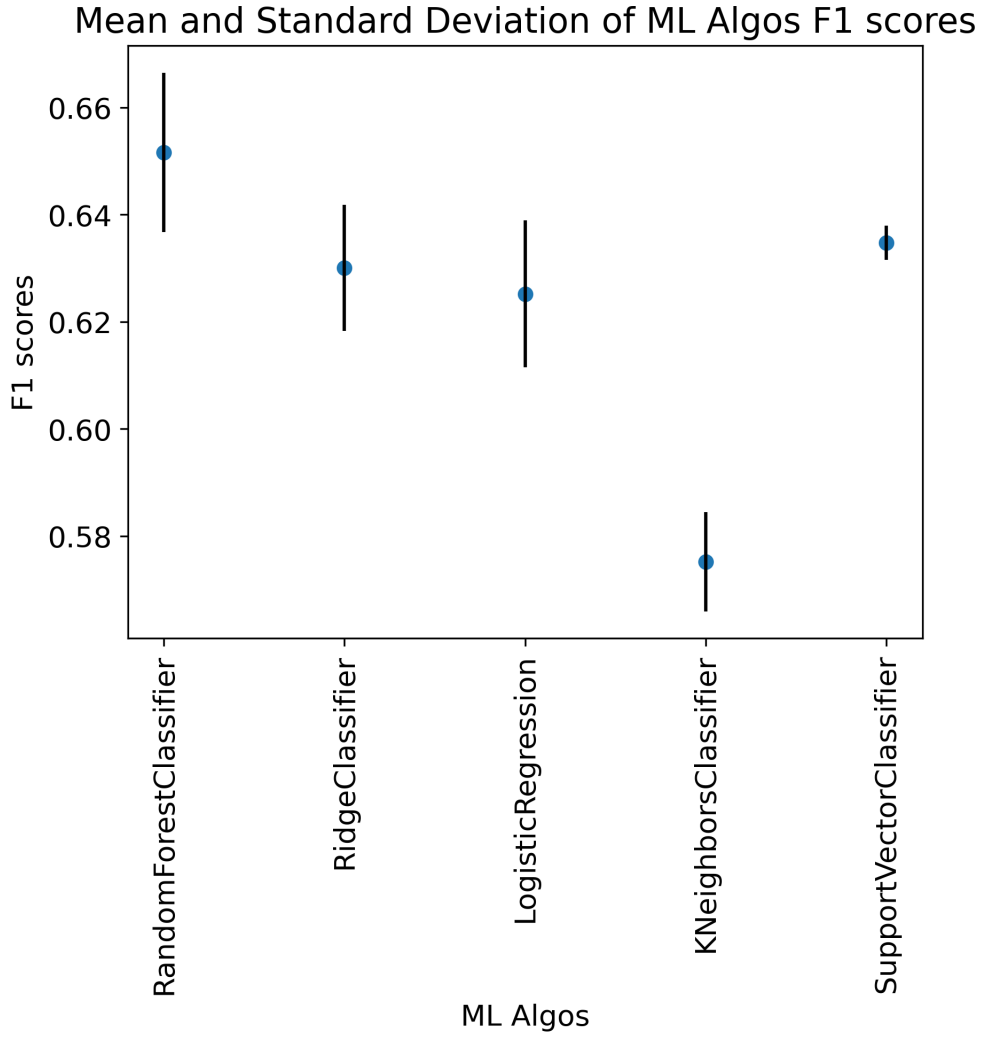
```
'randomforestclassifier__n_estimators': [100, 300]
```

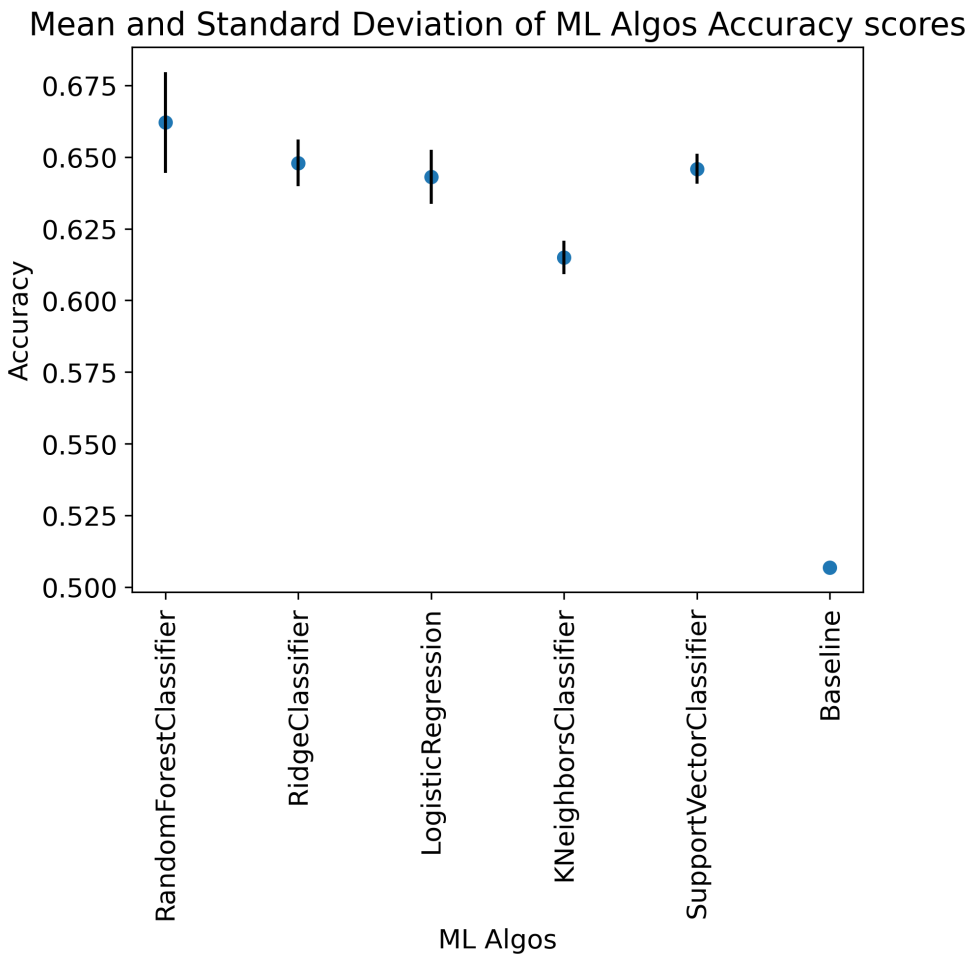
Metrics

I decided to use the metrics Accuracy, F1 Score, RMSE, and ROC to measure the results of my ML algorithm. I chose these because they give me a diverse spread of results. Accuracy because it show the ability for my ML algorithm to predict correctly, F1 score because it balances precision and recall on the positive class, RMSE because we want to prevent large errors, and ROC-AUC because it is built for classification problems.

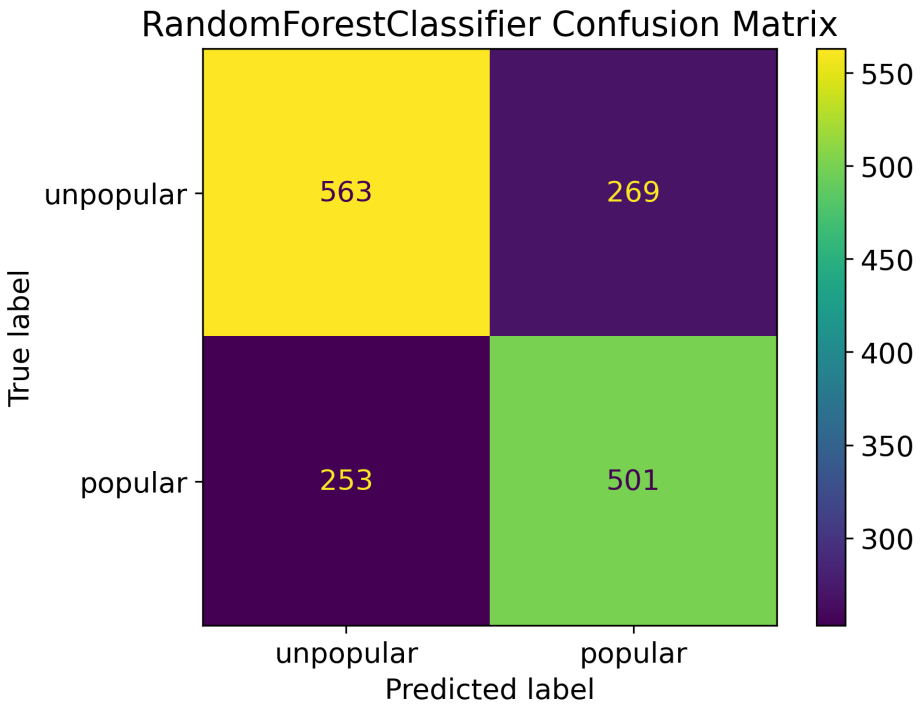
Results

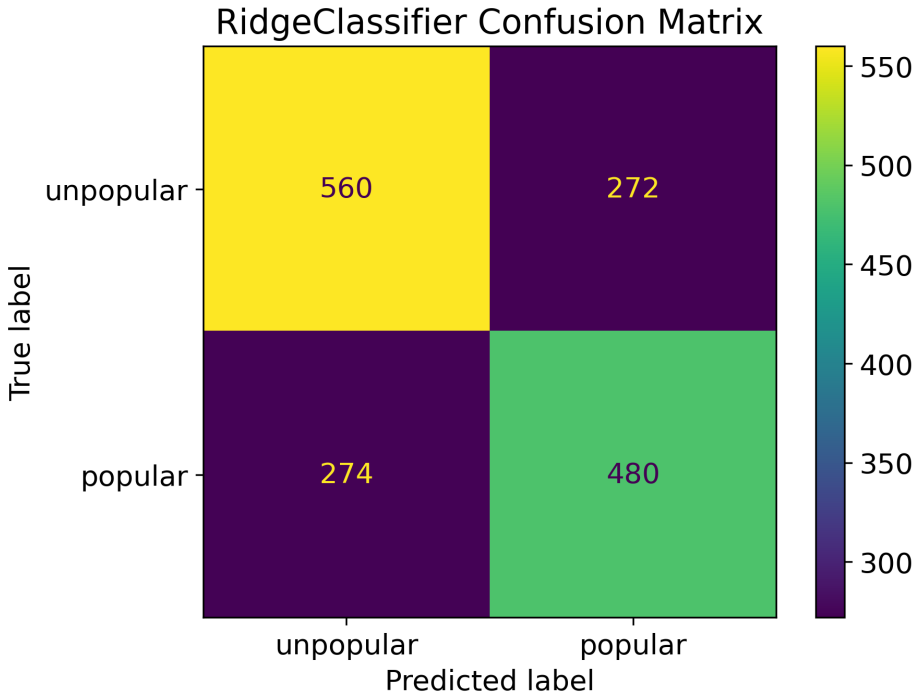
	model	Accuracy	F1Scores	RMSE	ROC
0	RandomForestClassifier	0.662043	0.651628	0.581146	0.661913
1	RidgeClassifier	0.647961	0.630047	0.593290	0.648181
2	LogisticRegression	0.643127	0.625191	0.597337	0.643392
3	KNeighborsClassifier	0.614964	0.575225	0.620494	0.616095
4	SupportVectorClassifier	0.645860	0.634722	0.595080	0.645648
5	Baseline	0.506936			

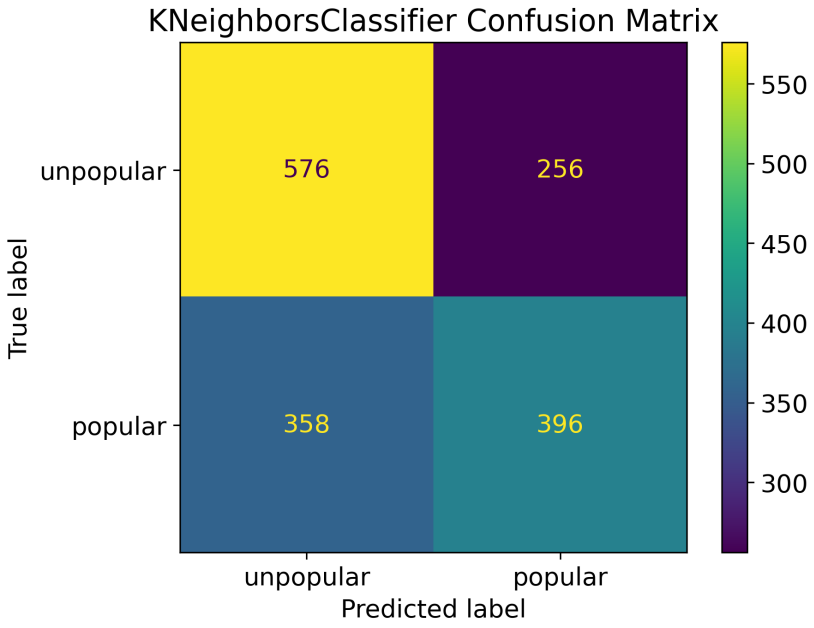


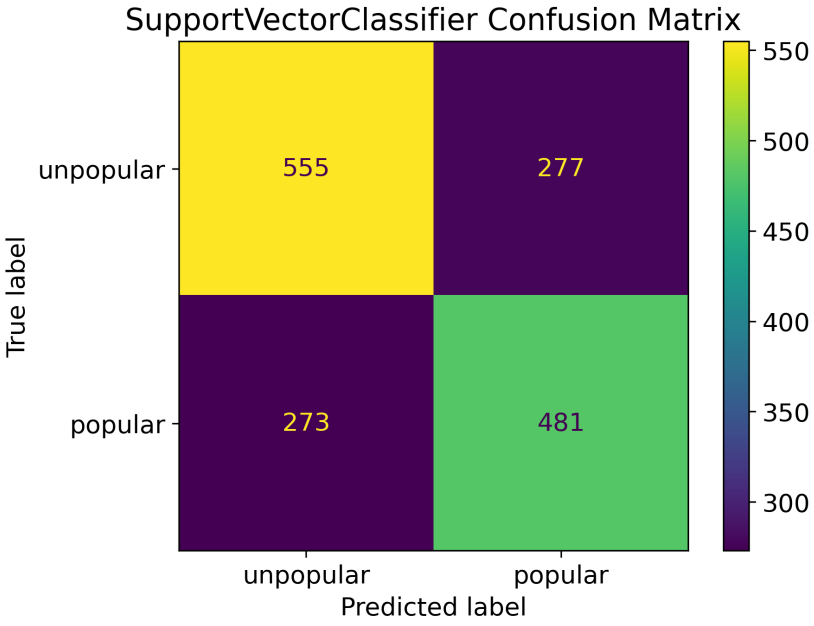


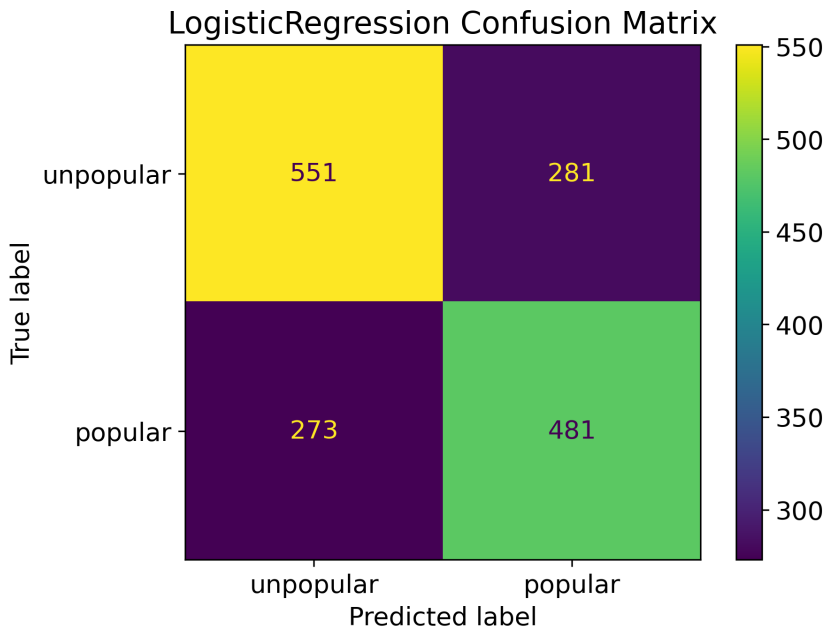
As seen by the results above, the Random Forest classifier did better than the other type of classifiers on every metric, with an average accuracy of 66%. However, it does have a larger standard deviation, which likely could be shrunken by more runs of the GridSearchCV and a larger dataset. While the Random Forest classifier's standard deviation is larger than the rest of the classifiers, it is still significantly better than all of them, which led me to choose the Random Forest Classifier to refine further. All of the classifiers do significantly better than the baseline accuracy of around 50%. The Random forest classifier, having the largest standard deviation, still achieved an accuracy 5 standard deviations above the baseline.


RandomForestClassifier Confusion Matrix
unpopular
popular
True label
563
269
253
501
unpopular
popular
Predicted label
550
500
450
400
350
300


RidgeClassifier Confusion Matrix
unpopular
popular
True label
560
272
274
480
unpopular
popular
Predicted label
550
500
450
400
350
300


KNeighborsClassifier Confusion Matrix
unpopular
popular
True label
576
256
358
396
unpopular
popular
Predicted label
550
500
450
400
350
300


SupportVectorClassifier Confusion Matrix
unpopular
popular
True label
555
277
273
481
unpopular
popular
Predicted label
550
500
450
400
350
300



As seen above through the confusion matrices, the Random Forest classifier was able to achieve the best overall accuracy rate. Interestingly, the K Neighbors Classifier was able to predict truly unpopular articles the most accurately, but was unable to predict truly popular articles as well as the other types of classifiers.

Final Model

I then tuned the hyperparameters for the Random Forest classifier more thoroughly:

```
'randomforestclassifier__n_estimators': [100,300,500],
'randomforestclassifier__max_features': [3,7,10],
'randomforestclassifier__min_samples_leaf': [1,3,7]
```

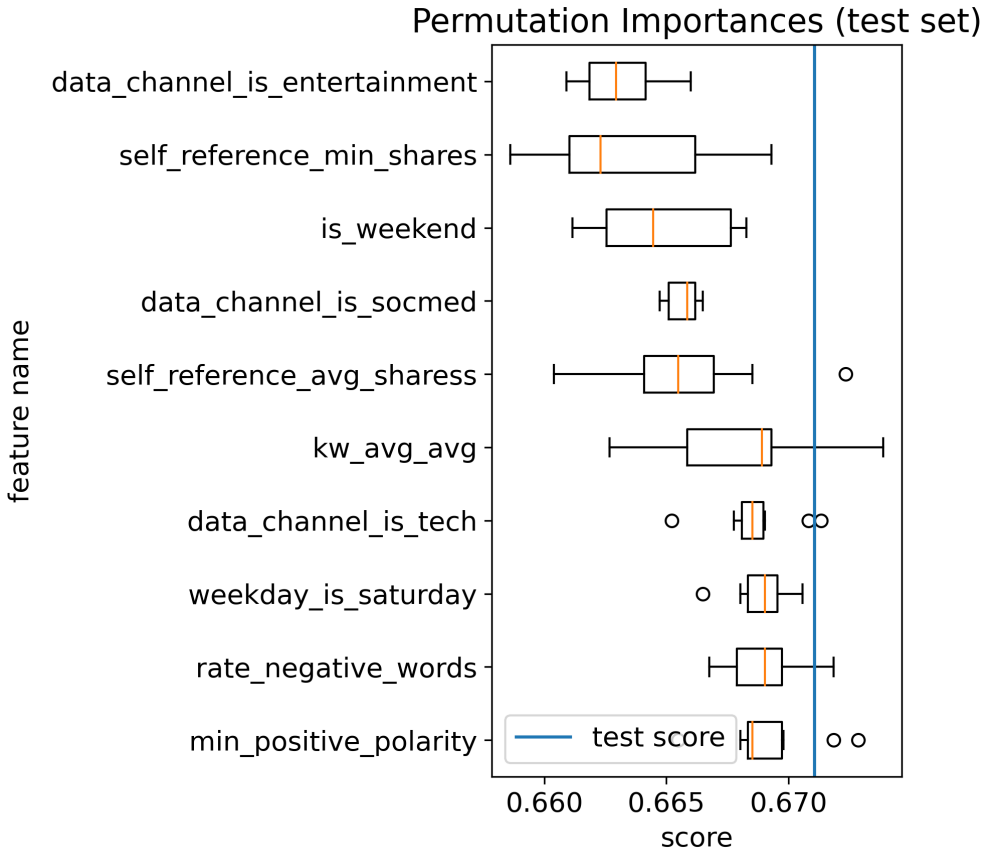
This led to a final model with the parameters:

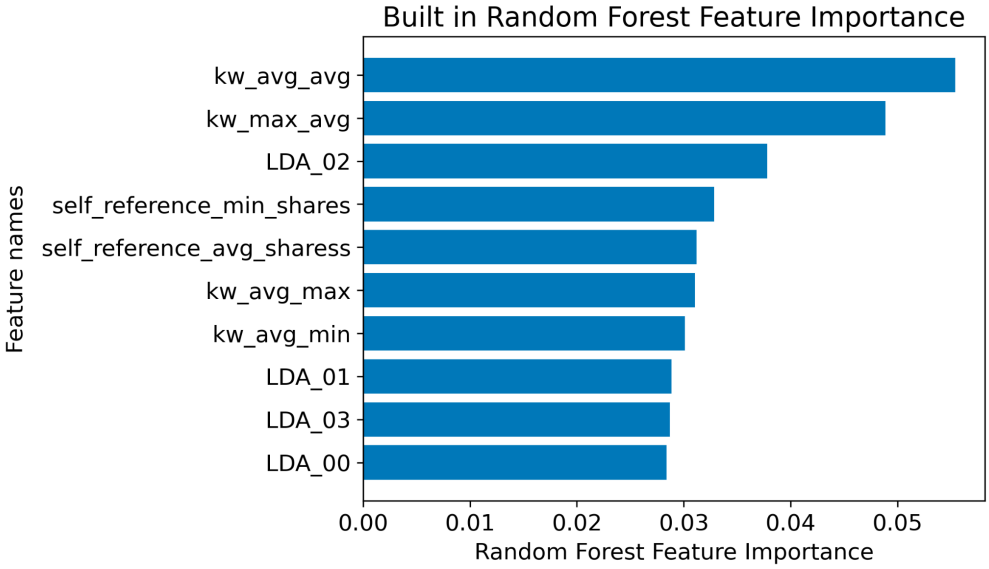
```
{'randomforestclassifier__max_features': 10,
'randomforestclassifier__min_samples_leaf': 3,
'randomforestclassifier__n_estimators': 300}
```

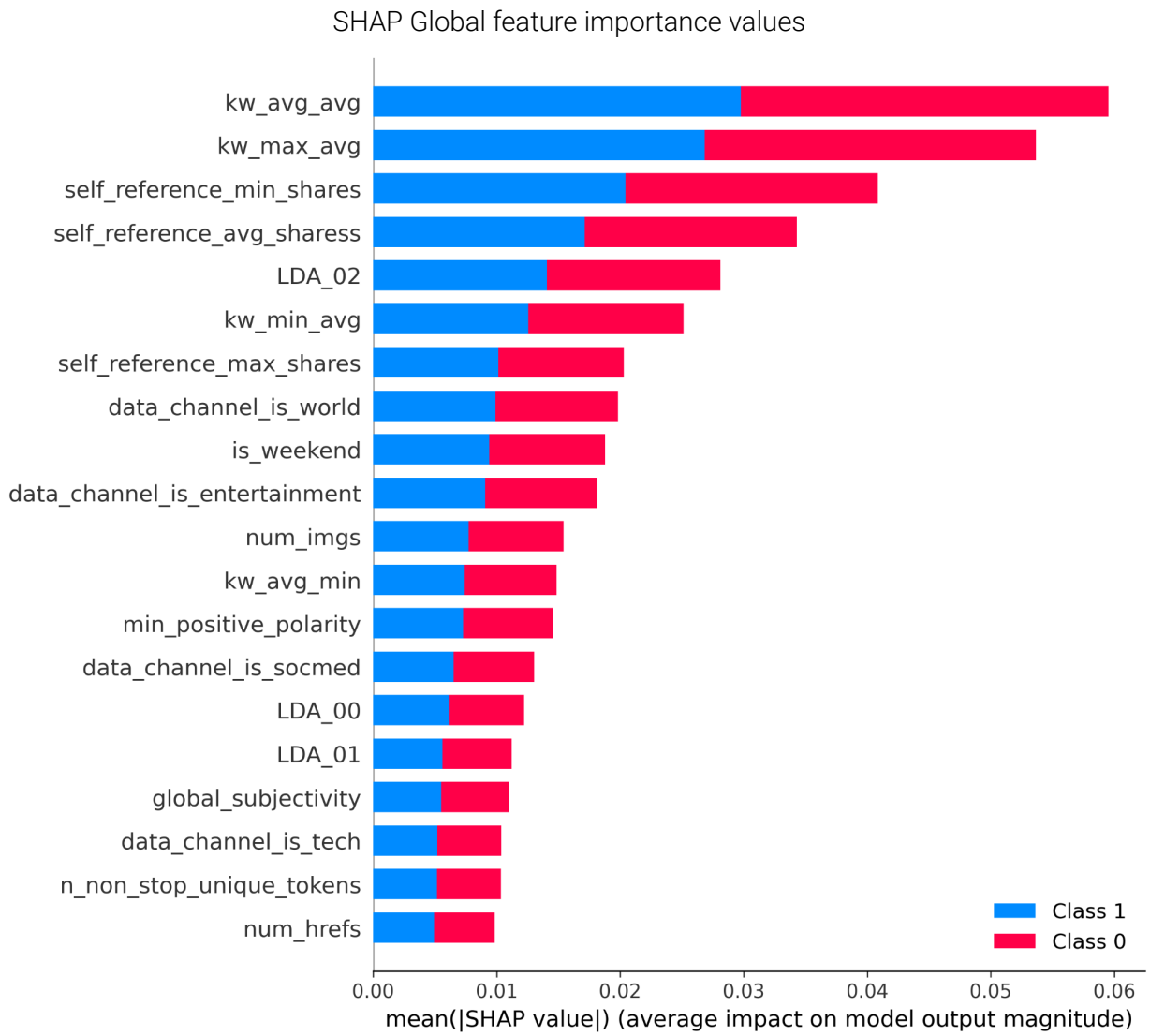
And the scores:

```
RMSE: 0.5747964166162268
Accuracy: 0.669609079445145
F1: 0.6601815823605708
ROC: 0.669614183764496
```

Global feature importance





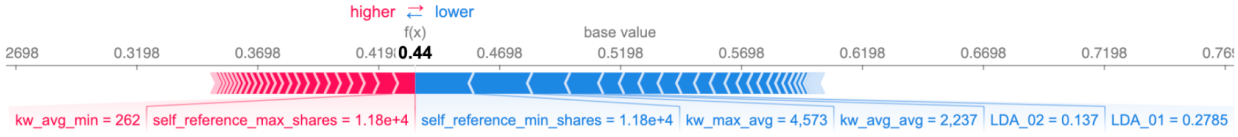


From the three types of calculations for global feature importances, several grous pf features appear to emerge. First is kw_avg_avg, kw_avg_avg, kw_min_avg, kw_avg_min, all appear to have substantial impact on the algorithm's predictions. Those are measuring the popularity of the keywords in the article, which makes sense, as they indicate how popular the article is and can help digital articles with SEO. The second is LDA, which is short for Latent Dirichlet allocation, which is natural language algorithm that detects how related the articles are to the top most popular topics at the time of publication. The last two are data_channel and is_weekend, which show category the article was a part of and the day of publication for the articles. As we expected from EDA, those features would have a significant impact on the prediction.

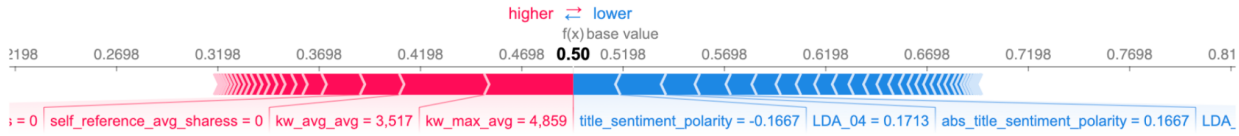
In comparison, features that did not seem to have a large impact on the prediction were subjectivity, the polarity of the article, positivity or negativity of keywords, and the number of videos in the article. This is interesting because it implies that being more

radical or approaching the article with a different tone doesn't impact popularity, which runs contrary to much popular sentiment about the radicalization of news.

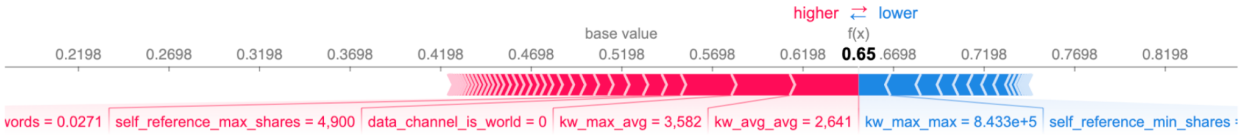
Local Feature importance



("Woody Allen Turns Wolverine Into Superhero of Hipsters")



("Windows 8.1 Cranks Up Support for 3D Printing")



("The Best Ways to Follow the Winter Olympics on Facebook")

We see similar trends showing up in local feature importance, with keyword popularity playing a role in the positive and negative probability of popularity, as well as the popularity of articles linked.

(Informal) A/B Testing

I drew a random sample of 30 articles from the original dataset to perform an informal AB test, testing my algorithm against my own ability to determine the popularity of the article. I drew upon my own knowledge of the world, impressions of what would be popular, and experience in the newsroom to predict popularity. My accuracy score was 56.5%, whereas the model's was 86.6%.

Conclusion

In the greater context of digital news distribution, this indicates that popularity is most strongly tied to keyword popularity, relation to popular news at the time, category, and day of the week of publication. This has multiple implications for newsrooms, as they can delay publication to a weekend for larger audience reach or publish more articles in categories that are popular. For articles that the newsrooms themselves believes are

important, they can also add more keywords and link to articles that are popular and considered relevant at the time. Additionally, this algorithm could provide a guidepost for SEO and social media teams to know which articles would naturally perform better than others, allowing them to understand which articles to put more emphasis on in promotions.

Outlook

Improvements in the model could be made, first and foremost by including all the data available. As my machine was not capable of handling more than 20% of the data, I was forced to use less than the optimal amount, which I expect to have improved the accuracy and standard deviation of the predictions. Additionally, I could have scraped the internet for my own data to not only include articles by Mashable, but from other news sources like the New York Times, Washington Post, Bloomberg, and more. Mashable has cultivated a very particular audience through its history of publication, which I'm sure also skews the popularity results.

I would also either remove some data points of really viral articles or add more viral data points, as articles with hundreds of thousands of shares were far and few between, which may have skewed by model's prediction ability.

I would also like to look at more types of ML algorithms, as I am only aware of and equipped to use a limited number of possibilities. One of the papers I was read combined an autoencoder and one class SVM to obtain a higher accuracy, and I would like to read and understand how they deployed that algorithm.

Ultimately, there are many areas for improvement in my predictions, but I'm relatively satisfied with the prediction capabilities in my model.

References

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
Online News Popularity Data Set
[https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity]

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.

Min-Jen Tsai, You-Qing Wu, Predicting online news popularity based on machine learning, Computers and Electrical Engineering, Volume 102, 2022, 108198, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2022.108198.