SIADS-696 Milestone II final report

Mashable articles: share prediction, classification, & topicality clustering

By: Kyle Ziegler, Venkat Panyam, Chris Roy

Introduction

Our project developed supervised and unsupervised models related to online news popularity that enable exploration and understanding of the underlying characteristics that contribute to engagement. We employed supervised learning to predict article shares and topic/category classification, as well as unsupervised learning to uncover natural groupings of articles based on content features and popularity patterns. Our solutions could be leveraged to address the challenges associated with content popularity through two key applications: (1) an ad pricing framework that categorizes articles by expected share volume, enabling publishers to set tiered pricing for ad placements with greater confidence, and (2) an automated content classification system that can be integrated into content management systems to efficiently categorize large volumes of articles, saving editorial time and reducing human error.

We were motivated by the challenge of developing deployable solutions that can handle the volume and velocity of modern digital content. To solve these challenges, we implemented a comprehensive approach to article share prediction using multiple model families, including linear models, tree-based ensembles, and boosting with regularization. For classification, we evaluated 11 different models, including but not limited to; Dummy Classifiers, Gaussian Naive-Bayes, MLP Classifier, and SVCs with different kernels. For unsupervised learning, we explored natural groupings within articles to identify patterns that might not be captured by predefined categories, such as the relationship between content characteristics and sharing behaviors that drive engagement.

Our supervised learning exploration revealed that predictive power in article shares is highly concentrated in just a few features. For classification, our final XGBoost model achieved high accuracy scores, and we discovered that LDA topic modeling features were consistently among the most important features. Our unsupervised learning analysis revealed distinct article clusters based on content and engagement patterns, providing insights into how different types of content perform across various metrics beyond traditional categorization schemes.

Related Work

The first instance of related work is by Fernandes et al. (2015) who used the same dataset as our project to predict popularity using ensemble methods. Their work primarily focused on supervised binary classification of "popular" vs. "unpopular" articles using decision trees and random forests. Our project differed by 1) adding a new ML task of topic classification and 2) incorporating unsupervised learning to discover natural article groupings.

Another relevant work is by Kari et al. (2016) who investigated boosted online regression and proposed a novel family of regression algorithms. Their research introduced a boosting algorithm based on random updates that achieved faster performance while enhancing prediction accuracy. Our project differed by 1) focusing specifically on article share prediction rather than general regression problems and 2) comparing a broader range of model families beyond boosting algorithms to identify the most effective approach for this specific domain.

The final piece of related work is a foundational study by Tsagkias et al. (2009) that pioneered early efforts in predicting user engagement with online news/ articles by focusing on comment volume. Their research addressed predicting the number of comments articles would receive prior to publication using a two-stage classification approach by first identifying articles that could receive comments, and then classifying them in "low" or "high" categories. Our project differed by 1) focusing on article share prediction rather than comment volume as the target feature and 2) incorporating topic classification and unsupervised methodologies as additional ML tasks.

Data Source

We used the Online News Popularity dataset from the UCI Machine Learning Repository for our supervised and unsupervised learning tasks. The csv file contains a set of features related to articles published by Mashable in 2013 and 2014. The article content and the rights to reproduce them belong to Mashable, but the content can be publicly accessed using the urls provided in the dataset. In addition to the number of shares for an article, the donors of the dataset extracted several features such as counts of keywords, videos, and images from the article and also included closeness values for each article to the top 5 categories produced by Latent Dirichlet Allocation (LDA). The dataset contains 39,644 samples, each with 61 attributes. Appendix B shows summary statistics and additional details about the attributes.

Feature Engineering

The original dataset included numeric features such as number of keywords, number of images, number of videos, LDA values corresponding to 5 categories, etc,. that are based on the article content. We supplemented these with article text, title, author and publish date by scraping the article content using trafilatura and beautifulsoup. We then generated 384-dimensional dense vector embeddings for both the article titles as well as the article text using the Hugging Face sentence transformer model all-MiniLM-L6-v2. The title embeddings were then condensed to 64-dimensional vectors using PCA. The initial dataset did not have any missing data and no value imputation was needed.

The distribution of the shares column in the original dataset was skewed and a log transformed value was created as the target variable for the regression task of predicting article shares. The regression task did not use any new features and The final model was trained using 40 features from the original dataset.

For article category prediction, we added a new target column called “channel” that contained the index value of the one-hot encoded “data_channel_is_” column that had a value of 1. The dataset contains 6 such columns, resulting in our target taking on values from 0 to 5. Additional features such as the content and title embeddings, several combinations of LDA values, and the top 20 features of the XGBoost base model combined with other top 20 features were considered. Eventually the top 20 features from the XGBoost base model multiplied by num_keywords and the same 20 features multiplied by num_videos (40 new features) were used by the final classification model in addition to the original features.

The unsupervised methods also made use of the content and title embeddings. Appendix C lists the features used by the two supervised learning algorithms.

Supervised Learning- Article share prediction and categorization

Article share prediction

Our first supervised learning workflow focused on predicting article shares using a diverse set of regression models. Through the course of our work, we preprocessed the data, created log-transformed share counts to address the skewed distribution of the target variable (shares), and combined structured features with title and content embeddings to create our initial feature matrix. We selected three diverse model families to capture different underlying mechanisms: linear models, tree-based ensembles, and boosting with regularization. Linear models provided interpretable results with clear feature coefficients and served as important baselines. Tree-based ensembles captured non-linear relationships and feature interactions without requiring additional feature engineering. Finally, boosting with regularization implemented gradient boosting with additional regularization techniques to prevent overfitting.

For hyperparameter tuning, we employed grid search with cross-validation on our best performing model (Gradient Boosting). We explored parameters including: number of estimators (trees), maximum tree depth, learning rate, minimum samples split, and subsample ratio. We also conducted sensitivity analysis to understand how model performance varied with different hyperparameter values, providing insights into optimal configuration.

Overall Results Reporting

We selected Mean Absolute Percentage Error (MAPE) as our primary evaluation metric because it provided an interpretable measure of prediction error relative to the actual values, which is particularly important for share counts that vary widely in magnitude. We calculated MAPE on both the log-transformed shares (which was our direct model output) and the actual shares (after reversing the log transformation).

From a business context, the log-transformed MAPE values would be particularly useful for applications such as selling ad space on articles where the price per placement is determined by expected share volume. In such scenarios, we would use the MAPE log values to create a rules-based categorization system where articles could be classified as having low, medium, high, or premium expected share volume, with ad pricing negotiated based on these categories. However, for our model improvement analyses (ablation testing, sensitivity testing, failure analysis), we focused on actual MAPE as it provided more fruitful room for improvement compared to the log-transformed MAPE, which already showed strong results across models.

Table 1: Comparison of Model Performance (sorted by MAPE on actual shares)

Model	RMSE	MAE	MAPE (log)	MAPE (actual)	R2
Gradient Boosting Regressor	0.846323	0.624634	0.082904	0.785088	0.164428
Gradient Boosting (structured features only)	0.851341	0.627058	0.083317	0.793217	0.154491
Linear Regression	0.867304	0.642525	0.085332	0.813759	0.122487
Ridge Regression	0.866596	0.641822	0.085241	0.814175	0.123918
Random Forest Regressor	0.868591	0.646905	0.086059	0.832261	0.119880
Lasso Regression	0.925897	0.703341	0.093642	0.882649	-0.000084
XGBoost Regressor	0.903115	0.671789	0.089131	0.891315	0.048525

Our 5-fold cross-validation results for all models revealed significant differences in stability and performance:

	model	rmse Mean	rmse Std	mae Mean	mae Std	r2 Mean	r2 Std
0	linear regression	0.877779	0.047942	0.649820	0.049338	0.098666	0.041679
1	ridge regression	0.877779	0.047943	0.649824	0.049342	0.098667	0.041675
2	lasso regression	0.930430	0.049361	0.705461	0.044325	-0.011896	0.007794
3	random forest	0.880597	0.046303	0.656413	0.044296	0.092068	0.054579
4	gradient boosting	0.876239	0.053554	0.646912	0.048318	0.098902	0.095053
5	XGBoost	0.942560	0.053577	0.704035	0.047818	-0.042543	0.102758

With the highest mean R2 and lowest RMSE, Gradient Boosting was confirmed as our best performing model, cementing it as our primary model for further analysis and optimization.

Feature Importance and Ablation Analysis

For our Gradient Boosting model, we conducted feature importance analysis which revealed the top five most influential features were: kw_avg_avg, self_reference_avg_sharess, self_reference_min_shares, num_hrefs, and kw_max_avg. These features collectively accounted for over 50% of the model's predictive power, with keyword-related metrics and self-reference metrics being particularly important.

Our ablation study tested different feature subsets, focusing on actual MAPE as our evaluation metric. Using only the top 5 features resulted in a MAPE of 0.8310, while using the top 20 features achieved a MAPE of 0.8014, nearly matching full model performance, and using only bottom-half features resulted in significantly worse performance. This analysis demonstrated that a smaller subset of well-chosen features can achieve comparable performance to the full feature set.

Sensitivity Analysis

We conducted sensitivity analysis on three key hyperparameters: number of estimators, maximum tree depth, and learning rate. Overall, performance improved with more trees, with ideal performance at 300 trees, a depth of 5, and a learning rate of 0.2. We also analyzed sensitivity to training data size, finding that performance improved consistently with more data, but with diminishing returns after using 60% of the training data. This suggests our dataset size was adequate, but that more data could still yield marginal improvements.

Performance Tradeoffs

Our evaluation revealed several important tradeoffs, all assessed using actual MAPE. An examination of accuracy vs. speed showed that more complex models like Gradient Boosting achieved better accuracy (MAPE of 0.8020) but took significantly longer to train compared to simpler models like Ridge Regression, with only slightly worse performance (MAPE of 0.8104). Comparing feature count vs. accuracy revealed that using all 40 selected features provided the best performance (MAPE of 0.7932), but using just the top 20 features achieved comparable results (MAPE of 0.8014) with half the feature complexity. Finally, an examination of embeddings vs. no embeddings conveyed that including embeddings increased the

feature space from 40 to 507 dimensions but provided negligible performance improvement (MAPE difference of just 0.0081), and that the model without embeddings was significantly faster to train and more interpretable.

Failure analysis

We identified three distinct categories of prediction failures, focusing on actual share counts rather than log-transformed values. Case 1 identified the highest APE, which was for the article: http://mashable.com/2013/04/01/troll-appreciation-day/. The actual shares were 4, while the predicted shares were 2,051 for an APE of 511.77. This article had extremely low shares despite having characteristics that typically indicated higher popularity. Case 2 surfaced a high-volume share article with overestimation, which was: http://mashable.com/2014/01/23/aol-gravity-acquisition/. Actual shares were 1,400, while the predicted shares were 6,105 for an APE of 3.36. While this article generated a solid amount of shares, the model significantly overestimated the number, suggesting that it overvalued certain features. Case 3 revealed an article that was estimated to perform well but did not, which was: http://mashable.com/2013/12/20/johnny-depp-transcendence-teaser/. Actual shares were 934, while predicted shares were 6,930 for an APE of 6.42. This article had reference patterns similar to popular content but failed to gain traction, suggesting that factors not captured in our features affected sharing behavior.

Potential improvements:

For article share prediction, we identified the following areas for further investigation as means to improve overall model performance: 1) Incorporate additional temporal features (time of day, seasonality), 2) add features related to content virality factors (emotional tone), 3) develop separate models for different content channels, 4) Implement ensemble methods specialized for different share ranges, and 5) Collect more training data, especially for content with extremely high or low shares.

Article categorization

Our second supervised learning task focused on categorizing articles into one of six classes- lifestyle, entertainment, business, social media, technology and world. We created a new target variable called “channel” from the one-hot encoded values for each of these 6 categories (original columns- data_channel_is_lifestyle, data_channel_is_entertainment, data_channel_is_bus, data_channel_is_socmed, data_channel_is_tech, data_channel_is_world) present in the original dataset. The “channel” column takes on values between 0 and 5 corresponding to the category description as shown.

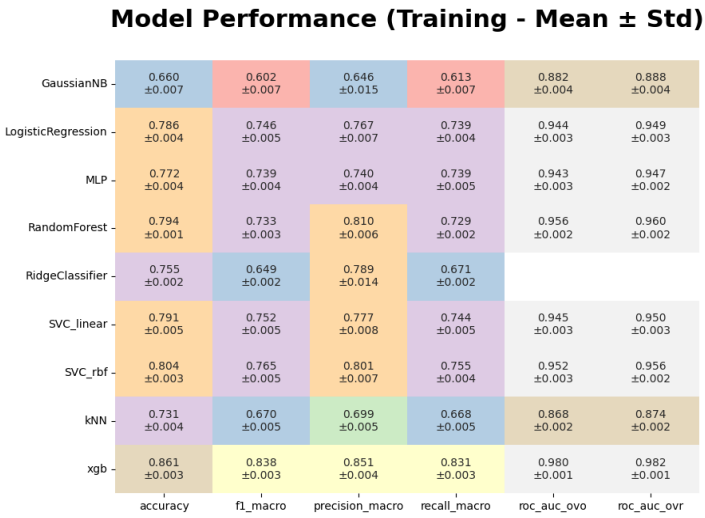
Category	Description
0	lifestyle
1	entertainment
2	business
3	social media
4	technology
5	world

The only features with missing data were the article text, title, author and publish date columns that we added later. These columns are not used for the category classification task. In addition to the one-hot encoded category columns, the categorical columns (article content, title, author, url), and the article publish date, two numeric features- shares and timedelta were excluded from the set of features used for training the base classification models.

Methods and evaluation:

Similar to the article share prediction methodology, we tried diverse model families such as linear models, tree-based models, boosting models, etc. to capture the relationship between the target variable (“class”) and the remaining features. For an initial comparison of different classification models, we trained 11

models using the original dataset. These include Dummy Classifier (most_frequent and uniform), Gaussian Naive-Bayes, k-Nearest Neighbors (k=3), Logistic Regression, Ridge Classifier, Random Forest (max depth=10), XGBoost, MLP Classifier (3 hidden layers with 128, 64, 32 nodes), SVC (kernel=rbf and kernel=linear). This gives us a comparison across several model families, exploring the strengths of each for the current task. We used scikit-learn implementations for all models except XGBoost. All models were trained with an 80/20 split of data and 5-fold cross validation over the training split. We then compared the mean values of several metrics such as accuracy, f1(macro), precision (macro) and recall(macro) to select one model to tune and evaluate using additional features. The most frequent article category (Channel #5 - “world”) accounts for about 21% of all the articles. So a dummy classifier that predicts the most frequent class will have an accuracy of around 0.21. Of the initial models, XGBoost, with the highest scores across all the metrics we considered, was the best performer. It had a training accuracy of 0.861 ± 0.003, roc_auc scores of 0.98 and an f1 macro score of 0.838 ± 0.003.



The top 5 features for the XGBoost model (from the model’s feature_importances_ values) for this training run were the 5 LDA features. We created new features by combining the top 20 features with other top 20 features to see how the XGBoost model performance changed with these new features. We eventually settled on 40 new features, created by multiplying each of num_keywords and num_videos with the top 20 features from the base model, resulting in a mean test accuracy of 0.884 ± 0.002, an f1 macro score of 0.869 ± 0.002 and corresponding test values of 0.896 and 0.882 respectively. The top 10 feature_importances_ columns include the LDA features LDA_00, LDA_02, LDA_04, the new features created by multiplying LDA values with num_keywords, num_keywords and kw_min_min, accounting for about 57% of the model’ predictive power. Adding combinations of LDA values and embeddings did not improve model performance.

Another notable observation was that the top-2 test accuracy of this XGBoost model was 0.980, indicating that when an article can potentially fall into one of two categories, the model can match the target variable with very high accuracy if we consider the top two predictions.

Hyperparameter Tuning:

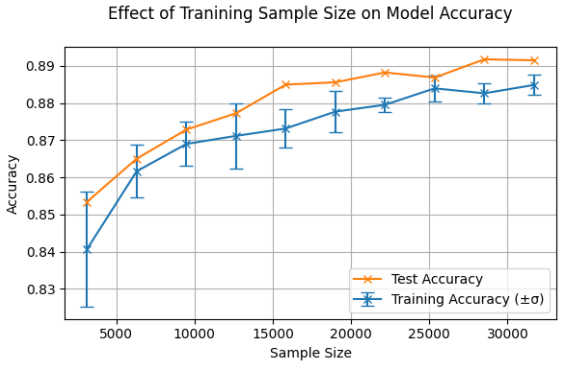
We used Optuna with 25 trials for hyperparameter tuning of the best base model (XGBoost) using various combinations of n_estimators, learning_rate, gamma, subsample, colsample_bytree and max_depth, once again using a 5-fold cross-validation. This process resulted in a model with a slightly better training accuracy (0.887 vs 0.884) but a slightly lower test accuracy (0.894 vs 0.896) compared to the previous best XGBoost model.

Learning Curve Analysis:

The original split had 31,715 training samples and 7,929 test samples. To see the effect of training sample size on the performance of XGBoost models, we ran several tests increasing training sample size from

9.9% to 99.9% of the 80% split and trained an XGBoost model on each of these training sets using a stratified 5-fold cross-validation. The models' mean training scores and test scores were compared to see how increasing the size of the training set affected model performance.

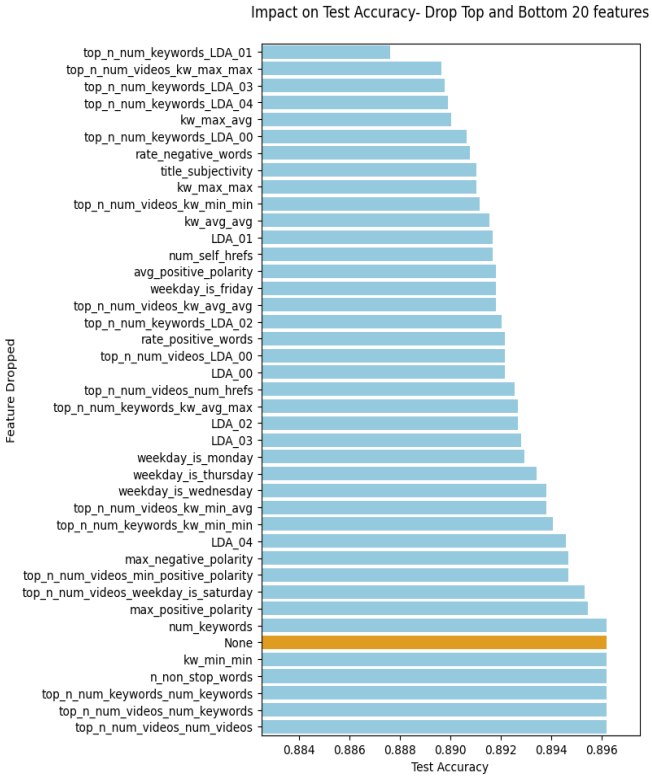
We can see that the mean training score increases with the sample size, beginning to flatten around a training set size of 80% of the original training set size (approximately 25,000 samples in training). The standard deviation of the training scores also decreases until this point and increases very slightly thereafter. The test accuracy increases with training samples, with a slight dip around the 80% mark. The training score is the highest at around the 90% mark and is flat from the 90% to 100% runs. This indicates that the size of the full training dataset is close to optimal for best model performance. The LDA features and the features created by multiplying num_keywords and LDA values figure in the top 20 even at the lower range of the training sample size.



Feature Importance and Ablation Analysis:

From the feature importance contributions for the 92 features used to train the best XGBoost model, we find that a large number of features have a low contribution and a few have large values. As previously mentioned, the LDA features LDA_00, LDA_02, LDA_04, the new features created by multiplying LDA values with num_keywords, num_keywords and kw_min_min are the top 10 features in the best model's feature importance list.

We carried out a round of ablation studies by removing the top and bottom 20 features, one at a time, observing the model accuracy on the test set. The baseline value with all features is 0.896. The chart shows how the test accuracy changes by dropping the top and bottom 20 features, one at a time. The orange bar shows test accuracy for the base model with all 92 features. We can see that there is very little impact of dropping the bottom 5-6 features.



For our second round of ablation studies, we dropped the bottom 6 features as a single group. The top-1 test accuracy in this case was lower at 0.894 compared to 0.896. The drop in top-2 accuracy was very small, at 0.0002, from 0.9798 to 0.9796. If we exclude the bottom 52 contributors (from the feature importance list), we see the top-1 accuracy drop from 0.896 to 0.893 and no significant change in the top-2 accuracy. This indicates some redundant

features that might be reducing the importance of the top contributors, which can be ignored if the lower accuracy is acceptable.

Finally, if we only kept the top 24 features, the top-1 accuracy drops to 0.890, with the top-2 accuracy remaining essentially unchanged. This indicates that a small number of features account for a large portion of the model's ability to identify the article category, but smaller contributions from many features add to the overall model performance.

Performance Tradeoff:
From the ablation analysis, it is evident that we could reduce the number of input features significantly without sacrificing model performance significantly.

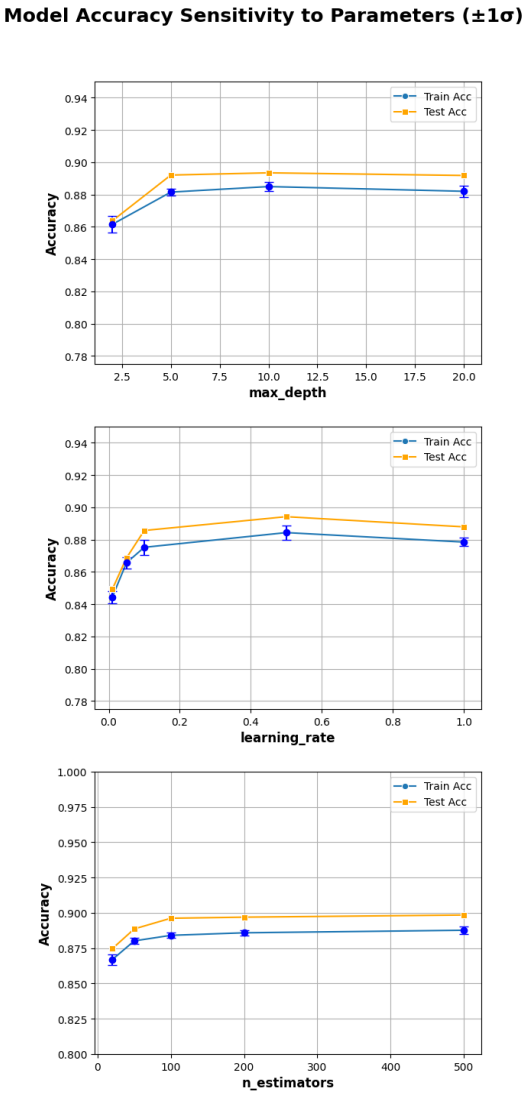
Sensitivity Analysis:

To analyze the model's sensitivity to changes in hyperparameter values, we used the same training set from the previous tests and a similar setup of 5-fold stratified cross-validation to evaluate different combinations of max_depth, learning_rate and n_estimators.

We see that the average training accuracy generally increases with max_depth values and is the highest with the max_depth parameter value of None. The test score is also the highest at this value. Interestingly, there is a slight drop in accuracy when the max_depth increases from 10 to 20 but then jumps to its best value with a parameter value of None (shown as 50 in the first of the three plots).

Model accuracy increases as the learning_rate is increased from 0.01 initially and peaks at 0.25 and then slowly decreases. Increasing the values of parameter n_estimator from 20 to 500 results in a monotonically increasing training and test accuracy scores.

All three parameters show a flat trend in training and test accuracy except at the low end of the range for each parameter indicating model stability over a fairly wide range of parameter values. Setting the learning rate to 0.25 on the previous best model gave us a training accuracy of 0.888 ± 0.003 and a test accuracy of 0.898, a new best model (not found during hyperparameter tuning),



Error analysis:

Our best model (parameters: learning_rate=0.25, n_estimators=500, max_depth=None) has a test accuracy of 0.898 and top-2 accuracy of 0.981. There are a total of 808 misclassifications on the 7,929 samples in the test set. We considered the following three types of errors-
1) **The primary prediction is incorrect but the second best prediction matches the target**

660 of the 808 misclassifications fall into this category. The remaining 148 misclassifications do not show any global patterns (publication month, year, or concentration in a single category).

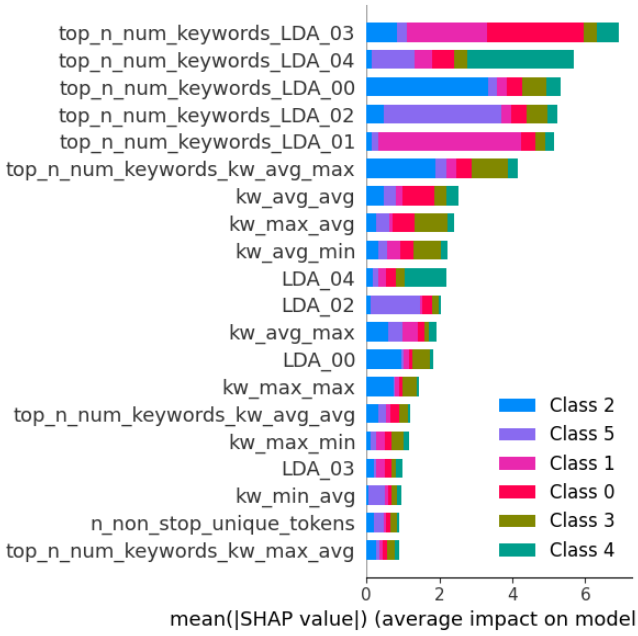
2) **Potentially misleading top LDA feature value in the input data**

An article might have a top LDA value that indicates a certain category but the model finds strong influence of features to indicate a different category.

3) **Incorrect classification by the model**

This could be due to random errors or the combination of feature values causing the model to give an incorrect prediction.

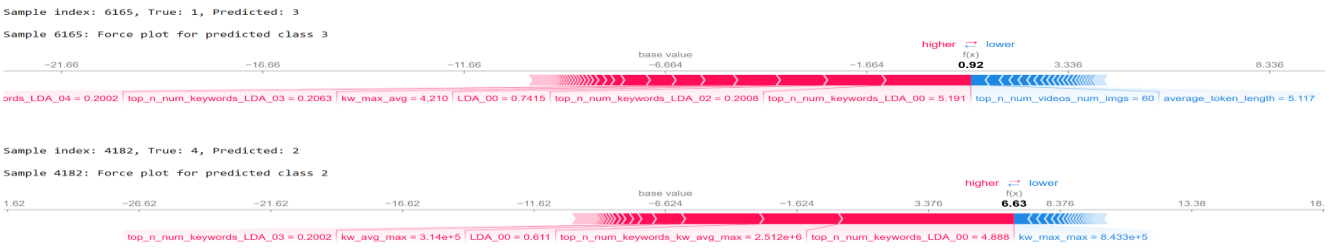
The following SHAP summary plot shows the features and their importance to various categories, which might help explain some of the misclassifications. Each LDA feature seems to have a strong association with a category/class- LDA_04 with Category 4, LDA_00 with Category 2, LDA_02 with Category 5, for example.



Case 1: Second prediction matches target (X_Test Index: 32781)
Target category/channel: 1 Entertainment
Top 2 Predictions and probabilities:
0 (0.527) Lifestyle, 1 (0.473) Entertainment
The probability values of the top 2 predictions are not that far apart and additional feature engineering might help tune the model to recognize the true category better.

Case 2: Potentially misleading LDA values in dataset (Offset 4182, Index 16986)
Target category/channel: 4 Technology
Prediction: 2 (0.999) Business
The article cites a report in Wall Street Journal about a potential Chinese tie-up/deal for Apple. Since our model relies heavily on the LDA values, it is possible that the LDA process incorrectly gave a higher value to the category that is associated with business than technology. In the chart from the potential Improvements section, we can see that Channel 4 is strongly associated with LDA_04 having the highest value (6510 of the 7346 samples). For this sample, the value of LDA_00 is the highest, which is generally associated with Channel/category 2.

Case 3: Incorrect classification by model (Offset 6165, Index 20096)
Target category/channel: 1 Entertainment
Prediction: 3 (0.953) Social media
The article is about HBO releasing a new comedy series, which implies that the prediction is incorrect. The first force plot shows that features created from num_keywords and LDA combinations are pushing the model to predict class 3, with no big contributors in the opposite direction, resulting in a confident but incorrect prediction. The combination of feature values for this sample contribute to the error.



Potential Improvements:

LDA_uci5_channel channel	0	1	2	3	4	All
0	424	661	122	5576	1450	8233
1	117	3632	361	2822	125	7057
2	5298	135	195	110	520	6258
3	1152	98	401	407	265	2323
4	211	127	408	90	6510	7346
5	191	106	7130	285	715	8427
All	7393	4759	8617	9290	9585	39644

Our final model has a fairly high top-1 and top-2 accuracy scores (0.898 and 0.981). The LDA features from the original dataset figured in all the models' most important features. A look at the data shows a clear association between the index corresponding to the highest of the 5 LDA values and the channel/category. The ability to match the original dataset's LDA values will allow us to generalize our model to predict the categories for articles not in the original dataset. We attempted to reengineer the LDA values using a CountVectorizer and scikit-learn's LatentDirichletAllocation using 5 topics, but did not have much success in matching the LDA values from the original dataset. We leave it as a future improvement task that might help address one type of misclassification and improve model performance a little more.

Unsupervised Modeling - Topicality Clustering

Our task in the unsupervised component of this project is to cluster similar articles together through their content. The overview of this workflow includes loading the stored feature embeddings, fitting a k-means model and DBSCAN, evaluating the precision, and plotting the results interactively for us to investigate specific articles. The embeddings we used for this part of the project were 784 dimension embeddings of the article content using a sentence embedder. We experimented with using a title embedding as well, but found the best results to be with the content one so we will focus on that in the experiments below.

Model Selection

K-Means++ and DBSAN were the two methods that we chose to model this problem as they are different algorithms that give us insight into how a density-based method vs a centroid-based model performs in this space. We did not want to choose algorithms/systems that were too similar to each other - like K-Means vs K-Means++ where only the initialization differs, as that would likely not give diverse results for us to have insight into which method would perform better. Put simply, we chose these methods as they operate very differently and will provide diverse results and data points for us to make decisions.

Starting with DBSCAN, a density based model that does not make assumptions about the shape of the clusters and has robust resistance to including outliers in clusters. We chose this method because of its contrast to KMeans where it does not make assumptions of the underlying cluster shapes, and can operate completely unsupervised (does not require us knowing ahead of time the number of clusters we want the algorithm to create). Some of the challenges of DBSCAN include dealing with datasets that could have varying degrees of density and also fine tuning hyperparameters. We will dive into the details of the hyperparameter tuning process for this model in the sensitivity section of evaluation.

The second method we evaluated was K-Means++. This method is a centroid-based method and is very popular in the unsupervised ML space. It expects clusters to be spherical in shape and you must know the number of clusters you desire beforehand (K). We experimented with many different hyperparameter options here, starting with k as 6 and going all the way to 20. Through our subjective evaluation of the clusters and their content through each round, we felt that 6 was a good place to leave it at since that is the number of discrete categories in the dataset. It also resulted in a clustering that yielded the highest precision at k results. Another HP that is often adjusted on this model is the n-inits which attempts to choose points that are far apart from each other for initial centroids. We experimented with numbers ranging from 5 to 50 and it did not seem to make much of a difference in performance after 10. The results of this modeling method were very good, and our intuition tells us that for problems where we know K, this tends to work well. Topics of similar content were co-located thanks to their embeddings being closer in distance to each other, and the intelligent centroid selection of K-Means++ provided adequate cluster separation.

Evaluation

Since we have the rough labels for each article that assigned one of 6 categories, we used that as a starting point to assess the precision of the clustering. We chose to do a majority vote of the true labels in a cluster and assign that entire cluster that label. Since this is an unsupervised space and the labels are very rough, we need to consider that the results here are merely a baseline and show us somewhat how our experiments are going, but they do not truly represent the content of the cluster as we will soon show.

Metric	Description
Coverage	Shows the recall of the clusters - outliers in the DBSCAN are not counted as a cluster for example.
Number of clusters	The total count of identified clusters.
Top-1 Precision	The proportion of the cluster that has the majority cluster label.
Top-2 Precision	The proportion of the cluster that has a label in the top 2 majority classes of the cluster.
Silhouette score	A measure of cohesion and separation - how close points are within the cluster, and how far they are apart from the nearest cluster.
Homogeneity	A measure of consistency within the cluster - how many points are from the same class.
Completeness	Measures the number of total class members that are in the same cluster. For example, a perfect completeness score would be 6 clusters and each one contains 100% of the points for that class.

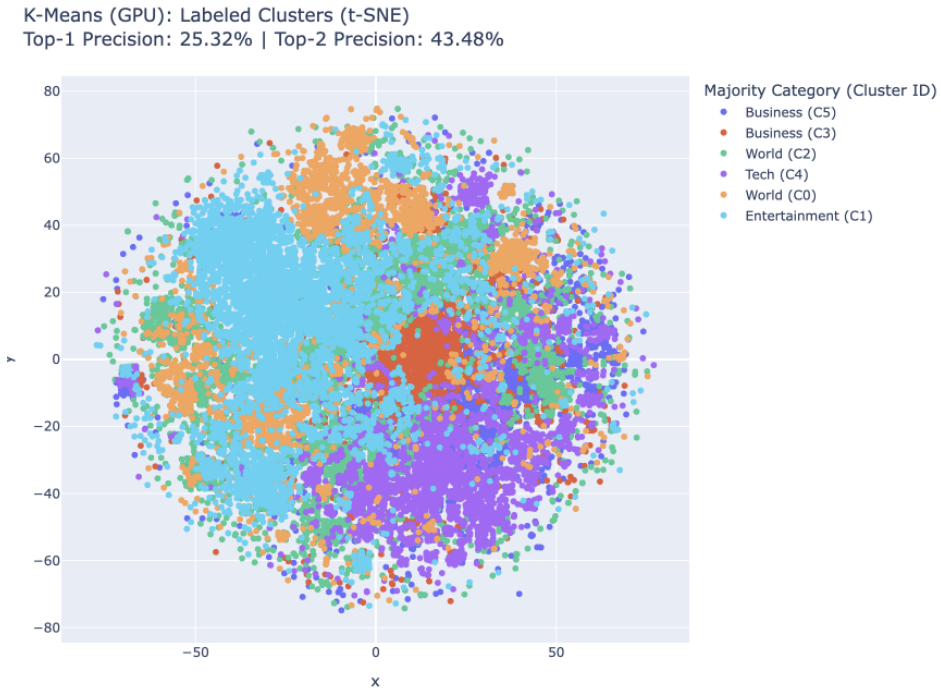
We chose these metrics as it gives us a holistic view into how our clustering systems are performing. Some of these metrics are for recall analysis like coverage, and others show the quality of labels we are predicting. One unique metric we chose to use is precision @ K, where it is often a metric to use in a ranking problem, and in this scenario with our sets of discrete labels, we can also view the problems as a ranking one. It answers the question: Are we creating clusters that have a high probability of assigning points containing the top 2 majority classes. In many areas in the cluster summary, DBSCAN actually produces high quality results - such as precision and completeness. However, the major drawback is on recall, where most data points are classified as noise, and therefore not a cluster. It has just 1/5th the recall of KMeans. With this summary view, it was easy for us to put more support behind our decision with choosing KMeans as the ideal model for this problem.

Performance Overview

🟢 = ideal model

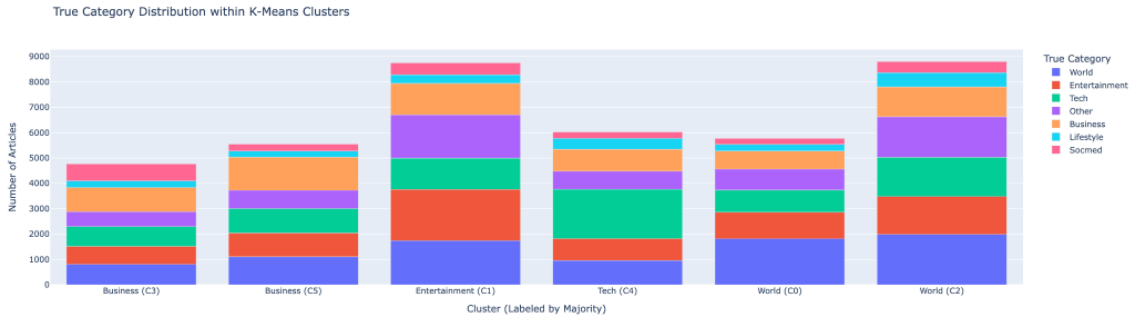
🟡 = underperforming model

Method	Num Clusters	Coverage	Top-1 Precision	Top-2 Precision	Silhouette Score	Homogeneity	Completeness
KMeans++	6	100%	25.32%	43.61%	0.021	0.017	0.017
DBSCAN	11	20%	25.95%	45.93%	0.016	0.01	0.023



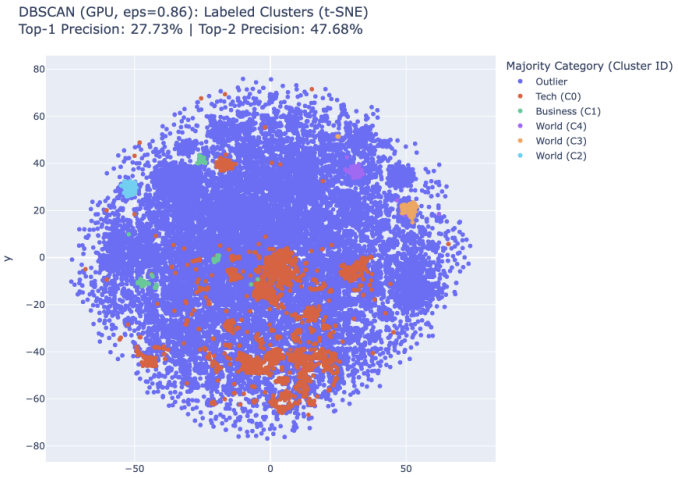
K-Means++

This is the result of the K-Means++ clustering performed. It is an interactive diagram where the coordinates, article title, and majority category in the cluster are displayed when you hover over points. This was the most useful tool in seeing how the model and hyperparameter adjustments were performing. We could quickly make an adjustment and plot the results, hover over a few clusters and get a general idea of what was in each cluster. The second visualization for this method is below where each bar represents a cluster and then the true labels are broken down within each of them. You can see that there are many labeled as other, world, business, and tech that tend to dominate the cluster majority vote. In our opinion the interactive graph is the best way to investigate specific points and view the shapes of your clusters which is crucial in this space.

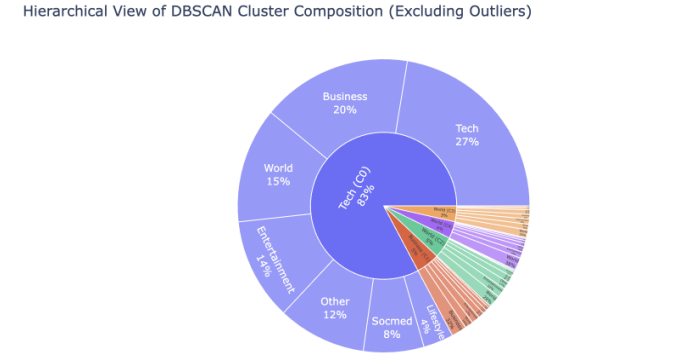


DBSCAN

This method in general created high quality groupings at a significant loss of recall as we noted in the summary section above. The additional visualizations created here include a similar one that was created for Kmeans where we use an interactive scatter plot that displays the article information on hover. This enabled the fine tuning of hyperparameters and its effect on each class.

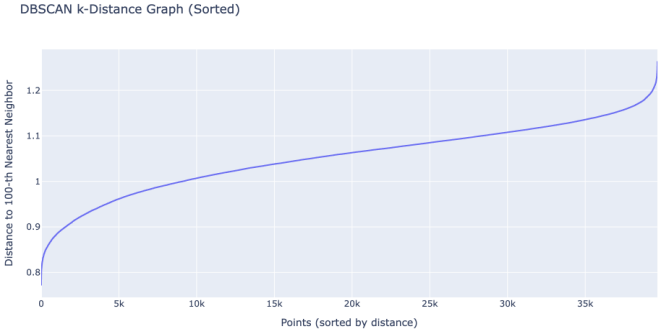


The second visualization used here is a hierarchical view of the cluster composition. When you combine with the chart above, we notice that the tech category is detected well, but we do not know what % it makes up with respect to the other classes. We can make a deduction from this visualization that business, world, and entertainment are classes that we are not performing well in, and further analysis is needed to potentially add additional features, or examine each article through LLMs (See future work ideas), or have human raters add a more granular label.



Sensitivity Analysis

DBSCAN is very sensitive to the epsilon param (abbreviated as eps), and one way that we find the ideal value for this is through the elbow method where we create a chart sorted by distance of nearest neighbors created through the KNN model. We then choose the point in the graph where the slope is the steepest, and that should give us an ideal density clustering parameter that creates clusters while leaving out outliers. An epsilon that is too large will put everything in one cluster, and one that is too small will have most of the dataset being identified as noise. To the right is the chart where you apply the elbow method, and we chose a value of .86 which yielded a top 2 precision of roughly .46. In order to explore the options, we closely examined the slope area and chose values ranging from .5 to 1 and examined the output in the DBSCAN chart with point interactions. When we vary other parameters such as the min samples ones on DBSCAN, we do not see as large of a difference in clustering. Extremes of the min samples param should be avoided as we saw through experimentation, but leaving it fixed at 10 and focusing on the eps parameter yielded the highest performing clusterings.



Discussion

Our supervised learning exploration revealed that predictive power in article share prediction is highly concentrated to just a few features, with keyword and self-reference metrics accounting for over 50% of our model's effectiveness. Additionally, text embeddings for article title and content, despite adding

hundreds of additional dimensions, provided negligible performance improvements compared to simpler structured features. Several findings surprised us in terms of article share prediction, most notably XGBoost underperforming despite its general performance capabilities. We tackled several challenges related to share prediction, such as feature redundancy in our dataset and extreme outliers in share counts. We addressed feature redundancy through regularization techniques and feature importance analysis, and outliers by using log transformations and evaluating models on both log-transformed and non-transformed metrics. With additional time, we could enhance our article share solution through several avenues, such as: temporal modeling to capture cyclical trends, advanced NLP techniques to extract emotional tone, and deep learning architectures like LSTM to identify complex content sharing relationships.

For the category prediction task, we consistently found the 5 LDA features and new features created by combining them with num_features were the top features explaining the model performance. XGBoost emerged as the best model in the initial comparison over several metrics, which was not surprising considering its strength of delivering high accuracy with structured data. Adding article content embeddings and title embeddings to the input features did not help model performance, as the LDA features seem to have captured the articles' theme and show a strong connection to the article category. It is possible for some articles to fall into more than one category. So we also evaluated the top-2 accuracy of the XGBoost model which came in at a very high at 0.981 (the top-1 accuracy was 0.898). This indicates a potential mismatch between the LDA feature values and the categorization of the channel in the original dataset for such samples. Sensitivity analysis shows model performance was fairly flat over a wide range of hyperparameter values, implying good model stability. 92 features were used as input to the model but dropping the bottom half of the feature importance list only reduced the top-1 accuracy by 0.003. So it may be worth the effort to see if a dataset with a smaller number of features can be constructed that will result in a model with almost the same performance as the best current model. Reengineering the LDA feature logic may also help. We could use the same process to add 6 new LDA values to match the number of article categories (instead of the current 5 LDA values) to see if model performance improves.

Unsupervised Discussion

On the unsupervised task of clustering articles by topic, we were very surprised at how accurate K-Means++ was at creating clusters of related content. Exploring the results in the interactive plot show us that articles could actually be under a few different categories depending on how you interpreted them, and the clusters did a really good job at grouping similar content together. For example, some of the clusters that had the wrong label them according to a majority vote of the cluster actually fit in there from a content perspective - one article in tech discussing recycling old iphones, and another right next to it in the clustering space about a celebrity losing an iphone for example that is in the entertainment category.

Some challenges in the unsupervised space were figuring out which combination of features to use for creating the cluster groups. The way we went about solving this is through experimentation and taking into account the problem we were trying to solve. For example, it did not make sense for us to use share counts in the clustering side since we were focused on group articles by topic and not share counts, this limited our search space substantially. We knew that we needed to leverage the content of either the title or the article itself, and the main question was how. Through the experimentation of different embedding combinations, we finally landed on using the content embeddings alone vs taking into account the title. This gave us the result we were looking for where articles of similar topic were closer to each other in the clustering space.

If we had more time to extend the project, we would look into additional methods of creating embeddings of the articles content - fine tuning them for articles can be both very long or less than 300 words. In addition, we would then use an LLM to label the cluster by sampling X amount of articles in the cluster. I believe in this way we could create more accurate results. The topics that are in this UCI dataset are too rough, and we may want to leverage LLMs to identify trends in the cluster space and assign around 20 labels that each article would fit well into. The result of this additional effort would be higher quality clusters, as well as labels that accurately represent the trend.

Ethical Considerations

The Mashable article dataset is relatively “benign” in the sense that it does not contain any sensitive personal data. It consists mostly of numerical features extracted from article content and metadata such as keyword statistics, readability, topic indicators, etc. The only non-numeric field, url, is not used in modeling. However, the following ethical questions are relevant and should be given consideration.

The regression task seeks to model factors associated with article popularity as measured by shares, independent of content quality or accuracy. While this task may have valid applications such as optimizing ad placement or content targeting, its deployment in real-world editorial or content-ranking systems could encourage sensationalism or clickbait. This could have the unintended consequence of penalizing or marginalizing critical content that may not be “popular” according to the model. By making the model descriptive rather than being prescriptive, we can mitigate the negative consequences.

There is a potential for representation bias, if the training samples do not reflect the diversity (topic distribution) of real-world article topics and styles. This could result in biased predictions where unrepresented or underrepresented categories are consistently categorized as unpopular. Care must be taken to ensure that these models are used only in the domains where they have been validated and to always include human oversight.

If interpreted without context, model outputs could be misused to prioritize engagement over informational value, especially in news feeds or recommender systems.Interpretability of model outputs and transparency around model goals can support ethical and responsible use.

Statement of Work

Kyle Ziegler	Venkat Panyam	Chris Roy
● Environment creation ● Web scraping ● Feature engineering ● Unsupervised learning ● Report writing	● Web scraping ● Feature engineering ● Supervised learning (classification models for article category prediction) ● Report writing	● GitHub repo creation and version control ● Feature engineering ● Supervised learning (Regression models for article share prediction) ● Report writing

Appendix A

References

- Fernandes, et al. (2015). A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. 10.1007/978-3-319-23485-4_53.

- Kari, et al. (2016). A Novel Family of Boosted Online Regression Algorithms with Strong Theoretical Bounds. *arXiv: Statistics Theory*

- Tsagkias et al. (2009). Predicting the volume of comments on online news stories. International Journal of Press-politics - INT J PRESS-POLIT. 1765-1768. 10.1145/1645953.1646225.
- For coding only: Google. (2025). *Gemini* (June 25 version) [Large language model]. https://gemini.google.com

Appendix B

Dataset - Attributes and descriptive statistics

Title: Online News Popularity
Source Information

- Creators: Kelwin Fernandes, Pedro Vinagre and Pedro Sernadela
- Donor: Kelwin Fernandes
- Date: May, 2015

Number of Instances: 39,644
Number of Attributes: 61 (58 predictive attributes, 2 non-predictive, 1 target field “shares”)
Attribute Information:

Attr id	Attribute Name	Attribute Description
0	url	URL of the article
1	timedelta	Days between the article publication and the dataset acquisition
2	n_tokens_title	Number of words in the title
3	n_tokens_content	Number of words in the content
4	n_unique_tokens	Rate of unique words in the content
5	n_non_stop_words	Rate of non-stop words in the content
6	n_non_stop_unique_tokens	Rate of unique non-stop words in the content
7	num_hrefs	Number of links
8	num_self_hrefs	Number of links to other articles published by Mashable
9	num_imgs	Number of images
10	num_videos	Number of videos
11	average_token_length	Average length of the words in the content
12	num_keywords	Number of keywords in the metadata
13	data_channel_is_lifestyle	Is data channel 'Lifestyle'?
14	data_channel_is_entertainment	Is data channel 'Entertainment'?
15	data_channel_is_bus	Is data channel 'Business'?
16	data_channel_is_socmed	Is data channel 'Social Media'?
17	data_channel_is_tech	Is data channel 'Tech'?
18	data_channel_is_world	Is data channel 'World'?

19	kw_min_min	Worst keyword (min. shares)
20	kw_max_min	Worst keyword (max. shares)
21	kw_avg_min	Worst keyword (avg. shares)
22	kw_min_max	Best keyword (min. shares)
23	kw_max_max	Best keyword (max. shares)
24	kw_avg_max	Best keyword (avg. shares)
25	kw_min_avg	Avg. keyword (min. shares)
26	kw_max_avg	Avg. keyword (max. shares)
27	kw_avg_avg	Avg. keyword (avg. shares)
28	self_reference_min_shares	Min. shares of referenced articles in Mashable
29	self_reference_max_shares	Max. shares of referenced articles in Mashable
30	self_reference_avg_sharess	Avg. shares of referenced articles in Mashable
31	weekday_is_monday	Was the article published on a Monday?
32	weekday_is_tuesday	Was the article published on a Tuesday?
33	weekday_is_wednesday	Was the article published on a Wednesday?
34	weekday_is_thursday	Was the article published on a Thursday?
35	weekday_is_friday	Was the article published on a Friday?
36	weekday_is_saturday	Was the article published on a Saturday?
37	weekday_is_sunday	Was the article published on a Sunday?
38	is_weekend	Was the article published on the weekend?
39	LDA_00	Closeness to LDA topic 0
40	LDA_01	Closeness to LDA topic 1
41	LDA_02	Closeness to LDA topic 2
42	LDA_03	Closeness to LDA topic 3
43	LDA_04	Closeness to LDA topic 4
44	global_subjectivity	Text subjectivity
45	global_sentiment_polarity	Text sentiment polarity
46	global_rate_positive_words	Rate of positive words in the content

47	global_rate_negative_words	Rate of negative words in the content
48	rate_positive_words	Rate of positive words among non-neutral tokens
49	rate_negative_words	Rate of negative words among non-neutral tokens
50	avg_positive_polarity	Avg. polarity of positive words
51	min_positive_polarity	Min. polarity of positive words
52	max_positive_polarity	Max. polarity of positive words
53	avg_negative_polarity	Avg. polarity of negative words
54	min_negative_polarity	Min. polarity of negative words
55	max_negative_polarity	Max. polarity of negative words
56	title_subjectivity	Title subjectivity
57	title_sentiment_polarity	Title polarity
58	abs_title_subjectivity	Absolute subjectivity level
59	abs_title_sentiment_polarity	Absolute polarity level
60	shares	Number of shares (target)

Missing Attribute Values: None
Summary Statistics:

Feature	**Min**	**Max**	**Mean**	**SD**
timedelta	8.0000	731.0000	354.5305	214.1611
n_tokens_title	2.0000	23.0000	10.3987	2.1140
n_tokens_content	0.0000	8474.0000	546.5147	471.1016
n_unique_tokens	0.0000	701.0000	0.5482	3.5207
n_non_stop_words	0.0000	1042.0000	0.9965	5.2312
n_non_stop_unique_tokens	0.0000	650.0000	0.6892	3.2648
num_hrefs	0.0000	304.0000	10.8837	11.3319
num_self_hrefs	0.0000	116.0000	3.2936	3.8551
num_imgs	0.0000	128.0000	4.5441	8.3093
num_videos	0.0000	91.0000	1.2499	4.1078
average_token_length	0.0000	8.0415	4.5482	0.8444

num_keywords	1.0000	10.0000	7.2238	1.9091
data_channel_is_lifestyle	0.0000	1.0000	0.0529	0.2239
data_channel_is_entertainment	0.0000	1.0000	0.1780	0.3825
data_channel_is_bus	0.0000	1.0000	0.1579	0.3646
data_channel_is_socmed	0.0000	1.0000	0.0586	0.2349
data_channel_is_tech	0.0000	1.0000	0.1853	0.3885
data_channel_is_world	0.0000	1.0000	0.2126	0.4091
kw_min_min	-1.0000	377.0000	26.1068	69.6323
kw_max_min	0.0000	298400.0000	1153.9517	3857.9422
kw_avg_min	-1.0000	42827.8571	312.3670	620.7761
kw_min_max	0.0000	843300.0000	13612.3541	57985.2980
kw_max_max	0.0000	843300.0000	752324.0667	214499.4242
kw_avg_max	0.0000	843300.0000	259281.9381	135100.5433
kw_min_avg	-1.0000	3613.0398	1117.1466	1137.4426
kw_max_avg	0.0000	298400.0000	5657.2112	6098.7950
kw_avg_avg	0.0000	43567.6599	3135.8586	1318.1338
self_reference_min_shares	0.0000	843300.0000	3998.7554	19738.4216
self_reference_max_shares	0.0000	843300.0000	10329.2127	41027.0592
self_reference_avg_sharess	0.0000	843300.0000	6401.6976	24211.0269
weekday_is_monday	0.0000	1.0000	0.1680	0.3739
weekday_is_tuesday	0.0000	1.0000	0.1864	0.3894
weekday_is_wednesday	0.0000	1.0000	0.1875	0.3903
weekday_is_thursday	0.0000	1.0000	0.1833	0.3869
weekday_is_friday	0.0000	1.0000	0.1438	0.3509
weekday_is_saturday	0.0000	1.0000	0.0619	0.2409
weekday_is_sunday	0.0000	1.0000	0.0690	0.2535
is_weekend	0.0000	1.0000	0.1309	0.3373
LDA_00	0.0000	0.9270	0.1846	0.2630

LDA_01	0.0000	0.9259	0.1413	0.2197
LDA_02	0.0000	0.9200	0.2163	0.2821
LDA_03	0.0000	0.9265	0.2238	0.2952
LDA_04	0.0000	0.9272	0.2340	0.2892
global_subjectivity	0.0000	1.0000	0.4434	0.1167
global_sentiment_polarity	-0.3937	0.7278	0.1193	0.0969
global_rate_positive_words	0.0000	0.1555	0.0396	0.0174
global_rate_negative_words	0.0000	0.1849	0.0166	0.0108
rate_positive_words	0.0000	1.0000	0.6822	0.1902
rate_negative_words	0.0000	1.0000	0.2879	0.1562
avg_positive_polarity	0.0000	1.0000	0.3538	0.1045
min_positive_polarity	0.0000	1.0000	0.0954	0.0713
max_positive_polarity	0.0000	1.0000	0.7567	0.2478
avg_negative_polarity	-1.0000	0.0000	-0.2595	0.1277
min_negative_polarity	-1.0000	0.0000	-0.5219	0.2903
max_negative_polarity	-1.0000	0.0000	-0.1075	0.0954
title_subjectivity	0.0000	1.0000	0.2824	0.3242
title_sentiment_polarity	-1.0000	1.0000	0.0714	0.2654
abs_title_subjectivity	0.0000	0.5000	0.3418	0.1888
abs_title_sentiment_polarity	0.0000	1.0000	0.1561	0.2263

Appendix C

Consolidated List of Features

Final features used in regression analysis (predict log transformed “shares” column):

1. num_hrefs
2. num_imgs
3. num_videos
4. average_token_length
5. num_keywords
6. data_channel_is_lifestyle
7. data_channel_is_entertainment
8. data_channel_is_bus
9. data_channel_is_socmed
10. data_channel_is_tech
11. data_channel_is_world
12. kw_max_min
13. kw_avg_min
14. kw_avg_max
15. kw_min_avg
16. kw_max_avg
17. kw_avg_avg
18. self_reference_min_shares
19. self_reference_max_shares
20. self_reference_avg_sharess
21. weekday_is_tuesday
22. weekday_is_wednesday
23. weekday_is_saturday
24. weekday_is_sunday
25. is_weekend
26. LDA_00
27. LDA_01
28. LDA_02
29. LDA_03
30. LDA_04
31. global_subjectivity
32. global_sentiment_polarity
33. global_rate_positive_words
34. rate_negative_words
35. avg_positive_polarity
36. max_positive_polarity
37. avg_negative_polarity
38. title_subjectivity
39. title_sentiment_polarity
40. abs_title_sentiment_polarity

Final features used in category classification:

1. N_tokens_title (**The first 52 columns are original features from the dataset**)
2. n_tokens_content
3. n_unique_tokens
4. n_non_stop_words
5. n_non_stop_unique_tokens
6. num_hrefs
7. num_self_hrefs
8. num_imgs
9. num_videos
10. average_token_length
11. num_keywords
12. kw_min_min
13. kw_max_min
14. kw_avg_min
15. kw_min_max
16. kw_max_max
17. kw_avg_max
18. kw_min_avg
19. kw_max_avg
20. kw_avg_avg
21. self_reference_min_shares
22. self_reference_max_shares
23. self_reference_avg_sharess
24. weekday_is_monday
25. weekday_is_tuesday
26. weekday_is_wednesday
27. weekday_is_thursday
28. weekday_is_friday
29. weekday_is_saturday
30. weekday_is_sunday
31. is_weekend
32. LDA_00
33. LDA_01
34. LDA_02
35. LDA_03
36. LDA_04
37. global_subjectivity
38. global_sentiment_polarity
39. global_rate_positive_words
40. global_rate_negative_words
41. rate_positive_words
42. rate_negative_words
43. avg_positive_polarity
44. min_positive_polarity
45. max_positive_polarity
46. avg_negative_polarity
47. min_negative_polarity

48. max_negative_polarity
49. title_subjectivity
50. title_sentiment_polarity
51. abs_title_subjectivity
52. abs_title_sentiment_polarity
53. top_n_num_keywords_LDA_02
    (**The next 20 columns are new features created by multiplying num_keywords and the top 20 features from the base XGBoost model**)
54. top_n_num_keywords_LDA_04
55. top_n_num_keywords_LDA_00
56. top_n_num_keywords_LDA_03
57. top_n_num_keywords_LDA_01
58. top_n_num_keywords_kw_max_max
59. top_n_num_keywords_kw_min_min
60. top_n_num_keywords_kw_avg_avg
61. top_n_num_keywords_num_keywords
62. top_n_num_keywords_kw_max_avg
63. top_n_num_keywords_kw_avg_max
64. top_n_num_keywords_num_self_hrefs
65. top_n_num_keywords_kw_min_avg
66. top_n_num_keywords_min_positive_polarity
67. top_n_num_keywords_num_videos
68. top_n_num_keywords_kw_min_max
69. top_n_num_keywords_n_tokens_content
70. top_n_num_keywords_num_imgs
71. top_n_num_keywords_num_hrefs
72. top_n_num_keywords_weekday_is_saturday
73. top_n_num_videos_LDA_02
    (**The next 20 columns are new features created by multiplying num_videos and the top 20 features from the base XGBoost model**)
74. top_n_num_videos_LDA_04
75. top_n_num_videos_LDA_00
76. top_n_num_videos_LDA_03
77. top_n_num_videos_LDA_01
78. top_n_num_videos_kw_max_max
79. top_n_num_videos_kw_min_min
80. top_n_num_videos_kw_avg_avg
81. top_n_num_videos_num_keywords
82. top_n_num_videos_kw_max_avg
83. top_n_num_videos_kw_avg_max
84. top_n_num_videos_num_self_hrefs
85. top_n_num_videos_kw_min_avg
86. top_n_num_videos_min_positive_polarity
87. top_n_num_videos_num_videos
88. top_n_num_videos_kw_min_max
89. top_n_num_videos_n_tokens_content
90. top_n_num_videos_num_imgs
91. top_n_num_videos_num_hrefs
92. top_n_num_videos_weekday_is_saturday

93. channel (**target variable created from the 6 data_channel_is_ features**)