News Popularity Prediction
Milestone: Draft of Project Report

Group 7

Student 1 Pooja Laxmi Sankarakameswaran
Student 2 Shaun Kirthan

sankarakameswaran.p@northeastern.edu
lnu.shau@northeastern.edu

Percentage of Effort Contributed by Student 1:_______50________

Percentage of Effort Contributed by Student 2:_______50________

Signature of Student 1: [illegible]

Signature of Student 2: [illegible]

Submission Date:____________22-03-2024_____________________

Problem Setting

In the digital era, news virality is crucial, rapidly shaping public discourse through user engagement and social sharing. Predicting online news popularity empowers Mashable for strategic content optimization and enhanced social media engagement, maximizing audience reach and revenue. Conversely, the absence of prediction hampers Mashable's competitiveness, risking suboptimal engagement and inefficient resource allocation. The predictive capability becomes a strategic asset for proactive measures, optimizing content delivery and user interactions. In its absence, Mashable faces challenges in adapting to emerging trends, hindering overall effectiveness in the dynamic digital media landscape.

Problem Definition

The problem aims to predict the popularity of articles published by Mashable over a two-year period based on a diverse set of features. The dataset includes various statistics associated with the articles, and the primary objective is to build a predictive model that can estimate the number of shares an article is likely to receive on social networks. This predictive task is essential for understanding the factors influencing the online popularity of articles, providing insights for content creators and publishers to optimize their strategies for increased social media engagement.

Data Sources

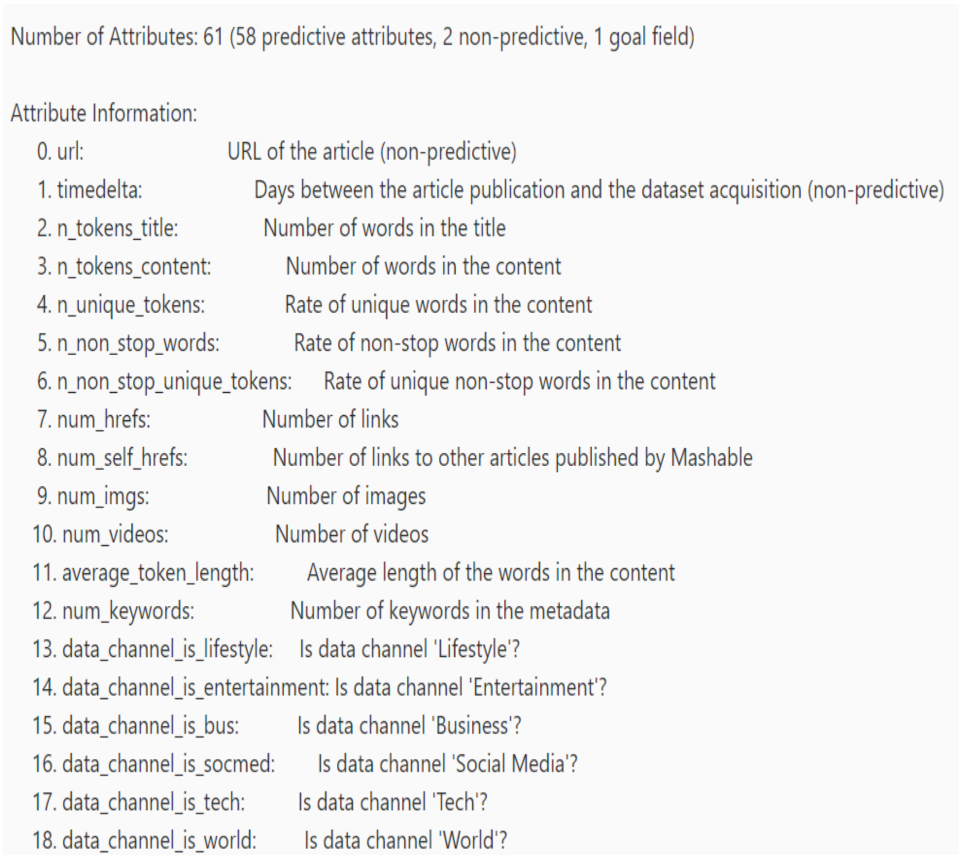
The dataset is hosted on the UCI Machine Learning Repository, a well-known repository for machine learning datasets. The specific reference to the dataset on the UCI website is Online News Popularity Dataset. The dataset contains statistics and features related to articles published by Mashable, a news and entertainment website (www.mashable.com).

Data Description

The dataset contains 39797 instances. Each of these instances have 61 variables in total, 58 of which are predictive and 2 are non-predictive and 1 variable is the target / response / goal variable. The target variable is **"shares"**, which is a numerical variable describing the number of times each online article was shared. An abridged version of the description is mentioned below with the variables grouped and the detailed description can be found in figures below.

<u>Groups of Variables:</u>

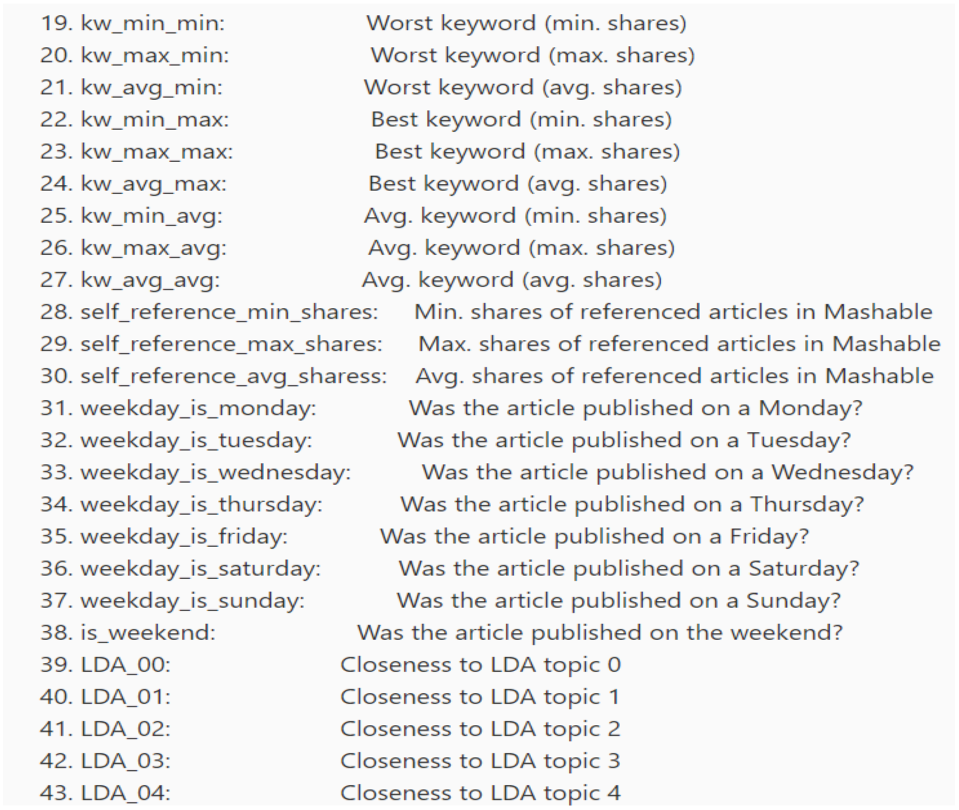
- Non predictive variables
- Number of words in categories
- Number of attachments
- Average article length
- Data Channel
- Keyword statistics
- Self Reference statistics
- Day of Publication
- LDA Details
- Global Effect Rates
- Polarity statistics
- Title and Absolute Title polarity



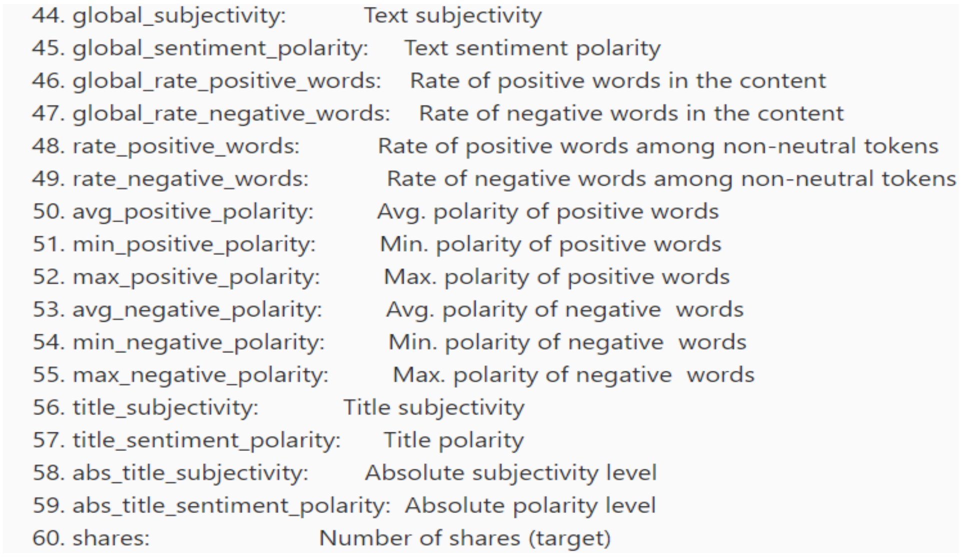
Number of Attributes: 61 (58 predictive attributes, 2 non-predictive, 1 goal field)

Attribute Information:

0. url:	URL of the article (non-predictive)
1. timedelta:	Days between the article publication and the dataset acquisition (non-predictive)
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)

<u>Data Mining Tasks</u>

a. Data Understanding

The dataset contains 39797 instances. Each of these instances have 61 variables in total, 58 of which are predictive and 2 are non-predictive and 1 variable is the target /

response / goal variable. The target variable is **"shares"**, which is a numerical variable describing the number of times each online article was shared.

b. Data Pre-processing

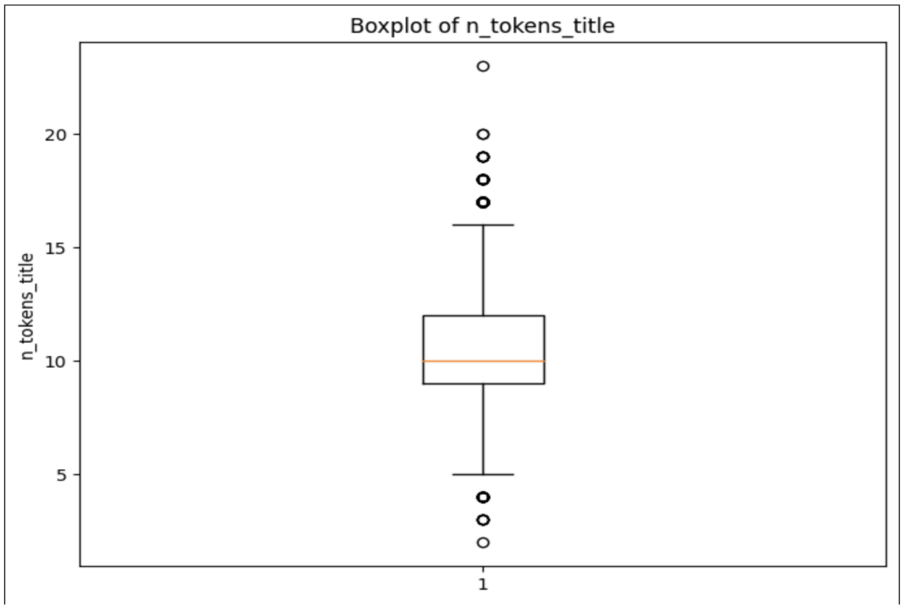
The dataset was initially free of any null or missing values, providing a solid foundation for examining variables and deriving insights. However, each variable contained extraneous spaces at the beginning or end, which have since been eliminated. We also identified and removed columns that would not contribute to the predictive model's development. Notably, all variables were initially categorized as float types, leading to numerous outliers. Upon closer examination, we identified variables that exclusively contained integer values and subsequently converted these variables to integer types to address this issue.Some of the conversions are mentioned below,

n_tokens_title	int64
n_tokens_content	int64
n_unique_tokens	float64
n_non_stop_words	float64
n_non_stop_unique_tokens	float64

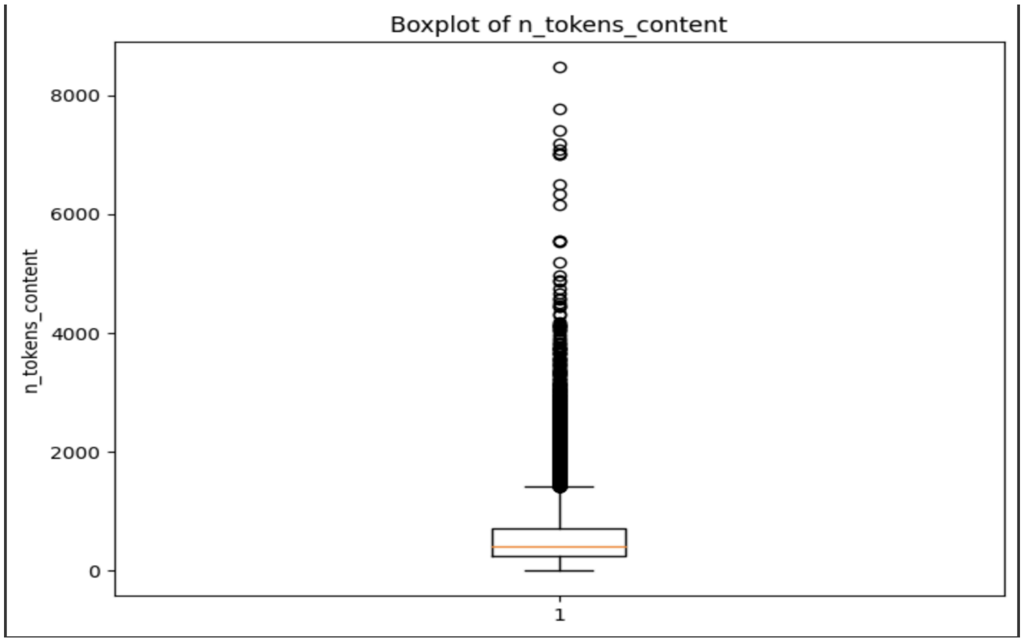
c. Data Exploration

The numeric and categorical variables were examined separately using exploratory data analysis. The observations from these evaluations are listed below

The boxplot displayed shows the distribution of the number of tokens (words or terms) in the titles of a dataset. The median number of tokens in titles is around 7, indicated by the line within the box. The interquartile range (IQR) is tight, suggesting that half of the titles have token counts close to the median. There are several outliers, indicating some titles have significantly more tokens than typical. The range of token counts extends from 0 to above 20, but most titles have fewer than 15 tokens. The boxplot's lack of lower whisker hints at a minimum token count close to the lower quartile, suggesting few or no titles with very few tokens.

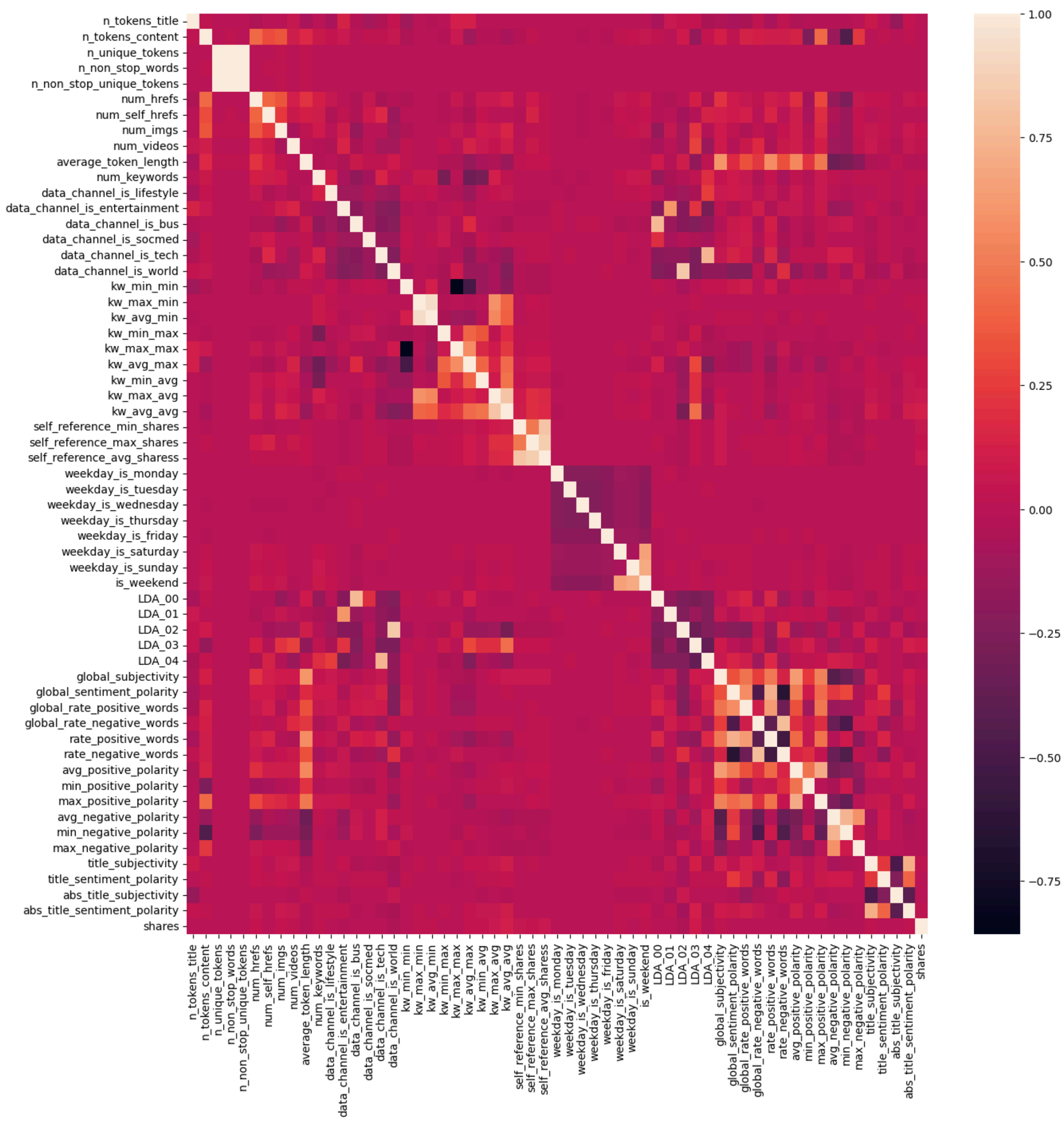


The box plot illustrates the distribution of the number of tokens (words or terms) in the content of a dataset. The median token count in content is low relative to the scale, suggesting that a typical entry has a modest number of tokens. There is a large range of token counts, with the upper quartile much higher than the median, indicating variability in content length. The data contains several outliers, where the content has a significantly higher number of tokens than the general trend. The box plot shows a large number of outliers above the upper whisker, which means there are many entries with exceptionally long content. The interquartile range is relatively small compared to the overall range, indicating that a majority of the content has a number of tokens closer to the median.

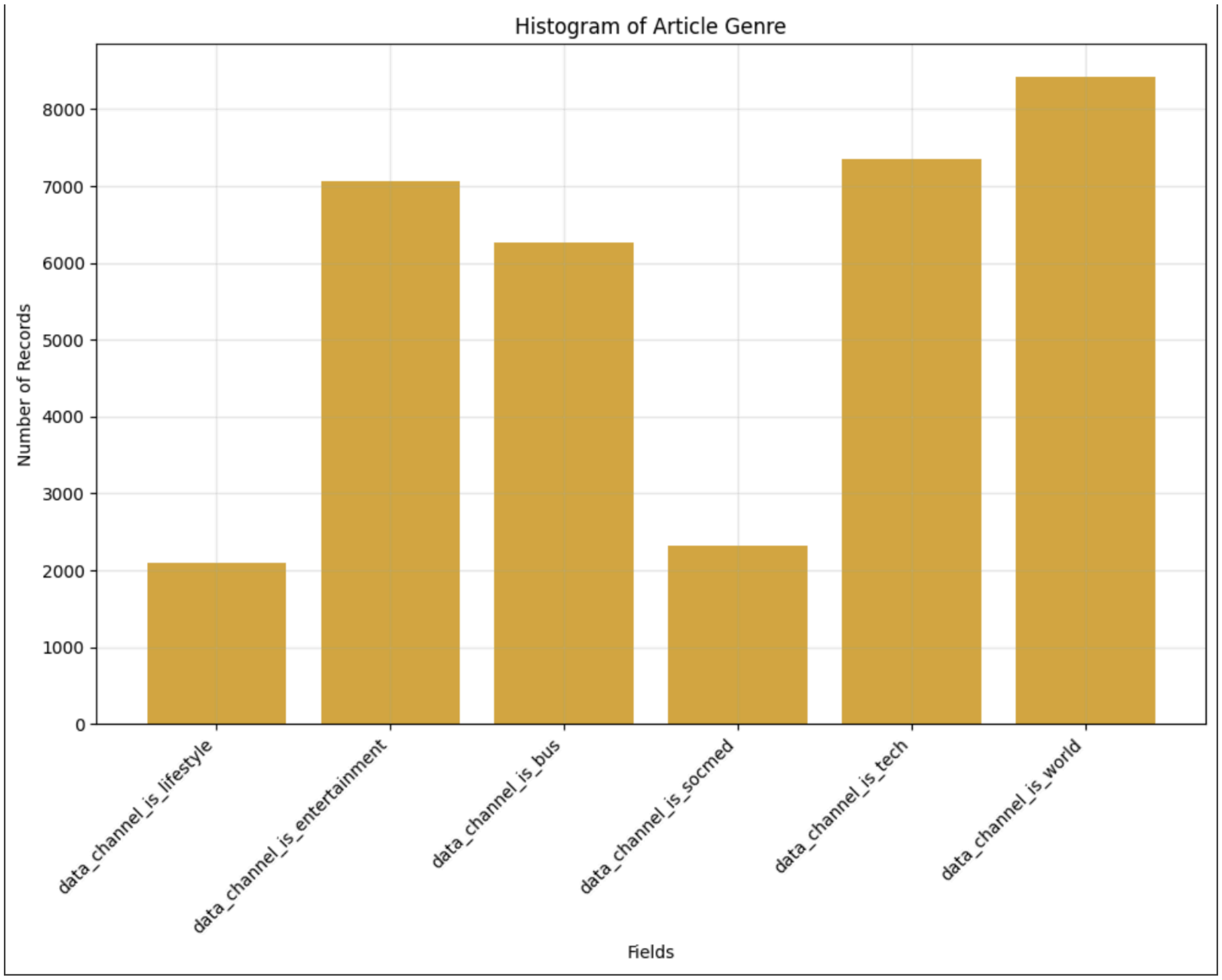


d. Correlation Matrix - Heatmap & Kernel Density Estimate

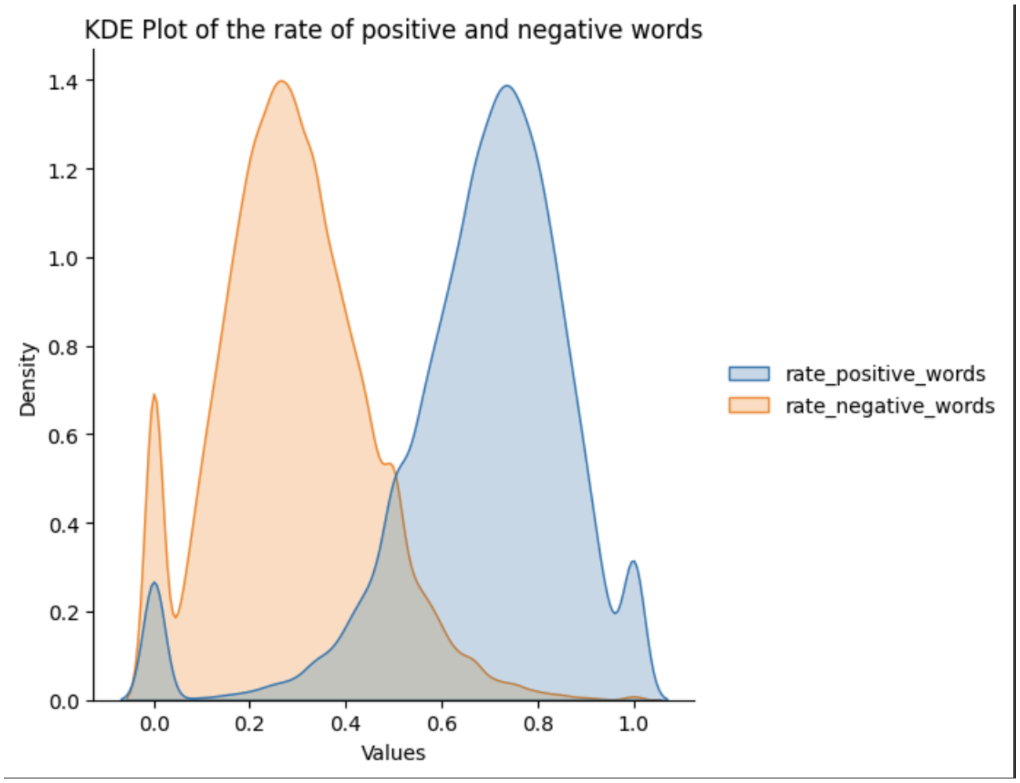
The correlation matrix is represented by the heatmap shown above. The matrix includes all 61 variables in the dataset and helps detect some of the most similar or linearly correlated variables, which are handled in the sections to follow.



The graph is a histogram that represents the distribution of records across different article genres or categories within a dataset. Each bar corresponds to a different article genre, such as lifestyle, entertainment, business, social media, tech, and world news. The height of each bar indicates the number of records (or articles) that fall into each genre. The 'tech' genre has the highest number of records, followed by 'world', indicating these are the most prevalent categories in the dataset. The 'lifestyle' category has the fewest records, suggesting it is less common or less frequently reported on in this particular dataset. The x-axis labels are rotated 45 degrees to fit the longer category names and make them readable.



The graph is a Kernel Density Estimate (KDE) plot that visualizes the distribution of rates of positive and negative words in a dataset. The x-axis represents the values of the rates, ranging from 0 to 1, which likely indicates the proportion of positive or negative words in the text. The y-axis shows the density of these rates, which gives an idea of how common certain rates are within the dataset. There are two distributions plotted: one for positive words (in blue) and one for negative words (in orange). The peaks of the distributions indicate the most common rates for positive and negative words. Both distributions show that certain rates are more common than others. The areas where the distributions have lower height suggest that there are fewer articles with those particular rates of positive or negative words.



e. Outlier detection and handling

All of the fields in the dataset contained outliers, however, some of them contained more than 1/4th of the total number of observations. The details regarding the first 5 variables can be found below.

Variable: n_tokens_title, Number of outliers: 415
First five outliers: [16, 16, 16, 18, 4]
Variable: n_tokens_content, Number of outliers: 1802
First five outliers: [1455, 1596, 1821, 1579, 1550]
Variable: n_unique_tokens, Number of outliers: 1264
First five outliers: [0.954545411157, 0.0, 0.0, 0.0, 0.0]
Variable: n_non_stop_words, Number of outliers: 2454
First five outliers: [0.999999979592, 0.999999983871, 0.999999982456, 0.99999998, 0.999999983607]
Variable: n_non_stop_unique_tokens, Number of outliers: 1344
First five outliers: [0.341637010271, 0.0, 0.0, 0.0, 0.0]
After the

For the outlier detection, modified z-score was used which utilizes the mean absolute deviation and if the score is over 3.5 it computes the values with the median (statistical measure used to impute the outliers). After outlier handling the details can be found as shown below,

```
Variable: n_tokens_title, Number of outliers: 0
First five outliers: []
Variable: n_tokens_content, Number of outliers: 0
First five outliers: []
Variable: n_unique_tokens, Number of outliers: 0
First five outliers: []
Variable: num_hrefs, Number of outliers: 0
First five outliers: []
Variable: num_self_hrefs, Number of outliers: 0
First five outliers: []
```

f. Feature Selection and extraction

a. Usage of Correlation Matrix

In the feature selection part of the code, columns from a dataset are dropped based on their intercorrelations to reduce redundancy and potentially improve the performance of subsequent machine learning models. First, we create a correlation matrix which we then convert into a DataFrame for easier manipulation. The code identifies pairs of features with a high absolute correlation (either above 0.9 or below -0.9) and then selects one variable from each highly correlated pair to be dropped, as indicated by the `columns_to_drop` assignment. Finally, these selected columns are dropped from the main DataFrame. Effectively reducing the feature space and possibly mitigating issues like multicollinearity. And we end up with 56 variables.

b. Recursive Feature Selection Elimination

In this process, logistic regression is employed as the wrapper model within Recursive Feature Elimination (RFE), while the mutual information metric and statistical significance tests, including the F-test for regression (f_regression), are utilized to identify and select features that demonstrate the highest relevance to the target variable. Mutual information quantifies the dependency between variables, offering insights into the strength and direction of the relationship between each feature and the target. Conversely, statistical significance tests, such as the F-test, assess the probability that the observed associations between the features and the target are not solely due to random chance. This dual approach ensures a robust feature selection mechanism. The `select_features_significance` function encapsulates this methodology, leveraging both mutual information and significance testing to sift through variables. It filters out the most predictive features by evaluating their individual contributions to the target variable's variance, effectively combining the distinct sets of variables highlighted by each test into a comprehensive and optimized feature subset.

Final Feature List:

n_tokens_content
num_hrefs
num_self_hrefs
num_imgs
num_videos
num_keywords
data_channel_is_entertainment
data_channel_is_tech
data_channel_is_world
kw_min_min
kw_max_max
kw_avg_max
kw_min_avg
kw_max_avg
kw_avg_avg
self_reference_min_shares
weekday_is_tuesday
weekday_is_wednesday
weekday_is_thursday
weekday_is_friday
weekday_is_saturday
LDA_02
LDA_03
global_subjectivity
global_sentiment_polarity
min_positive_polarity
min_negative_polarity

Data Preparation

We demonstrate the process of feature standardization in a machine learning workflow, specifically preparing data for models that are sensitive to the scale of input variables, such as logistic regression, support vector machines, and neural networks. The `StandardScaler` from a machine learning library (likely scikit-learn) is used to standardize features by removing the mean and scaling to unit variance. This process ensures that each feature contributes approximately equally to the final prediction, which is crucial for models that rely on gradient descent optimization as it can help in achieving faster convergence. In the code, `X_train` is first fitted and transformed by the scaler, establishing the scaling parameters (mean and standard deviation) based on the training data. These same parameters are then used to transform the validation set `X_val' and test set `X_test` to ensure consistency in scaling across all data splits. This step is vital for evaluating the model's performance accurately, as it simulates the application of the model to new, unseen data while maintaining the scale adjustments learned from the training data.

Data Partitioning

For data partitioning we illustrate the process of dividing a dataset into training, validation, and test sets, a common practice in machine learning to evaluate the performance of models. Initially, the `train_test_split` function is used to separate the dataset (`X` for features, `y` for labels) into two groups: training plus validation (80% of the data) and test (20% of the data), ensuring a representative distribution of data across sets via the `random_state` parameter for reproducibility. The selected 20% test size is a conventional choice, allowing sufficient data for training and validation while reserving enough examples for an unbiased evaluation of the model's performance. Subsequently, the training and validation set combined `X_train_val`, `y_train_val` undergoes a further split. This time, approximately 22.2% of it is allocated to the validation set, and the remainder to the training set. This secondary split ensures that 20% of the original dataset is used for validation (since 22.2% of the remaining 80% equals 20% of the total dataset), maintaining a balance in data distribution for effective training and validation. This approach allows for the comprehensive training of the model, fine-tuning of hyperparameters on the validation set, and a final, unbiased evaluation on the test set.

Data Mining Models/Methods

1. Logistic Regression

Logistic Regression is a predictive modeling technique used for classification problems. It is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead, or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. The logistic regression model uses one or more independent variables that may be either continuous or categorical to predict a dependent variable that is binary. The output of a logistic regression model is transformed using the logistic function to ensure that the output lies between 0 and 1, making it interpretable as a probability. This transformation is what differentiates logistic regression from linear regression and makes it suitable for binary classification problems.

<u>Advantages:</u>

- Logistic regression provides probabilities for outcomes, which can be useful for understanding the certainty of predictions and for making decisions that take into account the likelihood of different outcomes.

- It is relatively simple and efficient to implement, interpret, and train, making it a good baseline algorithm for binary classification problems.

<u>Disadvantages:</u>

- Logistic Regression may struggle to effectively model complex relationships between variables, especially when there are nonlinear interactions that significantly influence the outcome.

- It is sensitive to imbalance in the dataset, where the numbers of observations in different classes are significantly different, which can lead to biased predictions favoring the majority class.

<u>Implementation:</u>

In this implementation, the Logistic Regression model is trained on a preprocessed dataset. The `StandardScaler` is employed to normalize the features, ensuring uniform contribution of each feature to the model's decision-making process. The dataset is divided into training and test sets, with 80% of the data allocated for training purposes and 20% reserved for testing. Following the training phase, the model makes predictions on the test set, and its performance is evaluated using accuracy, precision, recall, and the F1 score. The accuracy metric in this implementation highlights the percentage of correctly predicted instances out of all predictions made, offering insight into the overall effectiveness of the model in classifying the dependent variable based on the independent variables.

2. <u>Decision Trees</u>

A Decision Tree Regression model is a type of decision tree that is used for predicting continuous values. The model involves dividing the dataset into distinct subsets based on certain conditions, aiming to predict the value of the target variable within each subset. Decision trees split the data into branches based on feature values, with the goal of reducing variance or bias in the predictions for each branch, leading to a tree-like structure of decisions and outcomes. This structure allows for intuitive and interpretable modeling of complex relationships between the input features and the continuous target variable.

Advantages:

-Neural networks excel in handling and learning from large volumes of data, making them highly effective for complex pattern recognition and prediction tasks across diverse domains such as image and speech recognition, natural language processing, and gaming.

-They have the ability to learn and model non-linear and complex relationships between inputs and outputs, which allows for more accurate predictions and analyses compared to traditional linear models.

Disadvantages:

- Neural networks require a large amount of data to train effectively, making them less suitable for tasks with limited data.

- They are often considered "black boxes" due to their complex internal architectures, making it difficult to understand how decisions are made.

Implementation:

In this implementation, the Decision Tree model is trained on a preprocessed dataset. The `StandardScaler` is utilized to normalize the features, ensuring that each feature contributes equally to the decision-making process of the model. The dataset is partitioned into training and test sets, with 80% of the data used for training and 20% set aside for testing. After the model is trained, it makes predictions on the test set, and its performance is assessed using metrics such as accuracy, precision, recall, and the F1 score. The accuracy metric in this context signifies the proportion of correctly predicted instances to the total predictions made, providing a measure of the model's overall ability to correctly classify the dependent variable based on the independent variables.

3. kNN Classifier

k-Nearest Neighbors (kNN) Classifier is a simple and intuitive machine learning algorithm used for classification and regression tasks. It works by finding the k nearest data points to a given query point and then classifying the query point based on the majority class or averaging the values of its nearest neighbors.

<u>Advantages:</u>

- KNN is easy to understand and implement, making it a good choice for beginners and as a baseline algorithm for comparison.
- Unlike many other algorithms, kNN does not require explicit training on the dataset. The prediction process involves only storing the data points, making it efficient for online learning scenarios.

- KNN is a non-parametric algorithm, meaning it makes no assumptions about the underlying data distribution. It can perform well with complex and nonlinear decision boundaries.
- KNN can handle multi-class classification problems and is versatile for both classification and regression tasks.

<u>Disadvantages:</u>

- The main drawback of kNN is its computational cost during prediction, especially with large datasets. As the number of data points increases, the search for nearest neighbors becomes more time-consuming.
- Storing the entire dataset in memory can be memory-intensive, particularly for large datasets.

- KNN relies on distance metrics to find nearest neighbors. If the features have different scales, features with larger scales may dominate the distance calculations, leading to biased results.

The choice of the parameter k (number of neighbors) can significantly impact the performance of the kNN algorithm. Selecting an appropriate value for k is crucial and often requires experimentation or cross-validation.

Overall, while kNN is simple and intuitive, its performance can vary depending on the dataset size, feature scaling, and choice of k. It is often used for small to medium-sized datasets or as a component of more complex machine learning pipelines.

<u>Implementation:</u>

The implementation involves splitting the dataset into training, validation, and test sets using `train_test_split()`. The features are standardized using `StandardScaler`. Then, a kNN classifier with 3 neighbors is initialized and trained on the standardized training data. Predictions are made on the validation set, and the model is evaluated using accuracy metrics. Subsequently, predictions are made on the test set, and the model is evaluated again for accuracy. Additionally, error metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score are calculated to assess the model's performance. The kNN classifier achieves low accuracy on both the validation and test sets, suggesting poor predictive capability. The MSE and RMSE values are high, indicating significant errors in prediction, while the negative R-squared score indicates poor model fit to the data.

4. <u>Random Forest Classifier</u>

Random Forest Classifier is an ensemble learning method that combines multiple decision trees to create a powerful predictive model. It operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

<u>Advantages:</u>

- Random Forests typically achieve high accuracy in classification and regression tasks. They are robust against overfitting, especially when using a large number of trees.

- Random Forests can handle a variety of data types, including numerical and categorical features, without requiring extensive preprocessing. They are also effective for both classification and regression tasks.
- Random Forests provide a measure of feature importance, allowing users to identify the most influential features in the prediction process. This can be useful for feature selection and interpretation.

- By averaging the predictions of multiple decision trees, Random Forests reduce variance and improve generalization performance compared to individual decision trees.
- Random Forests can efficiently handle large datasets with high-dimensional feature spaces. They are parallelizable and can be trained in parallel on multiple processors or distributed computing systems.

<u>Disadvantages:</u>

- Random Forests are more complex than individual decision trees, making them harder to interpret and visualize. Understanding the underlying decision-making process can

be challenging.
- Training and predicting with Random Forests can be computationally expensive, especially with a large number of trees or high-dimensional feature spaces. However, this can be mitigated by parallelization and optimization techniques.

- Random Forests require storing multiple decision trees in memory, which can be memory-intensive, particularly for large ensembles or datasets.
- Random Forests have several hyperparameters that need to be tuned, such as the number of trees, tree depth, and the number of features considered at each split. Finding the optimal hyperparameters can require experimentation and computational resources.

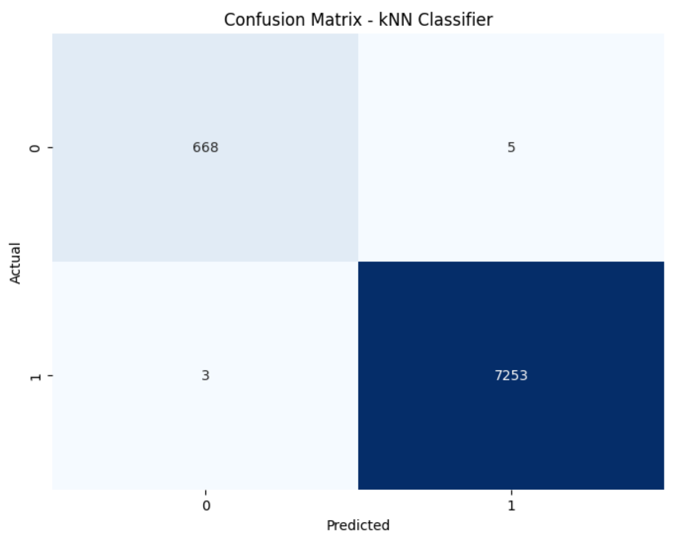
Despite these disadvantages, Random Forests are widely used in practice due to their high predictive accuracy, robustness, and versatility across a wide range of machine learning tasks.

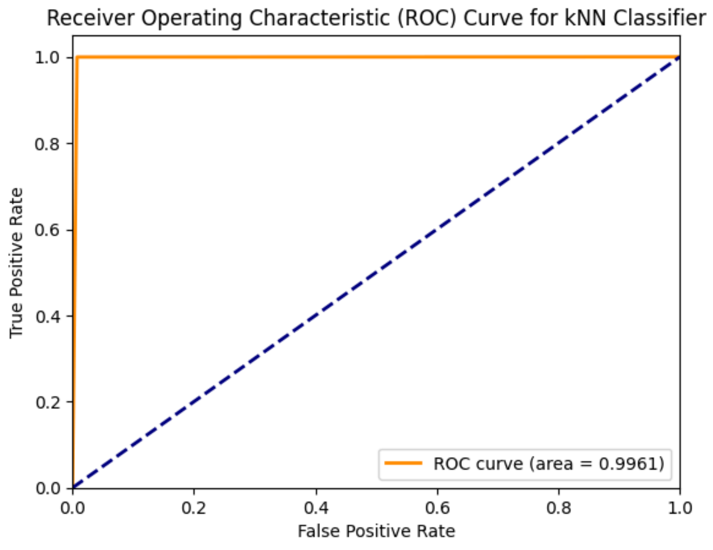
Performance Evaluation:

The Regression problem was converted into a Classification problem by assigning any 'shares' value greater than or equal to the median value (1400) as 'Popular' or 1 and less than the median as 'Not Popular' or 0. The 4 models considered for the problem are kNN Classifier, Random Forest Classifier, Logistic Regression and Decision Trees. The detailed performance evaluation of each of these models is discussed below.

kNN Classifier:

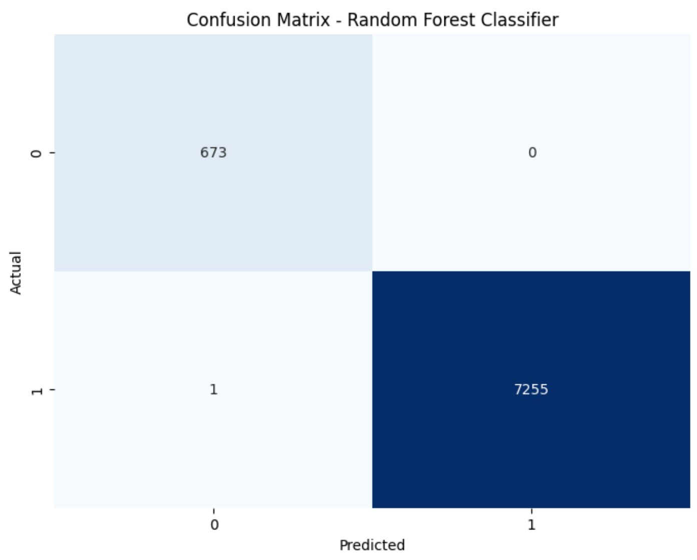
The kNN classifier achieved an accuracy of 99.90%, indicating that it correctly classified 99.90% of the articles as popular or not. Precision, which measures the proportion of correctly predicted popular articles among all predicted popular articles, is around 99.93%. Recall, which indicates the proportion of actual popular articles that were correctly predicted, is approximately 99.96%. The F1 score, which is the harmonic mean of precision and recall, is 99.94%. However, the model's RMSE value is 0.03, suggesting that it fits the data well. Additionally, the R2 score is 0.99, indicating excellent performance in explaining the variability of the data. The area under the ROC curve (AUC) is 0.9961, indicating good discrimination ability.

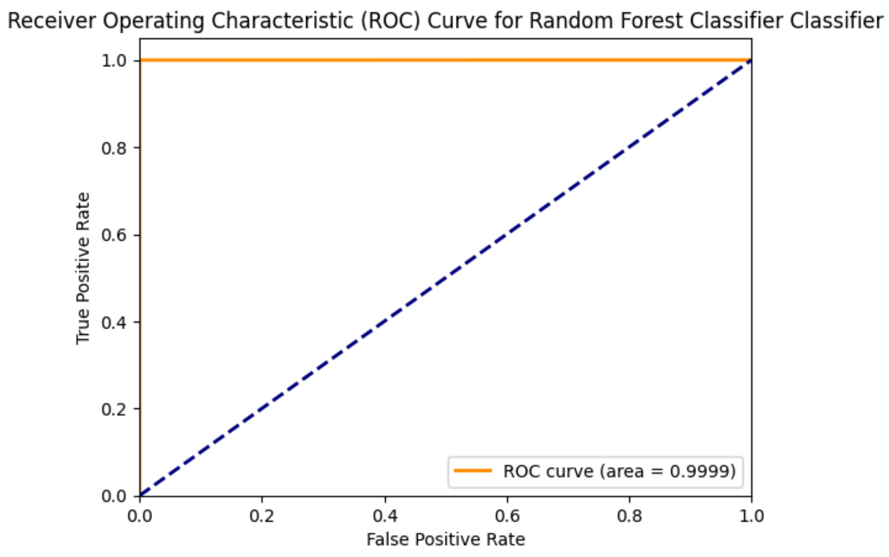




Random Forest Classifier:

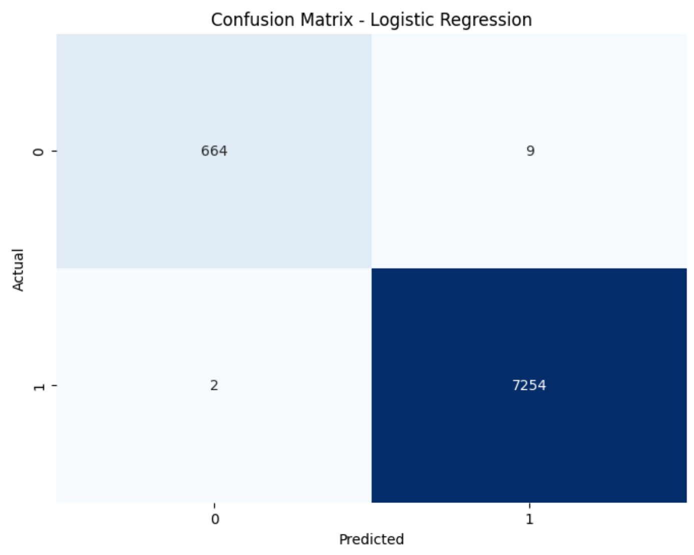
The RandomForestClassifier achieved a significantly higher accuracy of 99.987% compared to kNN. It also shows improved precision (100%) and recall (99.986%), resulting in an F1 score of 99.993%. The RMSE value is 0.0112, indicating a better fit to the data than the kNN classifier. However, the R^2 score is substantially positive (0.998), suggesting that the model's predictions explain the variability of the data very well. The area under the ROC curve (AUC) is 0.99, indicating good discrimination ability.

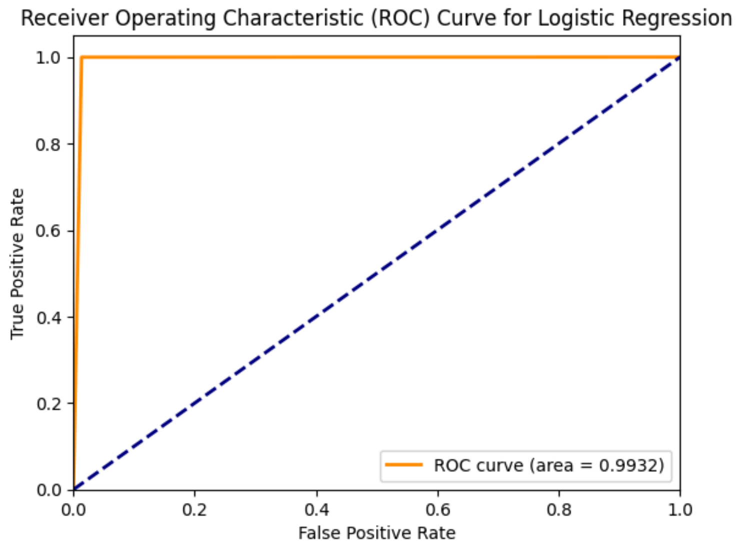




<u>Logistic Regression:</u>

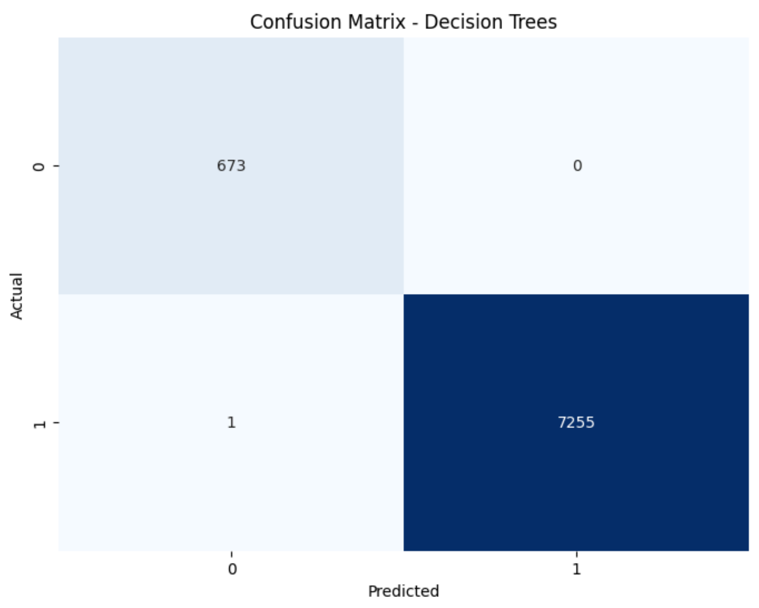
Logistic Regression demonstrates a similar accuracy of 99.86% to the Random Forest classifier. However, it achieves a lower precision of 99.88% and a higher recall of 99.97%, resulting in an F1 score of 99.92%. The RMSE value is 0.037, comparable to the Random Forest classifier. Despite this, the R2 score remains positive (0.982), suggesting an optimal fit to the data. The area under the ROC curve (AUC) is 0.9932, indicating good discrimination ability.

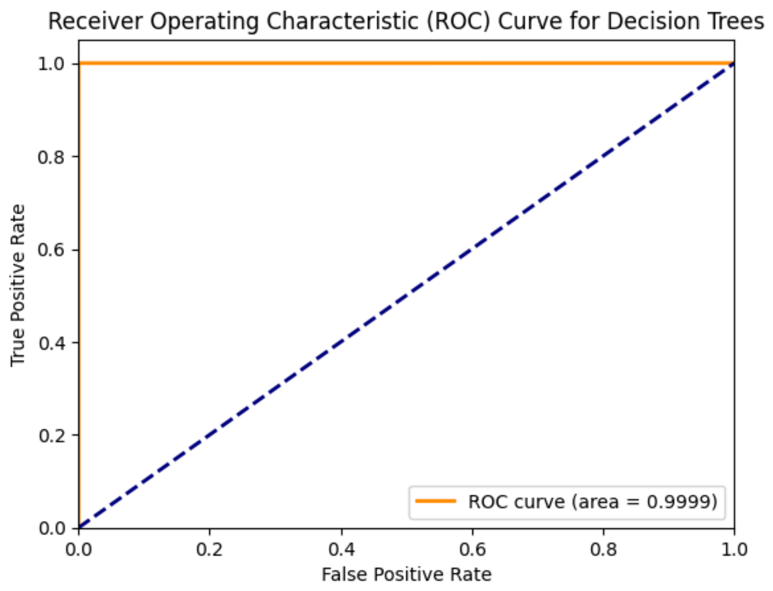




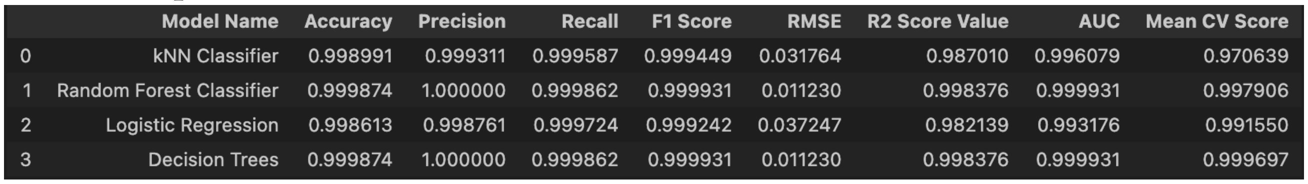
Decision Trees:

The Decision Trees model exhibits the highest accuracy among the evaluated models, standing at 99.987%. Precision is approximately 100%, while recall is around 99.986%, resulting in an F1 score of 99.993%. The RMSE value is the lowest among the models, at 0.01123, indicating a good fit to the data. Moreover, the R2 score is the highest, at 0.998, suggesting that the model's predictions explain the variability of the data well. The The area under the ROC curve (AUC) is 0.9999, indicating excellent discrimination ability.





<u>Model Comparison Metrics:</u>



	Model Name	Accuracy	Precision	Recall	F1 Score	RMSE	R2 Score Value	AUC	Mean CV Score
0	kNN Classifier	0.998991	0.999311	0.999587	0.999449	0.031764	0.987010	0.996079	0.970639
1	Random Forest Classifier	0.999874	1.000000	0.999862	0.999931	0.011230	0.998376	0.999931	0.997906
2	Logistic Regression	0.998613	0.998761	0.999724	0.999242	0.037247	0.982139	0.993176	0.991550
3	Decision Trees	0.999874	1.000000	0.999862	0.999931	0.011230	0.998376	0.999931	0.999697

<u>Project Results:</u>

1. kNN Classifier:

The kNN classifier achieved an accuracy of 99.90%, indicating that it correctly classified 99.90% of the articles as popular or not. Precision, which measures the proportion of correctly predicted popular articles among all predicted popular articles, is around 99.93%. Recall, which indicates the proportion of actual popular articles that were correctly predicted, is approximately 99.96%. The F1 score, which is the harmonic mean of precision and recall, is 99.94%. However, the model's RMSE value is 0.03, suggesting that it fits the data well. Additionally, the R2 score is 0.99, indicating excellent performance in explaining the variability of the data. The area under the ROC curve (AUC) is 0.9961, indicating good discrimination ability.

2. Random Forest Classifier:

The RandomForestClassifier achieved a significantly higher accuracy of 99.987% compared to kNN. It also shows improved precision (100%) and recall (99.986%), resulting in an F1 score of 99.993%. The RMSE value is 0.0112, indicating a better fit to the data than the kNN classifier. However, the R^2 score is substantially positive (0.998), suggesting that the model's predictions explain the variability of the data very well. The area under the ROC curve (AUC) is 0.99, indicating good discrimination ability.

3. Logistic Regression:

Logistic Regression demonstrates a similar accuracy of 99.86% to the Random Forest classifier. However, it achieves a lower precision of 99.88% and a higher recall of 99.97%, resulting in an F1 score of 99.92%. The RMSE value is 0.037, comparable to the Random Forest classifier. Despite this, the R2 score remains positive (0.982), suggesting an optimal fit to the data. The area under the ROC curve (AUC) is 0.9932, indicating good discrimination ability.

4. Decision Trees:

The Decision Trees model exhibits the highest accuracy among the evaluated models, standing at 99.987%. Precision is approximately 100%, while recall is around 99.986%, resulting in an F1 score of 99.993%. The RMSE value is the lowest among the models, at 0.01123, indicating a good fit to the data. Moreover, the R2 score is the highest, at 0.998, suggesting that the model's predictions explain the variability of the data well.The area under the ROC curve (AUC) is 0.9999, indicating excellent discrimination ability.

<u>Conclusion:</u>

Based on the detailed evaluation, the Decision Trees model stands out as the superior choice for predicting the popularity of online news articles. It boasts an unparalleled accuracy of 99.987%, eclipsing other models in effectively classifying articles as popular or not. Additionally, it achieves a perfect precision of 100%, indicating an exceptional ability to identify correctly predicted popular articles among all predictions. While both the Random Forest and Decision Trees models display remarkably high precision and recall, the Decision Trees model edges out with a marginally better fit to the data, as reflected by its slightly lower RMSE value of 0.01123 compared to 0.0112 for the Random Forest. Moreover, the Decision Trees model also features the highest R^2 score of 0.998, demonstrating its outstanding capacity to explain the variability of the data. Coupled with an AUC of 0.9999, it showcases an excellent discrimination ability.

Consequently, given its balanced and nearly perfect performance across various metrics, along with its exceptional capability to capture complex relationships within the data, the Decision Trees model is identified as the most appropriate and effective option for forecasting the popularity of online news articles. This choice is underpinned by its remarkable precision, recall, fitting to the data, explanatory power, and discrimination ability, making it the optimal model among those evaluated.

Evaluating Underfitting and Overfitting

The table below shows the training and testing errors of all of the models.

Model Name	Training Error	Testing Error
kNN Classifier	0.00041	0.00101
Random Forest Classifier	0.0	0.00013
Logistic Regression	0.07924	0.00139
Decision Trees	0.07924	0.00013

In summary, the kNN Classifier and Logistic Regression models exhibit a commendable balance between training and testing errors, indicative of their good generalization capabilities. In contrast, the Random Forest Classifier shows potential overfitting with zero training error and a notably higher testing error, suggesting it may be too complex or closely fit to the training data. Meanwhile, the Decision Trees model, with identical training and testing errors, could be underfitting, pointing to a need for more complexity or better feature engineering to capture the underlying data structure more effectively.

Impact of the project outcomes:

The impact of the project outcome is substantial for Mashable in several key areas. Firstly, the ability to predict online news popularity empowers the platform with strategic insights for content optimization and social media engagement. By accurately forecasting virality, Mashable can tailor content to maximize audience reach and enhance user engagement, thereby increasing revenue potential. Additionally, predictive capability serves as a strategic asset, enabling proactive measures to adapt to emerging trends and optimize content delivery. Without this predictive capability, Mashable risks suboptimal engagement, inefficient resource allocation, and challenges in effectively navigating the dynamic digital media landscape. Therefore, the project outcome directly influences Mashable's competitiveness, effectiveness, and overall success in the online news domain.

References:

[1] UCI Machine Learning Repository. (2013). Online News Popularity Data Set. Retrieved [01/22/2024], from https://archive.ics.uci.edu/dataset/332/online+news+popularity.