Topic: Predicting News Article Popularity using Machine Learning Models: A Comparative & Predictive Analysis

1. Data description and research question

In this fast content sharing world, the online news industry became increasingly competitive. Because of high volume, veracity, velocity and variety of data made difficult to understand the target market. The more the prediction would be accurate the more will be the benefits to the news companies to optimize the content making strategy, improve the user engagement, and increase the advertising revenue.

This study aims to answer the question, "Can we predict the popularity of online news articles using a combination of content features and user engagement metrics?" We'll use a dataset of online news items from Mashable, a well-known digital media outlet, to provide an answer to this query. The dataset includes a wide range of features reflecting user engagement measures, such as the number of shares on social media sites, as well as variables representing the content of the articles, such as the amount of words, links, images, and videos.

We hope to find the most essential elements that contribute to the popularity of online news stories by analysing this dataset and developing a predictive model that can reliably evaluate an article's popularity based on these features. The findings of this study have the potential to provide news publishers and content creators with useful information, allowing them to generate more engaging material that resonates with their target audience.

During this project, we will analyse the dataset thoroughly, pre-process the data, and apply several machine learning techniques to construct a prediction model. We will also evaluate the model's performance with relevant assessment measures and interpret the results to draw valuable insights. Finally, we will explore the limitations of our technique and potential future research directions in this subject.

Mashable, a well-known digital media platform, provided the dataset, which includes 39,644 online news articles. The dataset includes 59 features that are relevant to our research question, "Can we predict the popularity of online news articles using a combination of content features and user engagement metrics?" These factors assist us in understanding many aspects of the articles, including as their content, user involvement, and metadata, which can all contribute to their popularity.

Content features in the dataset include the number of words in the title (n_tokens_title) and content (n_tokens_content), rate of unique words (n_unique_tokens), rate of non-stop words (n_non_stop_words), and rate of unique non-stop words (n_non_stop_unique_tokens). Other content features consist of the number of links (num_hrefs), self-references (num_self_hrefs), images (num_imgs), and videos (num_videos), as well as average token length (average_token_length) and metadata keywords (num_keywords). These features help us quantify the articles' content complexity and richness.

User engagement is measured using features such as the number of shares (shares), which serves as our target variable. Metadata features include information about the publication date, such as whether the article was published on a weekday (weekday_is_monday to weekday_is_friday) or a weekend (weekday_is_saturday, weekday_is_sunday, is_weekend). Additionally, the dataset includes binary variables for the data channel (data_channel_is_lifestyle to data_channel_is_world) and LDA topic modeling (LDA_00 to LDA_04), allowing us to explore the relationship between article topics and their popularity.

Sentiment and subjectivity measures, including text and title sentiment polarity (global_sentiment_polarity, title_sentiment_polarity), subjectivity (global_subjectivity, title_subjectivity), and rates of positive and

negative words in the content (global_rate_positive_words, global_rate_negative_words), provide insight into how the emotional tone of articles may impact their popularity.

To better understand the distribution and relationships among features, we will conduct an exploratory data analysis. We will create visualizations such as histograms, scatter plots, and correlation matrices to examine feature distributions, detect outliers, and identify potential relationships and multicollinearity issues among variables. This analysis will help us gain insights into the data, inform our feature selection process, and guide the development of our predictive model.

2. Data preparation and cleaning

In this data preparation and cleaning section, we followed several steps to ensure the quality of the dataset and to facilitate further analysis. Here are the steps that we followed to prepare our data for further analysis:

1. **Missing Values and Type Conversion**: Firstly, we dropped the "url" and “timedelta” column, as it does not provide any relevant information for our research question. This step reduces the dimensionality and simplifies our analysis. Upon analysing the dataset, potential issues with outliers, datatype misinterpretations, and implausible values were identified. To address these concerns, missing values were first examined, revealing that none of the features contained missing data, thereby eliminating the need for handling them. However, Python misinterpreted the categorical features as numerical, so their datatypes were corrected accordingly. While this resolved some issues, it was essential to further investigate implausible values and other data cleaning steps for a comprehensive and reliable analysis.

2. **Implausible Values Handling**: During the outlier handling process, plausible ranges were defined for each feature based on domain knowledge and the nature of the features. For instance, rates such as 'n_unique_tokens', 'n_non_stop_words', and 'n_non_stop_unique_tokens' were limited between 0 and 1, as they represent proportions that cannot be less than 0 or greater than 1. 'Average_token_length' was assigned a minimum value of 1, considering the shortest possible word length is one character. For the share-related features, 'kw_min_min', 'kw_avg_min', and 'kw_min_avg', the minimum value was set to 0, as shares cannot be negative.

   Upon comparison of these ranges with the dataset, it was discovered that seven features contained values outside their respective defined ranges. Imputation was performed on these features to bring their values within the plausible limits, ensuring that the dataset becomes more reliable and robust for further analysis.

3. **Detecting Outliers**: After handling the implausible values, the next step was to assess the potential impact of outliers on the dataset. To do this, the Interquartile Range (IQR) method was applied to the continuous features. This involved calculating the first quartile (Q1), the third quartile (Q3), and the IQR (Q3 - Q1) for each continuous feature. Outliers were then identified as data points that fell below the lower limit (Q1 - 1.5IQR) or above the upper limit (Q3 + 1.5IQR).

   The IQR method was chosen due to its robustness against extreme values, which helps minimize the influence of potential errors or noise in the data. By identifying and analysing the number of outliers in each feature, we can gain insights into the distribution of the data and determine whether further outlier handling is necessary.

   Upon analysing the output, it was found that many features contained a substantial number of outliers. However, it's essential to evaluate these findings critically, as these values could be genuine observations that are inherent to the dataset. Blindly removing or modifying the identified outliers may lead to the loss of valuable information, which could affect the performance of any subsequent analysis or predictive models.

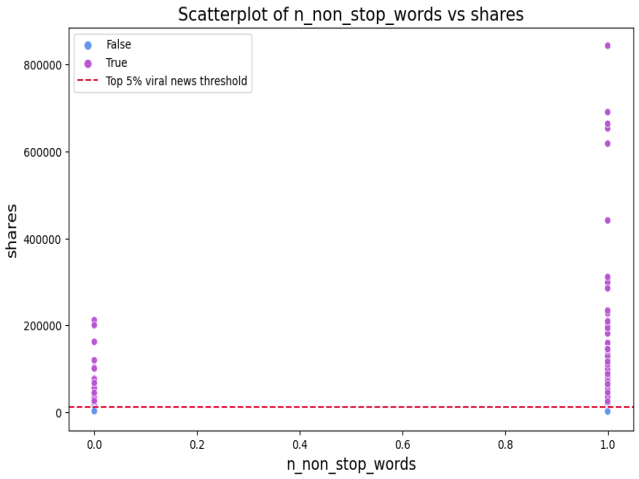
Given the nature of the dataset, it's possible that some features inherently have skewed distributions or extreme values. In such cases, it's crucial to investigate the data further to determine whether these values are genuine or due to data entry errors, measurement errors, or other factors.

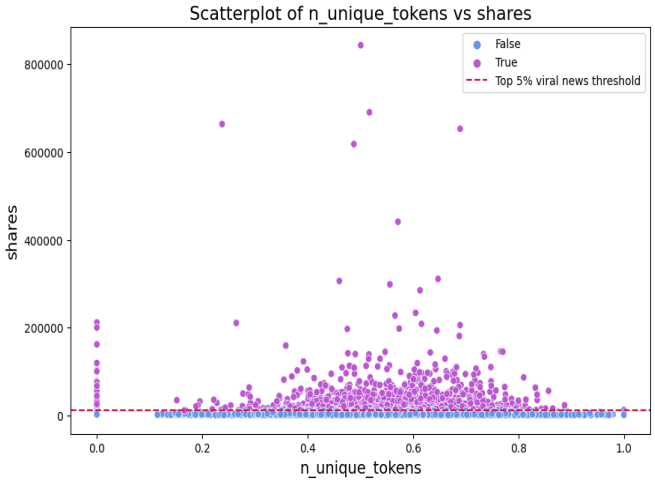
4. **Handling Outliers - Critical Evaluation and Insights**

   i. **Retaining valuable information**: Some features, such as 'num_imgs', 'num_videos', and 'title_sentiment_polarity', have a high number of outliers. These features might be important in understanding the virality of an article since multimedia content and sentiment could affect how readers engage with and share the articles. By retaining these outliers, we can ensure that our analysis captures the full range of values and potential factors that could influence article virality.

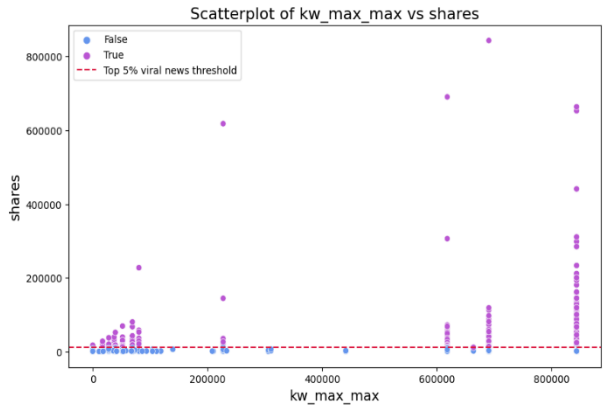
   ii. **Outliers in 'shares'**: There are 4541 outliers in the 'shares' feature, which could be essential in determining the factors driving the virality of news articles. If we remove these outliers, we might miss some critical information that could be valuable for our analysis. Comparing these outliers to the top 5% most viral news articles can provide more in-depth insights into the characteristics of highly viral articles.

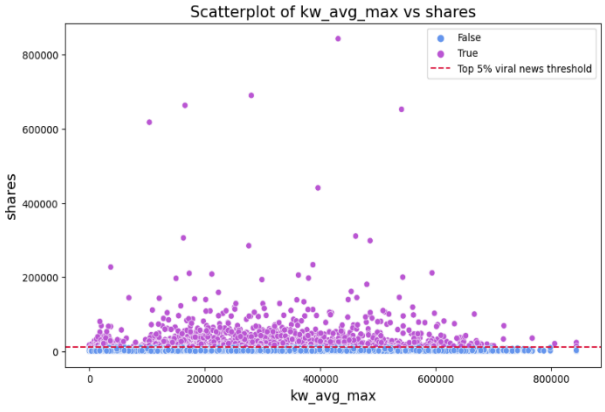
   iii. **Outliers in content-related features**: Features like 'n_tokens_content', 'n_unique_tokens', and 'n_non_stop_words' contain a significant number of outliers, which might indicate that these characteristics have some impact on an article's virality. For example, it could be that articles with a large number of unique tokens or non-stop words are more engaging, which could make them more likely to be shared. By retaining these outliers, we can study the impact of these content-related features on article virality more accurately.





   iv. **Other features with a high number of outliers**: Features such as 'kw_min_max', 'kw_max_max', and 'kw_avg_max' have a considerable number of outliers. This could suggest that keyword usage plays an essential role in article virality. By keeping these





outliers, we can analyze the potential influence of trending keywords and their impact on article shares.

Retaining the outliers in the dataset can help capture the full range of factors that could influence the virality of news articles. By comparing these outliers to the top 5% most viral news articles and analyzing the differences between them, we can gain valuable insights into what makes a news article go viral. Furthermore, we will perform normalization and transformation on the dataset to reduce the effects of these outliers, ensuring that our analysis remains robust and reliable. By combining these approaches, we can conduct a comprehensive analysis to better understand the factors contributing to article virality.

5. **Feature scaling**: We used the Box-Cox method for feature scaling, as it is effective in reducing the heteroscedasticity and ensuring a more normal distribution of the data. The method can help mitigate the impact of outliers and improve the performance of our models. Other feature scaling techniques, such as normalization or standardization, were not selected in this case because they do not specifically address the issues of heteroscedasticity or skewed data distributions that are present in our dataset.

   After applying the Box-Cox transformation, we observed that some of the variables have fewer outliers compared to before applying the method, while others still have a similar number of outliers. This can be attributed to the fact that the Box-Cox transformation works optimally when the data is approximately normally distributed or slightly skewed. If the initial distribution of a feature is heavily skewed or has a high degree of kurtosis, the transformation may not be as effective in reducing the number of outliers.

   The presence of these outliers after transformation could have an impact on our ability to answer the research question effectively. However, we will apply transformation techniques to further reduce the biasness due to outliers. In addition, by critically evaluating the dataset and understanding the limitations and assumptions of our analysis, we can still extract valuable insights about the factors contributing to the virality of news articles. However, the results will be interpreted with caution, considering the potential influence of outliers on our findings.

Throughout the data preparation and cleaning process, we critically analyse each step to ensure that we are not introducing bias or errors into our analysis. By taking these steps, we have created a clean and well-structured dataset, paving the way for a robust exploratory data analysis and modelling phase.

3. Exploratory data analysis

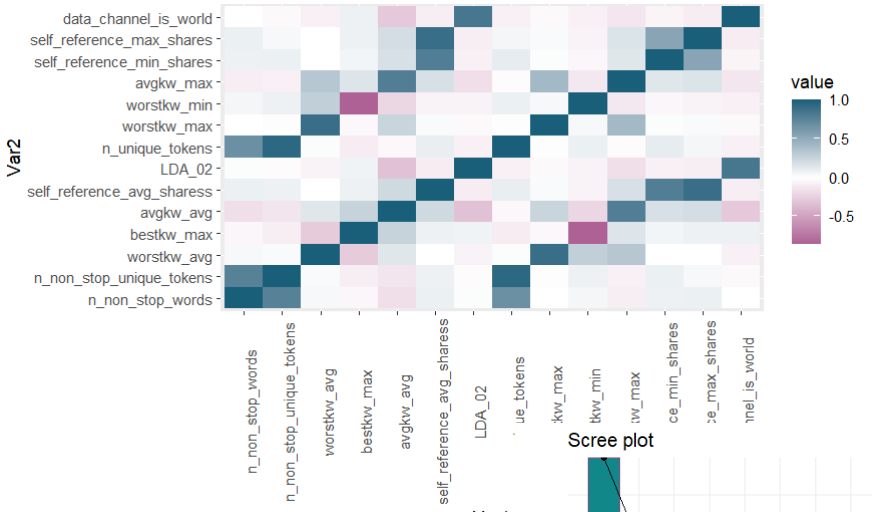
   In the Exploratory Data Analysis (EDA) section, our goal is to uncover patterns and insights from the dataset, which will help in understanding the factors influencing the virality of news articles. The EDA process will start with data visualization to examine the distribution and relationships among variables. We will then perform feature engineering, including Principal Component Analysis (PCA) and correlation analysis, to identify the most important features and reduce dimensionality. Next, we will apply clustering techniques, specifically K-means clustering, to find underlying patterns and groups in the data. Throughout the EDA, we will apply appropriate transformations to address skewness and reduce the impact of outliers on the analysis.

   By following this approach, we aim to identify key factors and trends that drive news article virality, enabling a deeper understanding of the dataset and facilitating the development of effective strategies to increase the likelihood of virality.

I. Feature Engineering

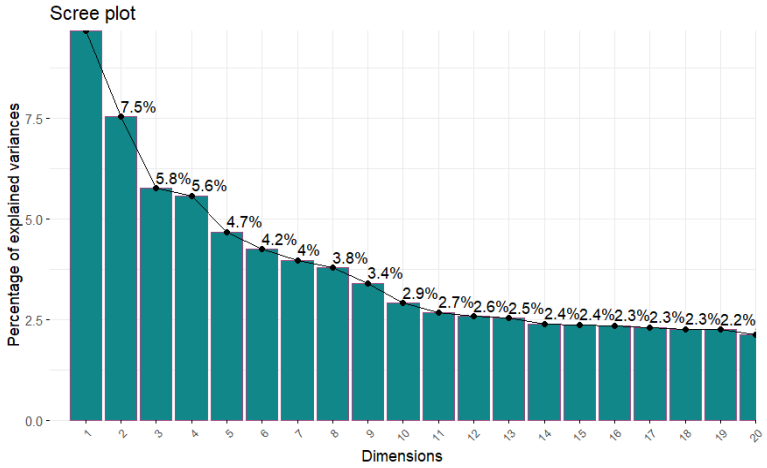
In the Feature Engineering step, we performed dimensionality reduction, correlation analysis, and data transformation to optimize our dataset.

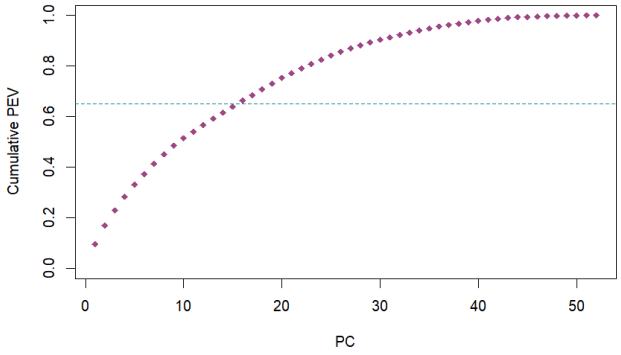
a. **Correlation Analysis**: We analyzed the correlation matrix graph and identified variables with high correlation (above 80%) with other independent variables, such as data_channel_is_world, n_non_stop_words, n_non_stop_unique_tokens, worstkw_avg, worstkw_min, avgkw_avg, and



self_reference_avg_shares. We removed these variables to reduce multicollinearity, as it might interfere with the model's performance. From the given figure it can clearly seen that which the variables are highly corelated. On this basis, we picked one feature out of two, based on domain knowledge.

b. **PCA**: To further reduce the dataset, we applied PCA with a threshold of 65% variance, which resulted in 17 principal components. The choice of 65% threshold was to strike a balance between retaining meaningful information and significantly





PC1	PC2	PC3
"global_subjectivity"	"rate_negative_words"	"bestkw_avg"
PC4	PC5	PC6
"n_unique_tokens"	"LDA_00"	"title_subjectivity"
PC7	PC8	PC9
"is_weekend"	"data_channel_is_entertainment"	"worstkw_max"
PC10	PC11	PC12
"LDA_02"	"self_reference_max_shares"	"bestkw_max"
PC13	PC14	PC15
"max_negative_polarity"	"weekday_is_wednesday"	"weekday_is_tuesday"
PC16	PC17	
"weekday_is_thursday"	"weekday_is_friday"	

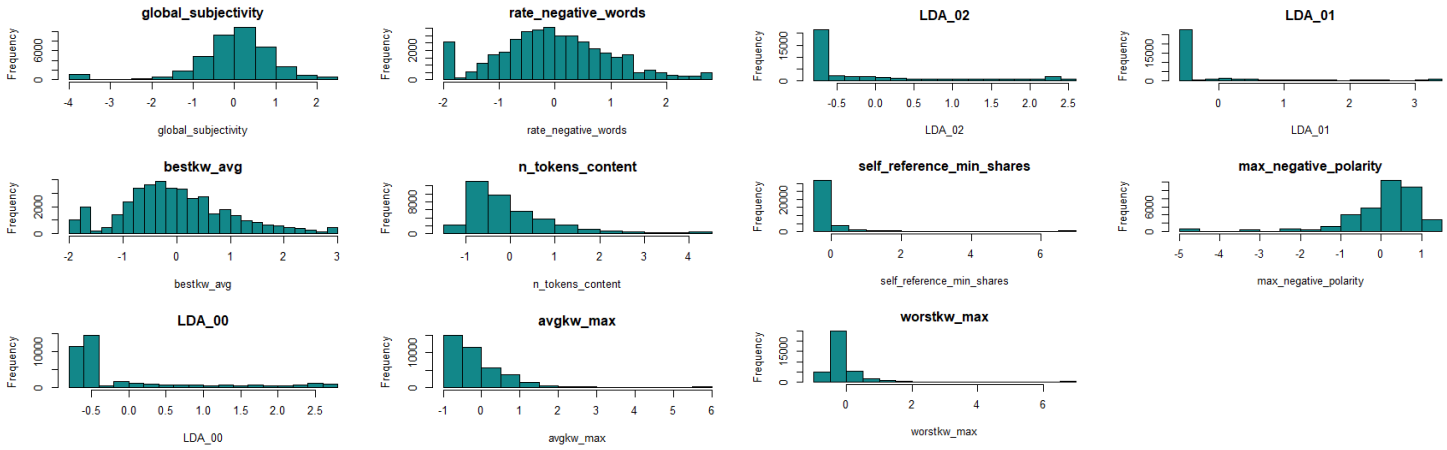
reducing the number of features to build a more efficient model. The 17 PCs captured essential variables such as global_subjectivity, rate_negative_words, bestkw_avg, and others, which could be important predictors for our research question.

c. **Data Transformation**: We transformed the variables based on their distribution to reduce the effect of outliers and computational resource consumption. For normally distributed features (global_subjectivity, rate_negative_words, avgkw_max), we applied Z-score standardization to reduce scaling. Right-skewed variables (n_tokens_content, LDA_00, LDA_01, LDA_02, self_reference_min_shares, worstkw_max, shares) underwent Log1p transformation, while left-skewed variable max_negative_polarity was transformed using the square transformation method.

These Feature Engineering steps were crucial to streamline the dataset, minimizing the impact of multicollinearity, reducing computational resources, and tackling outliers.

II. Insights from Data Visualization

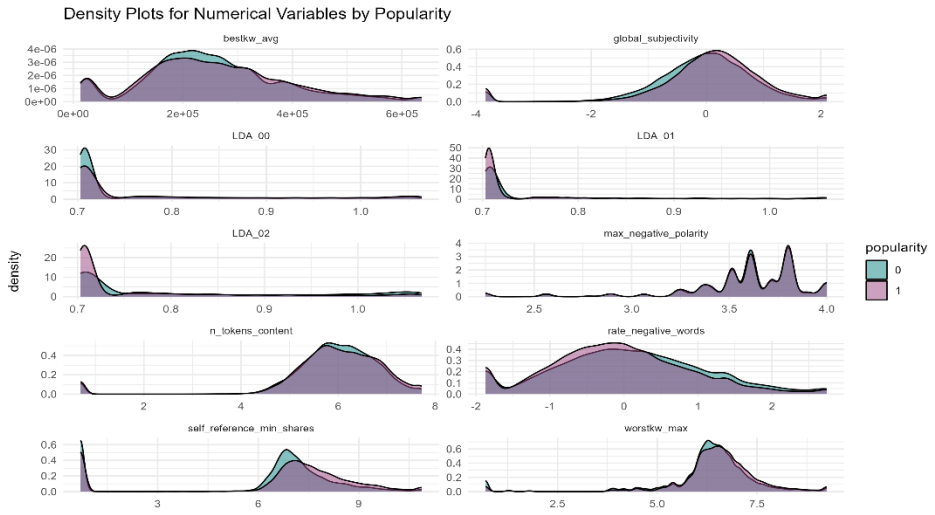
- **Univariate analysis**:



From the graphs, it can be interpreted that 'global_subjectivity', 'rate_negative_words' and 'bestkw_avg' are normally distributed. However, 'n_tokens_content', 'LDA_00', 'avgkw_mx', 'LDA_02', 'LDA_01' and 'self_reference_min_shares' are right-skewed distributed and 'max_negative_polarity' is left-skewed distributed. However, because of transformation performed in the previous step will reduce the effect of the skewed distribution.

- Multivariate analysis:

To analyze the frequency of popular news with the channel genre or day of publication, we created new column named 'popularity' and set the threshold value to 1400. If the number of shares went more than 1400 then it is popular (appending 1 in the popularity column) other, it is not popular (appending 0 in the popularity column). The graph indicates that news articles are more popular on weekends, possibly due to increased leisure time allowing for greater news consumption. During weekdays, especially Monday through Wednesday, the difference between popular and unpopular articles is minimal, suggesting that factors like content relevance, publication time, or topic prominence could influence virality. The weekend popularity surge might also be due to certain news categories, like entertainment or leisure, being more engaging for readers. Publishers should consider these trends when optimizing their content distribution strategy, targeting weekends for viral stories, and focusing on factors to improve article performance on weekdays.



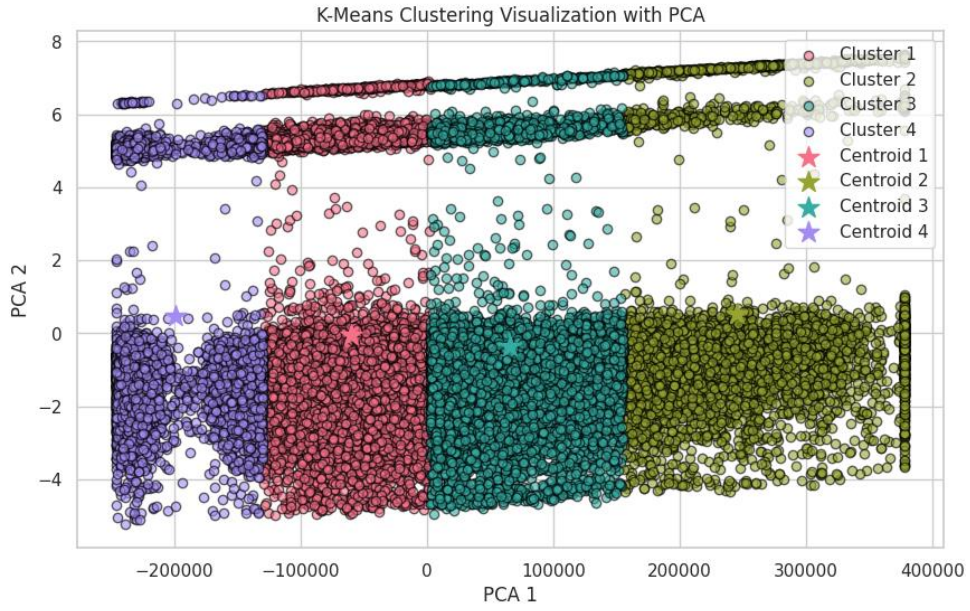
From the density plot,

1. LDA_00 seems to have a stronger relationship with popularity, as the non-popular articles have a higher peak compared to the popular ones. This suggests that LDA_00 might be a useful predictor of popularity.
2. LDA_02 and LDA_01 also appear to have some relationship with popularity, as the popular articles have higher peaks compared to the non-popular ones. These features could potentially be helpful in predicting article popularity.
3. Max_negative_polarity does not seem to have a clear relationship with popularity, as the distributions for popular and non-popular articles almost completely overlap.
4. Self_reference_min_shares has a higher peak for non-popular articles, and a flatter peak for popular ones, indicating that it might also be related to article popularity.
5. For other variables, the density plots show almost overlapping distributions for popular and non-popular articles, suggesting that they might not have strong relationships with popularity.

III. Clustering

After implementing K-means clustering, we aimed to identify patterns and group the dataset into similar clusters based on their properties or behavior. By applying the elbow method, we determined the optimal number of clusters (k) to be 4. We fitted the KMeans object with k=4, obtaining the following cluster distribution: Cluster 1: 6015, Cluster 2: 17418, Cluster 3: 4153, and Cluster 4: 12058.

To visualize the clustering, we reduced the dimensionality using PCA with two principal components, generating a new dataset with 'pca1' and 'pca2' columns. We also computed the centroids for each cluster using the attributes in the dataset.



Based on the further analysis of cluster centroids, and the cluster distribution, we can infer the following insights:

- The dataset can be divided into four distinct clusters, each potentially representing a different group of news articles. These groups could be characterized by factors such as the type of content, time of publication, or the prominence of the topic in the media landscape.
- Each cluster's centroid gives us valuable information about the average behavior of the features within each cluster. For example, Cluster 1 has a higher average `global_subjectivity` than other clusters, suggesting that the articles in this cluster have a more subjective tone. Cluster 2 has the highest

`bestkw_avg` value, indicating that the articles in this cluster have higher average keyword popularity. Comparing these centroids can help us identify the distinctive features of each cluster.

- The varying sizes of the clusters suggest that some groups of articles share more common properties than others. For instance, Cluster 2, being the largest cluster, might represent a set of articles with a high degree of similarity in terms of their attributes, whereas Cluster 3, the smallest cluster, might represent articles with more unique characteristics.
- By examining the attributes of articles within each cluster, we may be able to determine the factors that contribute to an article's popularity. For example, we can analyze the content type, publication time, or the topic prominence for articles within each cluster to identify patterns and trends that lead to higher popularity.
- Analyzing the centroid values for `weekday_is_wednesday`, `weekday_is_tuesday`, and `weekday_is_monday` reveals that articles published on these days have slightly different attributes on average. This information can be helpful for publishers to optimize their content distribution strategies according to the day of the week.

These insights from K-means clustering can be utilized to better understand the factors that influence news article popularity, ultimately helping publishers optimize their content distribution strategies and improve article performance. By identifying the distinguishing features of each cluster, we can shed light on the underlying patterns in the dataset and answer our research question more effectively.

Insights, Findings & New Knowledge discoveries:

The new knowledge discoveries from the EDA and data cleaning process can be summarized as follows:

- **Distinct Clusters**: The dataset can be divided into four unique clusters, each representing a different group of news articles, potentially characterized by factors such as content type, publication time, or topic prominence.
- **Cluster Characteristics**: The cluster centroids reveal distinct characteristics for each group, providing insights into the unique attributes of each cluster that can be leveraged when building ML models.
- **Important Features**: Through feature selection and interaction effects analysis, we can identify the most relevant features for predicting the number of shares, which can improve the performance of the regression model.
- **Outliers**: Detection and handling of potential outliers can improve the accuracy and generalizability of the ML model.

These discoveries provide a deeper understanding of the dataset and can be used to guide the development of a more effective ML model for predicting news article popularity.

4. Machine Learning Models Implementation & Evaluation

This section discusses about the implementation of Machine Learning models by me. The models I picked were Support Vector Regressor (SVR), Random Forest for regression (RF) and Linear Regression (LR).

Each model is applied on taking the sample of data to limited computation resources. I used Stratified Sampling technique to reduce the biasness while training our model.

1. **Support Vector Regressor (SVR):** A powerful non-linear regression technique derived from Support Vector Machines (SVM). SVR can handle large and complex datasets effectively and has been widely used for various regression tasks.
   Data Splitting:
   Due to limited computational resources, we employed simple random sampling to divide our dataset into training and testing sets. Although stratified sampling is preferred to maintain the proportion of each target variable class, simple random sampling was used to reduce the computational burden. This method still provided a reasonable representation of the original data in the training and testing sets, which helped improve the model's generalization and reduce the possibility of overfitting. However, it is important to note that using simple random sampling may introduce some bias in the data split, potentially impacting the model's performance.
   Model Implementation:

We implemented the SVR model in R using the svm() function from the e1071 package. We used both Polynomial and Radial Basis Function (RBF) kernels for our models. The model was trained on 70% of the dataset, and its performance was evaluated on the remaining 30%. The training process and model prediction timings were also measured.

```
## SVR modeling

```{r}
start_time <- Sys.time()
svr_model <- svm(shares ~ ., data = train_data, kernel = "radial", type = "eps-regression")
end_time <- Sys.time()
time_taken <- end_time - start_time
time_taken
```

Time difference of 3.639575 mins
```

Hyperparameter Tuning and Model Selection

We performed 5-fold cross-validation for both SVR models with different kernel types to select the optimal hyperparameters. For the Polynomial kernel, the final values were degree = 3, scale = 0.1, and C = 0.1, while for the RBF kernel, the final values were sigma = 0.1 and C = 1. The optimal models were chosen based on the smallest Root Mean Squared Error (RMSE).

Model Evaluation:

```
Poly Kernel Model Performance:
MSE: 0.433518
RMSE: 0.6584209
R-squared: 0.6408816
MAE: 0.3786656

RBF Kernel Model Performance:
MSE: 0.6291202
RMSE: 0.793171
R-squared: 0.4788484
MAE: 0.4447944
```

Based on these results, the Polynomial kernel SVR demonstrated better performance in predicting the number of shares for news articles compared to the RBF kernel SVR.

Insights:

Considering the use of simple random sampling, hyperparameter tuning, and resulting performance metrics, the SVR model with a Polynomial kernel exhibits better predictive accuracy compared to the RBF kernel, indicating its suitability for capturing non-linear relationships in the dataset. However, potential biases introduced by simple random sampling, which could result in an unbalanced representation of the target variable classes and affect model performance, should be considered. Investigating the impact of stratified sampling on performance in future studies may be beneficial.

2. **Linear Regression (LR):**

Linear regression is a widely used regression technique that tries to find the best-fitting line through the data points. In our case, the target variable is the number of shares for news articles. Here, we took the whole dataset because linear regression is simple model and doesn't require much computational resources.

Model Implementation:

```
# Create the linear regression model with the same parameters
alpha <- 1
lambda <- 0.1
fit <- glmnet(x_train_lr, y_train_lr, alpha = alpha, lambda = lambda)

# Make predictions on the test dataset using the model
predictions <- predict(fit, x_test_lr)

# Evaluate the model's performance (Root Mean Squared Error)
rmse <- sqrt(mean((y_test_lr - predictions)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))
```

```
[1] "Root Mean Squared Error (RMSE): 0.1006321464916"
```

We implemented the linear regression model in R using the glmnet package, which provides an interface for generalized linear models with Lasso, Ridge, or Elastic Net regularization. Initially, we set alpha to 1 and lambda to 0.1. Using these parameters, we trained the model on 80% of the data and evaluated its performance on the remaining 20%.

```
Linear Regression Model Performance:
MSE: 0.01012683
RMSE: 0.1006321
R-squared: 0.9883991
MAE: 0.07669861
```

With the above implementation, we got the good performance with normal parameters. However, need to check whether it will perform of better using Glmnet model (Hyperparameter tunning).

Hyperparameter Tuning:

```
Tuned glmnet Model Performance:
MSE: 3.502165
RMSE: 1.871407
R-squared: -216.5847
MAE: 1.368233
```

To further improve the model's performance, we performed hyperparameter tuning using the caret package in R. We used 5-fold cross-validation and a grid search technique to find the optimal values for the alpha and lambda parameters. The search space for alpha was 0, 0.5, and 1, while for lambda, it was 0.01, 0.1, and 1. The choice of 5-fold cross-validation was made to balance the trade-off between computational efficiency and the reliability of the performance estimates. With 5-folds, each fold is used once as a validation set while the remaining 4-folds are used for training, providing an average estimate of the model's performance. Despite our efforts in hyperparameter tuning, the model's performance metrics, such as R-squared, indicate that the current model is underperforming. Therefore, further improvements are needed, or maybe the best parameter would be the one we used before Hyperparameter tunning. The search ranges for alpha and lambda were chosen to cover a broad range of regularization possibilities, from no regularization to a strong regularization effect.

Model Evaluation:

The initial Root Mean Squared Error (RMSE) of the model, before hyperparameter tuning, was 0.1006. However, after the experimenting with the parameters, we found the optimal hyperparameters to be alpha = 0.5 and lambda = 0.01. We updated our model with these hyperparameters and retrained it. The new RMSE after hyperparameter tuning was 0.0100, which is a little improvement over the initial model's performance. However, in hyperparameter tunning we got the least performing model.

To conclude, the EDA and cleaning process helped us to understand the working of models. Furthermore, the insights helped optimize the SVR and LR models used in this project, reducing the error in prediction and potential issues.

5. High Performance Computational implementation

In this study, we implemented the High-Performance Computing Integration (HPCI) technique in supervised learning models to improve the efficiency and scalability of our analysis. HPCI allows us to handle large-scale datasets and complex computations more effectively, which is essential for machine learning tasks.

We used PySpark to implement the HPCI technique due to its ability to process large datasets in parallel across distributed systems. This improves both the speed and scalability of our analysis. We chose the Random Forest regression model for this task and compared the results obtained in PySpark with those from R.

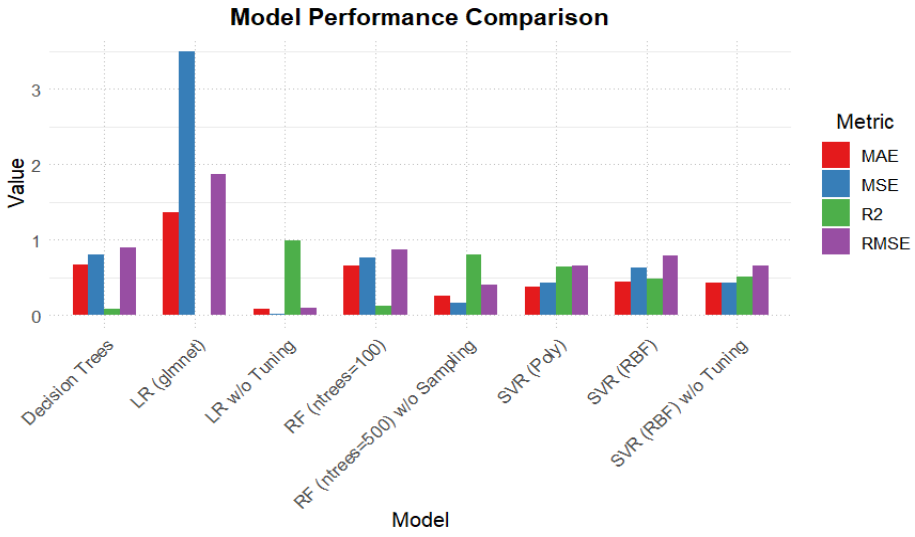
The PySpark implementation code and results are as follows:

Implementation	Execution Time (Training)	Root Mean Squared Error (RMSE)	Mean Absolute Error (MAE)
PySpark	50.26 seconds	0.8758	0.6547
R	11.37 minutes	0.8718	0.6565

Comparing the results, we can observe that the HPCI technique significantly reduces the execution time for training the Random Forest model. The PySpark implementation took only 50.26 seconds compared to 11.37 minutes in R, demonstrating the benefits of using HPCI in terms of computational efficiency. The

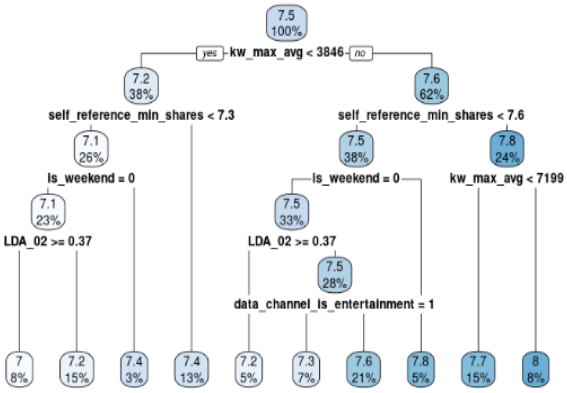
performance metrics, RMSE and MAE, are comparable between the two implementations, indicating that the HPCI technique does not compromise model accuracy while providing improved computational performance. Therefore, employing HPCI techniques can be beneficial for training large and complex models.

6. Performance evaluation and comparison of methods



From the visualization it can be said that LR (glmnet) is worst performing and the LR (without tuning) can give more accurate results when test on other online news. However, there could be the possible biasness.

1. Support Vector Regression (Poly Kernel): This model performs well when the relationships between features and the target variable are non-linear. The Polynomial kernel's ability to capture non-linearities makes it more suitable for such scenarios. It outperforms the RBF kernel in this case, likely because the Polynomial kernel better captures the underlying structure in the dataset.
2. Support Vector Regression (RBF Kernel): The RBF kernel is more flexible than the Polynomial kernel, making it capable of modeling complex, non-linear relationships. However, it underperforms compared to the Polynomial kernel in this dataset, indicating that the data may have a polynomial-like structure or the RBF kernel's flexibility is not advantageous here. The model may perform better on other datasets with more complex relationships.
3. Linear Regression (without tuning): The untuned Linear Regression model performs exceptionally well, suggesting that there is a strong linear relationship between the features and the target variable. This model would excel in scenarios where the relationships between variables are primarily linear, making it easier to interpret and less computationally expensive than other models.
4. Linear Regression (glmnet): The poor performance of the glmnet model implies that the regularization technique used by glmnet might be too aggressive for this dataset, causing the model to underfit the data. This method could perform better in cases where there is a high risk of overfitting or the presence of multicollinearity.
5. Random Forest Regressor (ntrees=100): This model's performance is not ideal, which could be due to the limited number of trees in the ensemble. It might perform better in situations where fewer trees are sufficient to capture the underlying structure or when computational resources are limited.
6. Random Forest Regressor (ntrees=500) (without sampling): This model performs well, suggesting that increasing the number of trees improves its predictive accuracy. It can be an excellent choice for cases where the data contains a mix of linear and non-linear relationships, and the additional trees help capture complex patterns. Random Forest is also robust to overfitting and handles noisy data better than other models.
7. Decision Trees: This model performs the worst among all, likely because it is prone to overfitting and is sensitive to small changes in the data. It might perform better on simpler datasets or when



combined with other methods, such as in ensemble techniques like Random Forest. It shows that 'kw_max_avg' is the major determining factor to answer our research question.

In conclusion, the choice of the best model depends on the nature of the relationships within the dataset and the specific requirements of the problem. The Linear Regression model performs best for linear relationships, while the SVR (Poly Kernel) and Random Forest Regressor (ntrees=500) are better suited for non-linear patterns. It is crucial to consider factors such as interpretability, computational resources, and the risk of overfitting when selecting a model.

7. Discussion of the findings

The analysis of various machine learning models for predicting news article popularity highlighted the superior performance of untuned Linear Regression, Support Vector Regression (SVR) with a Polynomial kernel, and Random Forest Regressor with ntrees=500.

Incorporating insights from the Exploratory Data Analysis (EDA) and data cleaning processes can improve model accuracy, as seen in the case of the SVR model with a Polynomial kernel. High-Performance Computational implementation using PySpark showcased significant time reduction in training the Random Forest model without compromising accuracy. This finding underscores the benefits of employing High-Performance Computing Integration (HPCI) techniques in large and complex models.

Future studies should investigate the impact of stratified sampling on model performance and explore additional ensemble techniques like gradient boosting to further enhance prediction accuracy. These considerations will help develop more effective machine learning models for predicting news article popularity, catering to various scenarios and addressing the unique challenges of the problem at hand.

8. Data Management Plan and Author Contribution statement

In this Distributed Data Analysis project, the data collection was a collective effort by the group and formed the research question in the process. Amit Kedia performed data cleaning. and implemented Linear Regression, Support Vector Regressor, and Random Forest Regressor (500 trees) models. Sooraj Krishnakumar handled data preparation and applied Random Forest Regressor (100 trees) and Decision Tree models. Payal Parida and Payalben Patel jointly conducted the Exploratory Data Analysis, implemented PCA, K-means and visualisations.